English

FUJITSU Software BS2000

interNet Services V3.4B

User Guide

Edition June 2017

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:
manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2008

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2008.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Contents

1 Preface

The interNet Services product supplements the TCP/IP functionality of openNet Server with the following standards:

- DNS Resolver and Server
- NTP Client and Server
- FTP Client and Server
- TELNET Client and Server
- OpenSSH
- Mail Sender in BS2000
- Mail Reader in BS2000
- Mail Server in POSIX

1.1 Target group

This manual is intended for BS2000 users and system administrators who want to use TCP/IP-specific applications and/or services. Knowledge of the BS2000 OSD/BC operating system and the basic concepts of TCP/IP is therefore assumed. Apart from this User Guide, there is also an Administration manual for interNet Services, which contains the relevant information for system and network administrators.

1.2 Summary of contents

This manual is laid out as follows:

- Chapter 2: Overview
  This chapter describes some basic concepts and the functionality of the individual components of interNet Services.
- Chapter 3: SSL
  This chapter describes aspects of communications security on the Internet and the fundamentals of cryptography, and provides an overview of SSL. The chapter also explains what must be borne in mind when applying for and creating certificates.
- Chapter 4: FTP
  This chapter describes the client and server functionality offered by interNet Services in BS2000 from a user-oriented standpoint. It also includes detailed descriptions of the client commands with illustrative examples.
- Chapter 5: FTAC interface
  This chapter describes the use of the FTAC interface for FTP and provides details on the FTAC command interface.
- Chapter 6: TELNET
  This chapter describes the client and server functionality of TELNET from a user-oriented standpoint. It also includes detailed descriptions of the client commands with illustrative examples.
- Chapter 7: OpenSSH
  This chapter describes the structure and client applications of OpenSSH. The basic utilities of OpenSSH are also explained.
- Chapter 8: Mail Reader
  This chapter describes the functionality of the Mail Reader.
- Chapter 9: Mail Sender
  This chapter describes the control of the Mail Sender by means of SDF commands and the ASSEMBLER macro interface or the C interface (include files). It also contains a detailed description of the configuration of the Mail Sender using the Mail Sender frontend (user option file).
- Chapter 10: Frequently asked questions (FAQ)
  This chapter describes the questions most frequently asked by interNet Services users and gives practical tips on problem solving.

1.3 Licensing regulations

The licensing regulations for the OpenSSL package and the TLS-FTP patch of Peter 'Luna' Runestig are printed below.

```
LICENSE ISSUES
==============

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of
the OpenSSL License and the original SSLeay license apply to the toolkit.
See below for the actual license texts. Actually both licenses are BSDstyle
Open Source licenses. In case of any license issues related to OpenSSL
please contact openssl-core@openssl.org.

OpenSSL License
---------------

/* ====================================================================
 * Copyright (c) 1998-2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
```

```
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

 Original SSLeay License
 -----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given
attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
```

```
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the
library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an
acknowledgement:
 *    "This product includes software written by Tim Hudson
(tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply
be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/*
 * Copyright (c) 1999 - 2002 Peter 'Luna' Runestig <peter@runestig.com>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modifi-
 * cation, are permitted provided that the following conditions are met:
 *
 *   o  Redistributions of source code must retain the above copyright
notice,
 *      this list of conditions and the following disclaimer.
```

```
 *
 *   o  Redistributions in binary form must reproduce the above copyright no-
 *      tice, this list of conditions and the following disclaimer in the do-
 *      cumentation and/or other materials provided with the distribution.
 *
 *   o  The names of the contributors may not be used to endorse or promote
 *      products derived from this software without specific prior written
 *      permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED
 * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE
LI-
 * ABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUEN-
 * TIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEV-
 * ER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABI-
 * LITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF
 * THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.

 */
```

1.4 Changes since the previous version of the manual

The following list of changes provides an overview of what is new in interNet Services V3.4B and relevant to this manual.

The changes that affect the interNet Services Administrator Guide are listed in the corresponding section of that manual.

OpenSSH

- The SSH protocol version 1 is no longer supported.
- Elliptic Curve Cryptography (ECDSA, Ed25519) is now supported.

SSL/TLS protocol versions and cipher suites for FTP, TELNET, Mail sender, and Mail reader

- Discontinuation of SSLv2
- Extension to include TLSv1.1 and TLSv1.2.
- New cipher suites for the SSL/TLS cipher suites

FTP commands

- New FTP server command *site svfs* for enabling/disabling TVFS.
- New FTP client commands *mlsd* and *mlst* for issuing the new FTP server commands MLSD and MLST.
- Support for alternative EOF marking for PAM files using Last Byte Pointer (LBP) mode: New option *lbp* for *setfile* (client) or *sfil* (server) command.

Discontinuation of crypto hardware

Crypto hardware is no longer supported. As a result, the *tlsUseCryptoHardware* and *-Z UseCryptoHardware* options are no longer available.

Modified manual structure

- The "Frequently asked questions (FAQ)" section is now a separate chapter (chapter 10).
- The structure of the cipher mnemonics for creating preference list specifications is now described centrally in the section “Specification of a cipher suite preference list” on page 60.

README files

The README files listed below were included in the manual:

– MAIL (BS2000/OSD) V3.2A
– TCP-IP-AP V5.1A

FTAC interface

Chapter 5, with the description of the FTAC interface, has been revised and refers to openFT V11.0.

The “Messages” chapter with the FTAC messages has been omitted.

1.5 Notational conventions

This manual uses the following notational conventions:

italics
: denote file names, program names, names of management windows, parameter names, menu titles and menu options as well as commands and variables in the main body of text.

<angle brackets>
: identify variables for which you have to enter values.

[square brackets]
: indicate optional input.

{braces} ...
: indicate a list of alternatives which are separated by “|”.

`fixed-pitch font`
: denotes input for the system, system output and file names in examples.

command
: Elements (names of commands and parameters) of the syntax description for commands that must be entered unchanged are highlighted in bold.

► indicates activities which the user must perform.

nnn is a placeholder for the respective version of a component, e.g. 053.

For informative text

CAUTION!

For warnings

References

References in the manual include the page concerned and the section or chapter as required. References to topics described in other manuals include the short title of the manual. The full title can be found in the list of related publications.

1.6 README files

Information on any functional changes and additions to the current product version described in this manual can be found in the product-specific README files.
You will find the README files on your BS2000 system under the file names listed below:

```
SYSRME.TCP-IP-SV.nnn.E
SYSRME.TCP-IP-AP.nnn.E
SYSRME.MAIL.nnn.E
```

nnn stands for the version, e.g. 034 stands for the version 3.4.

The user ID under which the README files are cataloged can be obtained from your system administrator. You can view the README files using the /SHOW-FILE command or an editor and print it out on a standard printer using the following command:

```
/PRINT-DOCUMENT <filename>,LINE-SPACING=*BY-EBCDIC-CONTROL
```

2 Internet services in BS2000 (overview)

With openNet Server, BS2000 OSD/BC provides a set of internet services that use the transport protocols TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).
openNet Server is a prerequisite for using interNet Services. The relevant internet services for internet servers were defined by the Open Group in the X98PS Product Standard. These services are integrated for BS2000 OSD/BC in the interNet Services delivery unit.

The table below gives an overview of the services provided by the individual products.

Product	**Service group**	**Service**	
openNet Server	TCP/IP Communications Services	TCP UDP IPv4 IPv6 -	Transmission Control Protocol User Datagram Protocol Internet Protocol Version 4 Internet Protocol Version 6 Requirements for internet hosts
interNet Services	File Transfer Service	FTP	File Transfer Protocol
	Terminal Service	TELNET	
	Name Service	DNS DDNS	Domain Name Service Dynamic DNS
	Time Service	NTP	Network Time Protocol
	Security Service OpenSSL	SSL TLS	Secure Socket Layer Transport Layer Security
	Security Service OpenSSH	SSH	Secure Shell
	Mail Services	SMTP POP IMAP	Simple Mail Transfer Protocol Post Office Protocol Internet Message Access Protocol
APACHE	Hypertext Services	HTTP	HyperText Transfer Protocol
SNMP Basic Agent BS2000	Network Management	SNMP	Simple Network Management Protocol

The supported protocols generally work according to the client/server model, i.e. the server provides services which are requested and used by one or more clients. The server and the clients may run on the same system or on different systems.

The individual services included in interNet Services are described in detail in the sections below.

2.1 FTP

Data exchange is of central importance when combining several computers in a network. Due to the vast number of computer types available on the market, it is essential to use a vendor-independent standard. FTP (File Transfer Protocol) enables data to be exchanged independently of the structure and the operating system of the computers involved. FTP is based directly on TCP and can transfer files of all types (e.g. text, image, sound, video or program files).

The user communicates with the FTP client via the user interface, and the FTP client sets up a connection to the FTP server through port 21 (control connection). The client sends commands to the server via this connection, and the server, in turn, acknowledges these commands by sending corresponding messages back to the client. The FTP server then establishes a second connection to the FTP client using port 20 (data connection) for the actual data exchange.

BS2000 OSD/BC provides both the server and the client functionalities of FTP. Apart from the standard protocol, the following functions are offered:

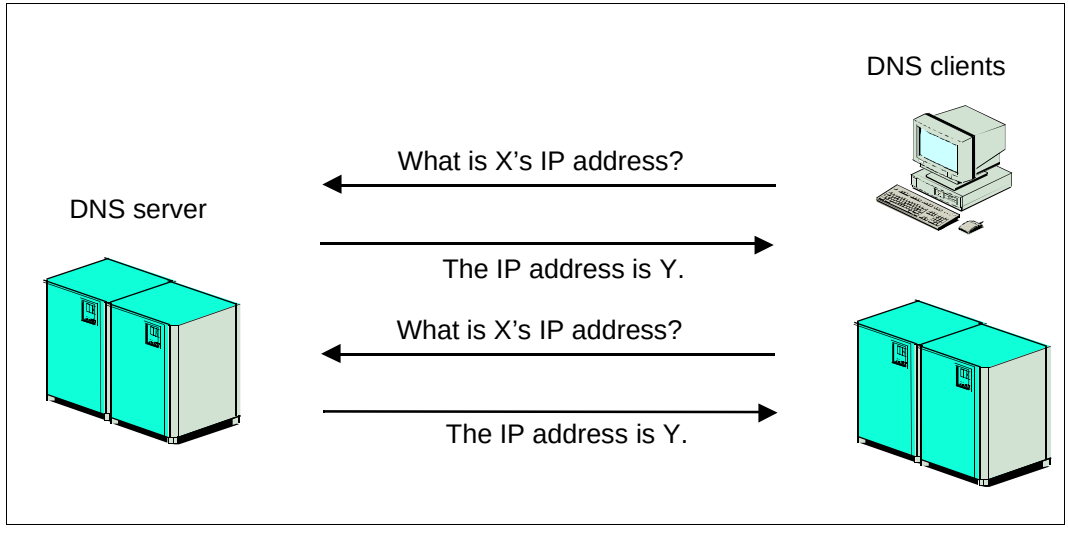
- support for BS2000 file formats (SAM, PAM)
- selection of code conversion tables for EBCDIC to ASCII, and vice versa
- security functions via the interface to the optional security product openFTAC, which permits FTP transfer admissions and login authorizations to be handled separately, user-specific access profiles to be defined, and access checks to be logged.
- Additional security functions through the use of TLS/SSL encryption for the control and/or data connection in order to ensure the confidentiality, authenticity and integrity of the data transferred between the server and client.

2.2 TELNET

TELNET enables terminal sessions to be run on networked computers. It offers essentially the same functionality as the rlogin command in Unix systems, but also includes some protocol elements specially tailored to mainframe systems. BS2000 OSD/BC supports both the client and the server functionality of TELNET.

2.3 DNS

DNS (Domain Name Service) is a global network of DNS servers that map names to IP addresses. Neither the internet nor the operation of intranets would be possible without DNS.



DNS names have a hierarchical tree structure spanning various domain levels. The root domain is the starting point for all searches within the entire DNS name space. Besides the names, the DNS also includes addresses and other information.

The Domain Name Service is a distributed and replicated database system with DNS servers and DNS clients (resolvers). The data is maintained on several DNS servers, each of which may be responsible for one or more DNS domains. If desired, redundant DNS servers may be used for failover security. The resolvers (or clients), by contrast, do not have a local database. For each DNS query, a client may contact one or more DNS servers to obtain the information it needs. These DNS queries can optionally be signed.

When queries are made, cryptographic signatures only secure communication between servers. Cryptographic signatures are not supported for communication between client and server.

BS2000OSD/BC provides its users with both the server and the resolver functionality of DNS.

Both the server and the resolver functionality were ported from the BIND coding, which is considered the standard implementation for DNS. This provides BS2000 users with access to all DNS functions and services. In addition, the high availability of BS2000 also guarantees reliable access to the DNS servers on the network.

2.4 NTP

The Network Time Protocol (NTP) allows a reference time (Universal Coordinated Time, UTC) to be distributed in a network and ensures that clocks are synchronized in networks of any size.

Time servers are in a hierarchical relationship to each other, with secondary time servers receiving their time over the network from a primary time server. For feeding the UTC time into the network it is advisable to equip a server with a radio-controlled hardware clock that receives a time signal generated by an atomic clock. A BS2000 server can carry out the function of a precise time server or client.

2.5 OpenSSH

OpenSSH is the free version of SSH (**S**ecure **Sh**ell).

SSH is a cryptographic protocol for performing the following tasks:

- Login on a remote computer
- Interactive/non-interactive command execution on a remote computer
- File transfer between different computers in a network
- Tunneling of unsecured TCP protocols over an encrypted SSH connection

SSH refers not just to the protocol itself but also to its concrete implementations.

OpenSSH is the secure alternative to the r utilities *rlogin*, *rcp* and *rhs* and the programs *telnet* and *ftp*, provided these programs are not secured with TLS/SSL. In contrast to the aforementioned programs, OpenSSH encrypts all network traffic (including passwords) and thus prevents eavesdropping, connection hijacking, and other attacks at network level. Furthermore, OpenSSH supports a raft of tunneling variants and a wide range of authentication methods.

2.6 Mail servers

Sending and receiving electronic mail (e-mail) are among the most important services on the Internet. The role of post offices is taken over by mail servers, which are also called Mail Transfer Agents (MTA). Mail servers handle transfer of e-mails over the network and ensure they are delivered to mailboxes.

The electronic mail service on the Internet is based on the Simple Mail Transfer Protocol (SMTP), which is defined in RFC 821 and RFC 2821. Mail servers that handle the electronic mail service on the basis of the SMTP protocol are also known as SMTP servers. Originally only pure text messages could be transferred, but today the MIME mechanism (Multipurpose Internet Mail Extensions, RFC 2045 through 2049) enables a wide range of formats (e.g. images) to be transmitted.

The SMTP server receives messages either from another SMTP server or from a Mail User Agent (MUA), the mail sender. The SMTP server can function as a mail relay or mail end system. The SMTP server uses the Domain Name Service (DNS) in order to select a suitable route to an end system.

The SMTP server used by the interNet Services in BS2000 is the product Postfix Version 3.1.2 (at the time the manual went to press), which has been ported to BS2000 and which is supporting IPv6 and TLS functionality. This Open Source SMTP Server created by Wietse Venema is characterized in particular by high performance, simple manageabilty and a high degree of security.

2.7 Mail sender and mail reader

The mail sender and mail reader are mail user agents (MUA), with which you can send and receive e-mails in BS2000:

- With the mail sender in interNet Services you can send e-mails to the local mail server in POSIX or to remote mail servers. The mail sender provides you with the usual functions for sending e-mails, allowing you, for example, to specify “To”, “Cc” and Bcc” recipients or attach files. In addition, you can also send automated lists, for example, as e-mails from within BS2000 procedures.

  You can sign and/or encrypt e-mails in accordance with S/MIME and send them to the mail server over a TLS/SSL-secured SMTP connection.

- The mail reader in interNet Services permits the automated processing of received e-mails. The mail reader allows you in BS2000 to use the access services POP3 and IMAP to access mailboxes located in POSIX or on a remote server and to retrieve and further process e-mails contained in them.

  The mail reader in BS2000 offers you two options for further processing e-mails:

  – Procedures allow you to access an e-mail’s header, message body and attachments.

  – At the C++ programming interface, the entire e-mail is transferred to the application as an instance of a C++ class for further processing.

    The open-source product mimelib is used to represent the e-mail at the software level. This makes classes available that provide easy access to the e-mail.

3 SSL

SSL (Secure Sockets Layer) is currently the most sophisticated security protocol in the Internet. Originally developed by the company Netscape Communications to permit secure data transfer using the HTTP protocol, SSL can in the meantime secure every protocol that is located above the Transport Layer (TCP) in the TCP/IP protocol stack.

This chapter deals with the following topics:

- Communications security on the Internet
- Fundamentals of cryptography
- Overview of SSL
- Applying for and generating X.509 certificates
- Overview of TLS/SSL support in FTP and TELNET

3.1 Communications security on the Internet

The explosive growth of the Internet and its advance into practically all areas of daily life mean that related security aspects are increasingly gaining in importance:

- Communications security, in other words data authenticity, data integrity, data confidentiality, etc.
- Reliability, in other words general availability of the systems involved
- Protection against viruses, worms, Trojan Horses, backdoors, etc.

The focus of SSL and thus of this chapter consists of aspects relating to communications security.

3.1.1 Dangers for communications security

The threats to communications security on the Internet fall into the categories of active and passive attacks.

Active attacks on communications security

Active attacks on communications security include:

- Forging the address of the message sender (address spoofing)
- Forging the contents of the messages (tampering)
- Intercepting messages and reading them in again (capture replay)
- Changing the sequence of the messages sent

Passive attacks on communications security

Passive attacks on communications security include:

- Reading message contents
- Analyzing the traffic flow of the messages:
  - Who are the communications partners?
  - Message volume in terms of periods
  - Length and frequency of messages

3.1.2 Communications security through cryptography

SSL counters the threats to communications security with the aid of cryptographic methods (see the section “Fundamentals of cryptography” on page 32).

The aims of the cryptographic methods are as follows:

- Authentication of the data origin
  Authentication of the data origin indicates that the specified data origin is the actual sender. This is required to ward off active attacks in which the attacker places himself between the two communications partners (“man in the middle”) and pretends to each partner to be the other communications partner.
- Data confidentiality
  Data confidentiality prevents data being read by unauthorized persons.
- Data integrity
  Data integrity guarantees that transferred data has not been modified.
- Anti-replay
  Anti-replay prevents data being intercepted and read in again by an intruder.
- Confidentiality of the traffic flow
  Confidentiality of the traffic flow prevents unauthorized analysis of the message traffic.
- Non-repudiation
  Non-repudiation ensures that the communications partners cannot deny that they sent the transferred data.

SSL enables the first four of these aims to be implemented. In the process, SSL offers a high degree of flexibility in selecting the cryptographic methods used, at the same time relieving the user of the need to have detailed cryptographic knowledge.

3.2 Fundamentals of cryptography

Cryptography implements the aims of communications security such as data confidentiality, data integrity and so on with the aid of the following cryptographic methods:

- Encryption methods
- Cryptographic hash functions and Message Authentication Codes (MACs)
- Digital signatures

Most cryptographic methods require strong random numbers. To prevent every individual cryptographic application having to implement methods for generating suitable random numbers, the random number generator PRNGD (**P**seudo **R**andom **N**umber **G**enerator **D**emon) is available centrally in BS2000 (see the “interNet Services Administrator Guide”).

3.2.1 Encryption methods

There are two classes of encryption methods which, because of their specific advantages and disadvantages, are tailored to different application areas:

- Symmetric key encryption methods

  Symmetric key encryption methods are used for encrypting the payload (confidentiality).

- Asymmetric (public) key encryption methods

  Asymmetric key encryption methods are used

  – in key exchange protocols,

  – to create digital signatures (non-repudiation).

Common to both classes is that security is based on the key(s) remaining confidential, while the method itself is known generally.

Symmetric key encryption

With symmetric key encryption, the cryptographic algorithms for encrypting the data at the sender end and decrypting it at the receiver end use the same key.

If, before it is used, the key is to be exchanged over the same medium as that over which the encrypted payload is transported, you must counter the danger of compromising the key. In this case it is practical to use asymmetric encryption methods such as RSA or DH. However, in contrast to RSA, the DH method cannot guarantee the authenticity of the partners involved in the key exchange. This must be implemented using an additional authentication mechanism, for example via DSS (Digital Signature Standard).

As each pair of communications partners requires a separate key, key management involves a considerable effort because the number of keys needed is proportional to the square of the number of group members.

The speed of the symmetric methods is high in comparison to that of the asymmetric methods.

The security of symmetric key encryption is dependent on the key length. To ensure secure encryption, the key should be at least 80 bits long.

The best-known symmetric key encryption methods are:

- DES (Digital Encryption Standard)
  DES should no longer be used due to its short key length of 56 bits.
- 3-DES (“Triple DES”)
  3-DES comprises consecutive three-fold DES encryption.
- AES (Advanced Encryption Standard)
  AES is now the standard symmetric key encryption method.

Asymmetric key encryption (public key encryption)

With asymmetric key encryption each communications partner has two different keys between which a mathematical relationship exists:

- Public key

  The public key is known to all communications partners and is used to encrypt messages.

- Private key

  The private key is known only to the owner and is used to decrypt messages.

When asymmetric key encryption is used, message exchange between two communications partners A and B proceeds as follows:

1. Before A sends a message to B, A must know B's public key.
2. A encrypts his/her message using B's public key.
3. A sends the encrypted message to B. (The encrypted message can now be decrypted only with the aid of B's private key.)
4. B decrypts the message with the aid of his/her private key.

As one of the two keys can be known publicly, only one key pair is required per receiver. Consequently the total number of keys required is considerably lower than with symmetric methods.

Asymmetric methods are considerably slower than symmetric methods.

With asymmetric key encryption methods only the owner of the private key can perform operations with this key. Signature methods can be created on this basis ("electronic signature").

The security of asymmetric key encryption is dependent on the key length. To ensure secure encryption, the key should be at least 2048 bits long for RSA and DH.

The best-known asymmetric key encryption methods are:

- RSA
  RSA stands for the inventors Rivest, Shamir and Adleman.
- DH
  DH stands for the inventors Whitfield Diffie and Martin Hellman. DH cannot be used for digital signatures. DSS (Digital Signature Standard), for example, is available for this purpose. DSS is also known under the name of DSA (Digital Signature Algorithm).
- ECC (Elliptic Curve Cryptography)
  ECC-based methods offer a comparable level of security to RSA/DH with much smaller key lengths. These are therefore increasingly being used, particularly if CPU resource consumption is a key factor.

3.2.2 Cryptographic hash functions and MACs

A hash function is a mathematical function which maps a character string of any given length onto a character string of fixed length. In this way hash functions can be used to create a characteristic identifier for an extensive plain text. This identifier is referred to as a checksum, message digest or simply digest.

A hash algorithm suitable for cryptographic purposes must satisfy a number of requirements:

- For identical input, a hash algorithm must return the same output.
- Minimal changes to the input must result in a significantly changed message digest.
- Under no circumstances may it be possible to reconstruct the input from the message digest.
- It should be virtually impossible to find two different plain texts for which the hash algorithm returns the same message digest.

Hash functions with these characteristics are known as cryptographic hash functions. Cryptographic hash functions are very suitable for securing data integrity.

Frequently used hash algorithms are MD5, SHA-1, and SHA-2 (generic term for SHA-224, SHA-256, SHA-384, and SHA-512). The digest length is 128 bits for MD5, 160 bits for SHA-1, and 224/256/384/512 bits for SHA-2. For MD5 and now also for SHA-1, security-related deficiencies have been identified, which means that you should use methods from the SHA-2 class if possible.

Message Authentication Code (MAC)

Message Authentication Codes (MACs) are cryptographic hash functions which in addition use a secret key to generate the message digest. MACs secure integrity and authenticity of the data traffic between two communications partners who share one secret key.

The most commonly used MAC is HMAC. HMAC can be used with every cryptographic hash algorithm and is currently the only MAC supported in SSL and OpenSSL.

3.2.3 Digital signatures

In addition to ensuring data integrity, cryptographic hash algorithms are used to create digital signatures. For this purpose a hash value (message digest) is first calculated from a plain text, and this is then encrypted with a private key. Digital signatures are suited particularly for securing non-repudiation.

For security reasons, the minimum length of the public key used should be 2048 bits.

3.3 Overview of SSL

SSL (Secure Socket Layer) permits mutual authentication of two communicating applications and, in addition, guarantees confidentiality, integrity, and authenticity of the application data exchanged. Client/server systems can thus communicate via SSL without running the risk of exchanged data being intercepted or forged. The use of SSL is transparent for the protocols and applications involved.

SSL implements authentication, data integrity, and data confidentiality with the aid of two subordinate protocols:

- SSL Record Protocol
- SSL Handshake Protocol

The SSL Record Protocol defines the format used for transferring data. The SSL Handshake Protocol enables the SSL client and SSL server to authenticate themselves to each other and to exchange encryption algorithms together with the cryptographic key before an Application Layer protocol transfers the first data.

In interNet Services the implementations are based on the OpenSSL-Toolkit. At the time this manual went to print, Version 1.0.2k of the OpenSSL-Toolkit was supported. The protocol versions supported are SSLv3, TLSv1, TLSv1.1 and TLSv1.2.

3.3.1 SSL in the TCP/IP protocol stack

The SSL protocol is included in the TCP/IP protocol stack above the TCP protocol and below the Application Layer:

- The SSL Record Protocol is based directly on the TCP protocol.
- The SSL Handshake Protocol operates on the basis of the SSL Record Protocol.

3.3.2 SSL and TLS

The Transport Layer Security Protocol V1.0 is an enhancement of the SSL V3.0 protocol which was standardized by the Internet Engineering Task Force (IETF) in RFC 2246. Although no major differences exist between TLS V1.0 and SSL V3.0, interoperability between the two protocols is not possible without further provision. TLSv1.1 and TLSv1.2 are enhancements of the TLS protocol, which rectify the deficiencies found in TLSv1 (see note).

Although SSL was initially developed for the HTTP protocol, SSL and TLS can also be used for securing other protocols of the Application Layer, such as FTP, SMTP or TELNET.

CAUTION
Version 3 of the SSL protocol displays some security-related deficiencies and should therefore not be used if possible. Some security-related deficiencies are only fundamentally resolved with TLS version 1.2. You should therefore use TLSv1.2 if possible.

3.3.3 Cipher suites

Not all conceivable combinations of the various cryptographic methods can be used with TLS/SSL. On the contrary, in the TLS/SSL standards a number of permissible combinations of authentication methods (RSA, DSS), key exchange methods (RSA, DH), symmetric key encryption methods (DES, 3DES, AES etc.) and message digest have been defined. These combinations are referred to as cipher suites.

3.3.4 X.509 certificates, Certificate Authorities (CA) and CRLs

An X.509 certificate contains all the information required to identify the server or client, plus the public key of the certificate owner. Certificates are stored in separate files. When a connection is negotiated, SSL uses the certificate files to identify the server and, in some applications, also the client.

Certificate Authority

Certificates are issued by a central authority, the Certificate Authority (CA), by signing these with the CA's private key once the identity of the organization named in the certificate and of an authorized representative has been checked. The signature is contained in the certificate and is disclosed at the time of connection setup so that the client can verify the trustworthiness of the certificate. The server can also request a certificate from the client. However, in practice this rarely happens.

Certificates which are signed by a CA can be declared invalid in a Certificate Revocation List (CRL).

X.509 certificates

X.509 certificates are used in conjunction with SSL. X.509 certificates work with a hierarchical trust structure, at the top of which the Certificate Authorities are legally liable for ensuring the proven identity of the certificate owners. Depending on the trust level, the CAs may be satisfied with a valid e-mail address or a valid host name as proof of identity, or request more detailed information (see the section “Applying for and creating X.509 certificates” on page 41). Typical central CAs are VeriSign or Thawte.

You can view the content of a certificate using a browser or have it output using the SHOW.CERT procedure (see page 51). (The SHOW.CERT procedure calls the OpenSSL command program.)

The figure below shows a sample X.509 certificate.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=XY, ST=Snake Desert, L=Snake Town, O=Snake Oil,
OU=Certificate Authority, CN=Snake Oil CA/emailAddress=ca@snakeoil.dom
        Validity
            Not Before: Dec 14 16:22:17 2015 GMT
            Not After : Dec 14 16:22:17 2035 GMT
        Subject: C=XY, ST=Snake Desert, L=Snake Town, O=Snake Oil,
OU=Certificate Authority, CN=Snake Oil CA/emailAddress=ca@snakeoil.dom
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:9e:f9:2f:58:46:09:09:d0:14:8d:79:8d:03:22:
                    ed:ae:4f:a6:94:6c:97:a8:af:49:7d:1c:26:e3:27:
                    27:9d:aa:d4:04:90:40:99:b7:24:c0:27:63:33:a9:
                    de:58:a0:41:99:8e:56:e9:79:9a:ea:f3:c9:98:5c:
                    76:4b:8b:78:f6:49:4c:e6:8d:25:25:cd:e5:04:84:
                    2b:fd:fb:d1:51:e1:f9:e9:1a:da:5d:74:93:3e:24:
                    13:e2:33:9d:52:10:05:bd:f0:b6:38:81:1f:6b:3a:
                    a5:d9:ae:80:b2:30:0a:bd:70:1c:ff:4c:25:0c:3a:
                    b9:43:82:2e:d0:28:7b:6f:4d:4a:8e:ac:48:c2:c5:
                    e6:a2:70:a5:04:04:94:6b:44:f7:bc:27:20:99:77:
                    94:2f:c1:98:4d:51:e2:16:fb:8f:c9:15:e2:4b:31:
                    9d:d0:ee:16:89:bb:8e:2d:ea:90:f6:56:c7:ae:fd:
                    07:13:a1:2a:3c:4e:a1:a8:f4:f7:91:f6:3e:6a:fe:
                    ae:22:65:a7:be:9f:3b:57:1c:3b:90:77:85:6b:6f:
```

```
                    25:94:58:22:12:89:b6:bd:e6:ce:89:92:41:bf:7f:
                    02:89:53:1c:87:81:44:33:f7:ae:85:9d:3c:df:fb:
                    99:43:81:e3:dc:76:84:f4:b1:0a:d2:6f:98:91:4c:
                    b4:e1
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                email:ca@snakeoil.dom
            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         8a:e4:e9:27:ce:20:c7:76:9e:c1:76:c2:b0:83:74:4f:72:fd:
         a8:ea:f7:f7:94:01:f3:86:ee:e4:99:ba:46:03:c2:62:bf:73:
         7c:a3:7d:66:f7:d2:29:12:e3:f9:c4:88:04:47:bd:0b:e2:73:
         0d:cd:ed:fb:48:61:37:4f:7b:85:16:45:ec:f5:49:cf:d1:17:
         00:71:38:87:0e:10:24:b3:4f:ff:50:22:d9:67:25:17:5d:26:
         3f:0a:c4:d1:9a:fe:e3:d7:4b:09:c0:93:de:31:32:09:14:57:
         8d:9c:59:41:ab:05:08:6c:09:5b:c8:0f:5a:92:20:31:66:9c:
         5f:b0:ab:ba:29:19:04:04:94:b8:55:b5:a0:f3:2f:09:b2:5d:
         5e:47:da:da:c8:2d:38:57:48:27:a9:34:a9:dc:94:ed:c8:30:
         74:e1:ad:86:6c:60:d0:a7:fa:ed:ba:e7:67:bd:ae:e7:5e:f5:
         c2:50:d2:f1:93:00:53:51:ad:84:9e:7e:9d:c2:5a:b1:0a:2d:
         13:a9:92:97:8c:69:3a:48:9b:76:1e:df:29:16:6a:b9:75:77:
         91:57:4e:53:da:92:73:65:0c:95:65:01:eb:c6:1a:64:06:e0:
         40:c5:63:6d:03:85:36:e3:d7:17:a2:c7:4b:cb:7b:23:72:a8:
         4b:2a:bc:96
```

Sample X.509 certificate

Certificate Revocation List (CRL)

Systems which process certificates must be able to recognize when a certificate has been revoked. With X.509 certificate this status check is implemented with the aid of Certificate Revocation Lists (CRL). A CRL contains all the certificates issued by a particular Certificate Authority (CA) which are no longer valid. This enables certificates that were issued by a Certificate Authority to be declared invalid by publishing a CRL.

3.3.5 Applying for and creating X.509 certificates

Generally you obtain an X.509 certificate from a commercial Certificate Authority (CA) such as VeriSign (http://www.verisign.com), Thawte, to name but a few. The certificates issued by the CAs are normally classified by trust levels (for example “Class 3”).

A distinguishing feature of the individual trust levels is the effort involved in identifying the applicant:

- In the case of low trust levels it is sufficient to be able to deliver e-mails to the specified e-mail address.
- In the case of higher trust levels the applicant must, for example, supply a verified extract from the commercial register for the company involved. In addition, an authorized signatory or PKI executive of the company must identify himself/herself using the Post Ident procedure or something similar.

  Higher trust levels generally also mean higher warranty sums in the event of loss, for example if the CA issues a certificate to an unauthorized entity. Further details can be found on the CAs’ websites.

A new certificate must be obtained and installed in good time before the old one becomes invalid. If the private key has been compromised or the information in the certificate is no longer applicable, the certificate must be revoked.

If the certificates are only intended for inhouse applications, it may make sense to set up your own CA. However, before taking such a step you should gain a thorough knowledge of the topic PKI (Public Key Infrastructure), for example by reading the relevant literature.

In addition to the identification documents the applicant must also submit a Certificate Signing Request (CSR). You can create a CSR with the OpenSSL command line tool, for example (see the section “MAKE.CERT procedure - Generating test certificates and CSRs” on page 44).

3.3.6 SSL handshake

SSL communication between the client and server always begins with a so-called handshake. This handshake permits authentication of the server and agreement to be reached on the encryption method and key that are to be used.

With every handshake the server must authenticate itself to the client by means of public key encryption. The server can also request the client to authenticate itself (also by means of public key encryption). However, this is generally not done.

Expressed in simplified form, the following steps are required for the SSL handshake:

1. The client sends a list of the encryption methods (cipher suites) it supports to the server.
2. The server selects a method from this list and returns this to the client, together with the server's own certificate containing the server's public key.
3. The client checks whether the server certificate has already been signed by a CA whose certificate is present at the client end and which the client thus trusts implicitly. The client also checks whether the certificate was issued for the server to which it wants to set up the connection. Here the client checks
   - whether the CN (Common Name) of the subject is the same as the server's FQDN (Fully Qualified Domain Name), or
   - whether the DNS part of the X.509v3 Subject Alternative Name is the same as the server's FQDN.
4. With the aid of the server's public key the client encrypts a PreMaster Secret and sends this to the server. From the PreMaster Secret and the random numbers exchanged in the preceding steps the client and server then calculate the keys which are required for the various encryption and MAC algorithms.
5. Depending on the configuration, the server requests a certificate from the client in the context of the handshake. The server checks whether this certificate was signed by a CA it rates as trustworthy. No more detailed checks of the client certificate are performed within interNet Services.

3.4 Procedures for calling the OpenSSL-Toolkit

The OpenSSL-Toolkit is available for performing certain tasks when X.509 certificates are created, for example for

- generating X.509 Certificate Signing Requests (CSR) and test certificates
- displaying certificates in readable form
- selecting a suitable cipher suite list specification

The OpenSSL-Toolkit was used for implementing the SSL functions.

As the OpenSSL-Toolkit offers far greater functionality than is required for FTP and TELNET, operation is simplified by using some SDF-P procedures which are contained in the LMS library SYSSPR.TCP-IP-AP.*nnn*.

These procedures are available in the following form:

- As compiled procedures. This enables the procedures also to be used by users who only have SDF-P-BASYS, but not the chargeable subsystem SDF-P (SYSJ elements).
- As J elements. The source text of the procedures is thus also available.

A description of the following procedures is provided below:

- MAKE.CERT - Generating test certificates and CSRs (RSA or DSA, 2048/1024 bits)
- SHOW.CERT - Showing X.509 certificates in plain text
- SHOW.CIPHERLIST - Selecting a suitable cipher suite list specification
- GEN.DHPARAM - Generating DH parameter set

3.4.1 MAKE.CERT procedure - Generating test certificates and CSRs

You can use the MAKE.CERT procedure to generate test certificates and CSRs (RSA or DSA, 2048/1024 bits). MAKE.CERT has a number of call parameters.

3.4.1.1 Parameter description

Parameters for specifying the Snakeoil Certificate Authority (CA)

The parameters CA-SERIALFILE, CA-CERTFILE and CA-KEYFILE specify the files for the Snakeoil CA. With the aid of this CA you can use generated CSRs to create test certificates that allow you to test the TLS functionality before you purchase a “proper” certificate. The test certificates created by this Snakeoil CA may not be used for productive operation as they are not trustworthy. (The private key of this CA is not secret, so anyone who has access to this key can issue any certificate they wish, signed by the Snakeoil CA.)

CA-SERIALFILE

This parameter specifies the file in which the serial number of the generated test certificate is stored.

If this file does not exist when the procedure is called, it is created and initialized with a line containing “00”. As this number is entered in the relevant test certificate and some applications use this serial number to distinguish the certificates issued by a particular CA, this file should not be deleted after it has been generated once. If it is deleted, test certificates with the same serial number could be generated. This can lead to problems with the aforementioned applications.

The parameter defaults to SYSDAT.TCP-IP-AP.*nnn*.SNAKEOIL.SRL.

CA-CERTFILE

This parameter specifies the file in which the root certificate of the Snakoil CA is stored.

The parameter defaults to SYSDAT.TCP-IP-AP.*nnn*.SNAKEOIL.CERT.

CA-KEYFILE

This parameter specifies the file in which the private key of the Snakoil CA is stored.

The parameter defaults to SYSDAT.TCP-IP-AP.*nnn*.SNAKEOIL.KEY.

Parameter for specifying the generation data for the DSA key

DSA-PARAMFILE

This parameter specifies the file in which the parameters required for generating a DSA key are stored.

The parameter defaults to SYSDAT.TCP-IP-AP.*nnn*.DSAPARAM.

Parameters for specifying the test certificate, private key and CSR

The parameters CERTFILE, KEYFILE and CSRFILE specify the files in which the test certificate, the associated private key and the Certificate Signing Request (CSR) are stored.

CERTFILE

This parameter specifies the file in which the generated test certificate is stored. The name of this file can be specified in the RSA-CERTIFICATE-FILE operand of the FTP installation command (see the “interNet Services Administrator Guide”).

The parameter defaults to SYSDAT.TCP-IP-AP.*nnn*.NEW.CERT.

KEYFILE

This parameter specifies the file in which the private key belonging to the test certificate and CSR is stored. The name of this file can be specified in the RSA-KEY-FILE operand of the FTP installation command. The content of this file must be kept secret, especially if you later intend to apply for a certificate for productive operation with the associated CSR (see the “interNet Services Administrator Guide”).

The parameter defaults to SYSDAT.TCP-IP-AP.*nnn*.NEW.KEY.

CSRFILE

This parameter specifies the file in which the CSR is stored. When you wish to obtain a certificate for productive operation, send this file to a commercial CA. After you have received the certificate from the CA, you may reinstall FTP, but then you must specify in the RSA-CERTIFICATE-FILE operand the file name of the certificate you received from the CA.

The parameter defaults to SYSDAT.TCP-IP-AP.*nnn*.NEW.CSR.

Parameters for determining the key type and passphrase encryption

KEY-TYPE

This parameter species whether an RSA or a DSA key is to be generated (RSA / DSA).

The parameter defaults to RSA.

KEY-ENCRYPTION

This parameter species whether the generated RSA or DSA key is to be encrypted with a passphrase (YES / NO):

– Encryption of the private RSA or DSA key makes little sense for a server as the server then asks for the passphrase when it starts up and can therefore no longer be started automatically.

– If the generated key is to be used for a client certificate, this encryption of the private key can make sense.

The parameter defaults to NO.

3.4.1.2 Procedure run

After it has been called, the procedure behaves as follows:

1. The RSA or DSA key pair is generated with a key length of 2048/1024 bits.
2. The X.509 CSR is generated. For this purpose, some information is queried from the calling party in interactive mode.
3. A test certificate is generated from the CSR with the aid of the Snakeoil CA.

   For this purpose, further information is queried from the calling party:

   – Validity period of the test certificate

   – Version of the certificate (X.509v1 or X.509v3)

   – If “3” is specified (X.509v3), the DNS name is queried in subjectAltName. The DNS name is generally identical to the “Common Name” (CN) under 2).
4. The generated certificate is displayed in plain text.

Example

A log from a procedure call is printed below. The user entries are highlighted in **bold print**.

```
/CALL-PROCEDURE *LIB($.SYSSPR.TCP-IP-AP.nnn,MAKE.CERT)
SSL Certificate Generation Utility
Copyright (c) 2003 Fujitsu Technology Solutions, All Rights Reserved

Generating test certificate signed by Snake Oil CA (TEST)
WARNING: Do not use this certificate for real-life/production systems.
         However, you can use the generated Certificate Signing
         Request (CSR) for requesting a real Server Certificate
         from a commercial Certificate Authority (CA).
------------------------------------------------------------------------

STEP 1: Generating RSA private key (2048 bit)
%  BLSO523 ELEMENT 'OPENSSL', VERSION 'V05.3A00', TYPE 'L' FROM LIBRARY
':HPC3:$TSOS.SYSLNK.TCP-IP-AP.053' IN PROCESS
%  BLSO524 LLM 'OPENSSL', VERSION 'V05.3A00' OF '2016-12-05 19:30:57' LOADED
%  BLSO551 COPYRIGHT (C) 2016 FUJITSU TECHNOLOGY SOLUTIONS GMBH. ALL RIGHTS
RESERVED
Generating RSA private key, 2048 bit long modulus.....................+++
 ..........................................................+++

e is 65537 (0x10001)
------------------------------------------------------------------------

STEP 2: Generating X.509 certificate signing request
%  BLSO523 ELEMENT 'OPENSSL', VERSION 'V05.3A00', TYPE 'L' FROM LIBRARY
':HPC3:$TSOS.SYSLNK.TCP-IP-AP.053' IN PROCESS
%  BLSO524 LLM 'OPENSSL', VERSION 'V05.3A00' OF '2016-12-05 19:30:57' LOADED
%  BLSO551 COPYRIGHT (C) 2016 FUJITSU TECHNOLOGY SOLUTIONS GMBH. ALL RIGHTS
RESERVED
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----

1. Country Name             (2 letter code) [DE]:
*DE
2. State or Province Name   (full name)     [Bavaria]:
*Bavaria
3. Locality Name            (eg, city)      [Munich]:
*Munich
```

```
4. Organization Name        (eg, company)   [Manufacturer, Ltd]:
*Fujitsu Technology Solutions GmbH
5. Organizational Unit Name (eg, section)   [Marketing]:
*Internet Services
6. Common Name              (eg, FQDN)      [www.manufacturer.com]:
*ftp.ts.fujitsu.com
7. Email Address            (eg, name@FQDN) [info@manufacturer.com]:
*info@ts.fujitsu.com
-----------------------------------------------------------------------

STEP 3: Generating X.509 certificate signed by Snake Oil CA
%8. Certificate Validity     (days)         : 730
%Certificate Version (1 or 3)               : 3
%9. subjectAltName:dNSName   (eg, FQDN)     : ftp.ts.fujitsu.com
%  BLSO523 ELEMENT 'OPENSSL', VERSION 'V05.3A00', TYPE 'L' FROM LIBRARY
':HPC3:$TSOS.SYSLNK.TCP-IP-AP.053' IN PROCESS
%  BLSO524 LLM 'OPENSSL', VERSION 'V05.3A00' OF '2016-12-05 19:30:57' LOADED
%  BLSO551 COPYRIGHT (C) 2016 FUJITSU TECHNOLOGY SOLUTIONS GMBH. ALL RIGHTS
RESERVED
Signature ok
subject=/C=DE/ST=Bavaria/L=Munich/O=Fujitsu Technology Solutions
GmbH/OU=Internet
Services/CN=ftp.ts.fujitsu.com/emailAddress=info@ts.fujitsu.com
Getting CA Private Key
-----------------------------------------------------------------------

STEP 4: Show generated X.509 certificate
%  BLSO523 ELEMENT 'OPENSSL', VERSION 'V05.3A00', TYPE 'L' FROM LIBRARY
':HPC3:$TSOS.SYSLNK.TCP-IP-AP.053' IN PROCESS
%  BLSO524 LLM 'OPENSSL', VERSION 'V05.3A00' OF '2016-12-05 19:30:57' LOADED
%  BLSO551 COPYRIGHT (C) 2016 FUJITSU TECHNOLOGY SOLUTIONS GMBH. ALL RIGHTS
RESERVED
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3 (0x3)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=XY, ST=Snake Desert, L=Snake Town, O=Snake Oil,
OU=Certificate Authority, CN=Snake Oil CA/emailAddress=ca@snakeoil.dom
        Validity
            Not Before: Dec 16 16:07:38 2016 GMT
            Not After : Dec 16 16:07:38 2018 GMT
        Subject: C=DE, ST=Bavaria, L=Munich, O=Fujitsu Technology Solutions
GmbH, OU=Internet Services,
CN=ftp.ts.fujitsu.com/emailAddress=info@ts.fujitsu.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
```

```
                Modulus:
                    00:f0:7d:49:25:9a:45:29:80:15:5d:51:98:59:54:
                    c9:a4:70:47:fa:35:f9:31:55:03:35:27:de:e4:d7:
                    6a:a4:cc:6d:df:92:91:ae:14:58:55:94:31:8f:58:
                    63:44:9c:7a:1f:29:e0:eb:6f:4b:4c:9a:d6:fa:0b:
                    a1:33:05:be:3a:dd:c9:b3:1a:33:1c:39:45:6a:16:
                    1e:4f:75:bd:2a:29:eb:18:07:1b:23:0f:44:b5:01:
                    77:3b:fe:f6:a5:01:ac:df:8e:15:14:f0:1d:fd:29:
                    5b:3f:99:38:85:cc:38:c9:7c:10:59:1d:df:c8:e9:
                    53:bd:05:b6:f9:a9:8d:b6:82:68:32:b7:75:4c:1e:
                    b3:04:77:e1:7f:fb:49:5c:5c:d8:00:c9:3b:73:12:
                    2a:8c:b2:11:b2:50:dc:59:0b:03:7a:ef:0e:97:0b:
                    a2:d3:f6:53:bb:9c:77:4a:54:69:dd:ec:2e:38:fe:
                    75:b9:a1:e4:3f:aa:26:10:8d:e1:a3:d5:66:e3:82:
                    46:12:a3:29:dd:3a:73:a2:76:cc:35:1a:0a:88:57:
                    72:cc:9d:77:37:e1:16:af:5e:0d:14:50:3c:70:13:
                    89:53:19:59:37:61:ac:26:ba:2f:b2:1d:c4:6b:39:
                    1a:7f:13:e2:ab:65:3d:3e:f0:f7:6e:9e:1a:b4:91:
                    02:bf
                Exponent: 65537 (0x10001)
            X509v3 extensions:
                X509v3 Subject Alternative Name:
                    DNS:ftp.ts.fujitsu.com, email:info@ts.fujitsu.com
                Netscape Comment:
                    interNET SERVICES generated test server certificate
        Signature Algorithm: sha256WithRSAEncryption
             50:c7:0e:27:34:df:e9:c9:ba:06:c4:2b:a0:35:e5:56:bb:9c:
             dd:5b:71:41:15:64:36:9b:67:e2:08:22:a3:ea:9c:cc:e3:c5:
             5e:db:f7:96:c1:7a:a1:95:95:6d:54:92:a5:35:ab:2b:17:93:
             6e:f3:c7:35:a7:58:e8:46:49:34:ab:e6:a4:08:37:e2:4a:a1:
             81:e0:25:49:8a:8f:f5:fc:0f:7d:86:4f:18:68:5b:c0:ae:68:
             33:e6:d8:63:ef:42:f9:44:90:3f:a4:13:0c:dd:fc:38:78:a0:
             28:7c:67:72:33:18:cf:92:e1:81:1a:c2:f0:a0:57:09:79:70:
             58:0d:eb:be:d5:08:31:19:67:5a:37:b3:69:b3:cb:cf:4f:b0:
             38:c8:be:ba:6d:d3:0a:ba:78:76:b2:f4:cc:ea:10:a4:a6:db:
             a6:d4:e3:5b:12:76:11:70:68:ab:04:6b:80:7f:89:36:d5:3f:
             21:86:4b:ba:1d:56:31:32:61:6e:90:da:3f:1a:d4:40:63:b8:
             37:ae:f7:cc:fd:71:e4:02:9c:35:fc:98:f6:12:82:ff:c6:2f:
             3f:19:21:0b:c5:27:6f:af:45:bc:3e:a2:84:ae:e9:37:57:7d:
             1f:20:5d:0c:16:87:f1:04:37:4a:47:73:99:6e:77:94:6d:fe:
             46:d8:b2:c4
-----------------------------------------------------------------------

RESULT: Server certification files

o  SYSDAT.TCP-IP-AP.053.NEW.KEY
   The PEM encoded RSA private key file. KEEP THIS FILE PRIVATE.
```

- SYSDAT.TCP-IP-AP.053.NEW.CERT
  The PEM encoded X.509 certificate file.
  WARNING: Do not use this certificate for real-life/production systems.

- SYSDAT.TCP-IP-AP.053.NEW.CSR
  The PEM encoded X.509 certificate signing request file which you can send to an official Certificate Authority (CA) in order to request a real server certificate (signed by this CA instead of our demonstration only Snake Oil CA) which later can replace the SYSDAT.TCP-IP-AP.053.NEW.CERT file.

3.4.2 SHOW.CERT procedure

The SHOW.CERT procedure enables you to display the X.509 certificates, which are usually stored in PEM format, as plain text. SHOW.CERT has only the following parameter:

CERTFILE

This parameter specifies the file which contains the certificate to be shown.

The parameter defaults to SYSDAT.TCP-IP-AP.*nnn*.NEW.CERT.

Example

```
/CALL-PROCEDURE *LIB($.SYSSPR.TCP-IP-AP.nnn,SHOW.CERT),-
                (CERTFILE=SYSDAT.TCP-IP-AP.nnn.SNAKEOIL.CERT)
```

3.4.3 SHOW.CIPHERLIST procedure

The SHOW.CIPHERLIST procedure facilitates selection of the appropriate

- *tlsCipherSuite* option (see page 104) in the option files of FTP client and FTP server.
- *-Z CipherSuite* option (see page 294) in the option files of TELNET client and TELNET server.

SHOW.CIPHERLIST has no parameters.

Procedure run

After it is started, SHOW.CIPHERLIST asks for the cipher suite to be specified. Once you have entered this specification, SHOW.CIPHERLIST outputs a list of cipher suites, one beneath the other and separated by colons (:). In the handshake procedure, an FTP client started with this option would send this list (in this order) to the server as acceptable cipher suites. The order is relevant here as most servers select the first suite in this list which is included in the set of cipher suites it accepts.

After the list has been output, SHOW.CIPHERLIST asks again for a cipher suite to be specified. As soon as you have found the required option string, you can terminate the procedure by entering `quit`.

Example

```
/CALL-PROCEDURE *LIB($.SYSSPR.TCP-IP-AP.nnn,SHOW.CIPHERLIST)

SSL Cipher List Show Utility

Copyright (c) 2007 Fujitsu Technology Solutions, All Rights Reserved

Show SSL Cipher List corresponding to cipher selection string.

--------------------------------------------------------------------------------
--


%  BLS0523 ELEMENT 'OPENSSL', VERSION 'V05.3A00', TYPE 'L' FROM LIBRARY
':HPC3:$TSOS.SYSLNK.TCP-IP-AP.053' IN PROCESS

%  BLS0524 LLM 'OPENSSL', VERSION 'V05.3A00' OF '2016-12-05 19:30:57' LOADED

%  BLS0551 COPYRIGHT (C) 2016 FUJITSU TECHNOLOGY SOLUTIONS GMBH. ALL RIGHTS
RESERVED

OpenSSL>

%Cipher selection string: ALL:!EXP:!ADH:!AECDH:!LOW:!NULL:!RC4:!SRP:!PSK:-
SEED:-CAMELLIA

ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-
SHA:DH-DSS-AES256-GCM-SHA384:DHE-DSS-AES256-GCM-SHA384:DH-RSA-AES256-GCM-
SHA384:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-
SHA256:DH-RSA-AES256-SHA256:DH-DSS-AES256-SHA256:DHE-RSA-AES256-SHA:DHE-DSS-
AES256-SHA:DH-RSA-AES256-SHA:DH-DSS-AES256-SHA:ECDH-RSA-AES256-GCM-
SHA384:ECDH-ECDSA-AES256-GCM-SHA384:ECDH-RSA-AES256-SHA384:ECDH-ECDSA-AES256-
SHA384:ECDH-RSA-AES256-SHA:ECDH-ECDSA-AES256-SHA:AES256-GCM-SHA384:AES256-
SHA256:AES256-SHA:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-
SHA:ECDHE-ECDSA-AES128-SHA:DH-DSS-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-
SHA256:DH-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-
SHA256:DHE-DSS-AES128-SHA256:DH-RSA-AES128-SHA256:DH-DSS-AES128-SHA256:DHE-
RSA-AES128-SHA:DHE-DSS-AES128-SHA:DH-RSA-AES128-SHA:DH-DSS-AES128-SHA:ECDH-
RSA-AES128-GCM-SHA256:ECDH-ECDSA-AES128-GCM-SHA256:ECDH-RSA-AES128-
SHA256:ECDH-ECDSA-AES128-SHA256:ECDH-RSA-AES128-SHA:ECDH-ECDSA-AES128-
SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:ECDHE-
ECDSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:EDH-DSS-DES-CBC3-SHA:DH-RSA-DES-CBC3-
SHA:DH-DSS-DES-CBC3-SHA:ECDH-RSA-DES-CBC3-SHA:ECDH-ECDSA-DES-CBC3-SHA:DES-
CBC3-SHA

OpenSSL>

%Cipher selection string: ALL:!EXP:!ADH:!AECDH:!LOW:!NULL:!RC4:!SRP:!PSK:-
SEED:-CAMELLIA:-DH:-3DES
```

```
ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-
SHA:ECDH-RSA-AES256-GCM-SHA384:ECDH-ECDSA-AES256-GCM-SHA384:ECDH-RSA-AES256-
SHA384:ECDH-ECDSA-AES256-SHA384:ECDH-RSA-AES256-SHA:ECDH-ECDSA-AES256-
SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDH-RSA-AES128-
GCM-SHA256:ECDH-ECDSA-AES128-GCM-SHA256:ECDH-RSA-AES128-SHA256:ECDH-ECDSA-
AES128-SHA256:ECDH-RSA-AES128-SHA:ECDH-ECDSA-AES128-SHA:AES128-GCM-
SHA256:AES128-SHA256:AES128-SHA

OpenSSL>

%Cipher selection string: ALL:!EXP:!ADH:!AECDH:!LOW:!NULL:!RC4:!SRP:!PSK:-
SEED:-CAMELLIA:-DH:-3DES:-ECDH:ECDH

AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:AES128-GCM-SHA256:AES128-
SHA256:AES128-SHA:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-
SHA:ECDHE-ECDSA-AES256-SHA:ECDH-RSA-AES256-GCM-SHA384:ECDH-ECDSA-AES256-GCM-
SHA384:ECDH-RSA-AES256-SHA384:ECDH-ECDSA-AES256-SHA384:ECDH-RSA-AES256-
SHA:ECDH-ECDSA-AES256-SHA:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-
SHA:ECDHE-ECDSA-AES128-SHA:ECDH-RSA-AES128-GCM-SHA256:ECDH-ECDSA-AES128-GCM-
SHA256:ECDH-RSA-AES128-SHA256:ECDH-ECDSA-AES128-SHA256:ECDH-RSA-AES128-
SHA:ECDH-ECDSA-AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:ECDHE-ECDSA-DES-CBC3-
SHA:ECDH-RSA-DES-CBC3-SHA:ECDH-ECDSA-DES-CBC3-SHA

OpenSSL>

%Cipher selection string: quit
```

3.4.4 GEN.DHPARAM procedure

Cipher suites with Diffie-Hellman key negotiation require DH parameter sets when used on the TLS server side.

The BS2000 FTP server is supplied with a 2048 bit fixed DH parameter set. The 2048 bit length currently appears to be a sensible compromise between security and CPU resource requirements.

The GEN.DHPARAM procedure enables the FTP server operator to deviate up or down from the 2048 bit length or to generate a separate DH parameter set. For security reasons, you should not use parameters sets with less than 1024 bits. Conversely, it is important to be aware that a 4096 bit DH parameter set, for example, can slow down the TLS handshake and hence the login time significantly and increase CPU resource consumption dramatically.

3.4.4.1 Parameter description

SIZE

This parameter is a measure of the size of a DH parameter set. The greater the size, the greater the security, but also the higher the CPU resource consumption for generating and in particular for using the parameter set. The parameter is set to 2048 by default.

G

This is referred to as the generator value. Without sufficient knowledge or good reason to change the value, you should leave it set to the default value 2.

PARAMFILE

This parameter specifies the file in which the DH parameter set is to be stored. The parameter defaults to SYSDAT.TCP-IP-AP.*nnn*.DHPARAM.

3.4.4.2 Procedure run

Once you start the procedure, it runs without further user interaction until it finishes. The runtime can be anything from minutes to hours depending on the selected SIZE parameter, the available system performance, and the random numbers received from the PRNGD subsystem. The activity of the generation program is displayed by the continuous output of the '.', '+' and '*'characters.

Example

```
/CALL-PROCEDURE $.SYSSPR.TCP-IP-AP.nnn(GEN.DHPARAM),(SIZE=1024)

DH Parameter Set Generation Utility.
Copyright (c) 2016 Fujitsu Technology Solutions, All Rights Reserved

Generate DH parameter set.
------------------------------------------------------------------------

%  BLS0523 ELEMENT 'OPENSSL', VERSION 'V05.3A00', TYPE 'L' FROM LIBRARY
':HPC3:$TSOS.SYSLNK.TCP-IP-AP.053' IN PROCESS
%  BLS0524 LLM 'OPENSSL', VERSION 'V05.3A00' OF '2016-12-05 19:30:57' LOADED
%  BLS0551 COPYRIGHT (C) 2016 FUJITSU TECHNOLOGY SOLUTIONS GMBH. ALL RIGHTS
RESERVED
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long
time..............+..............................
.............................................................+...........
..
...........................................................+.............
..
.....................................................................+....
..
...+.....+.....................................................................
..
...............+.....................+.....................+............
..
.....................+.........................................+...........
.+
............+.............+.....................+.......................
..
.................................................+............................
..
.....................................................................+.........
..
.....................................................................+.........
..
..................+................+.....................................+
..
.................................................+..........................
..
..................+.................+.....................................
..
.............+.................+.........................................
..
.+.............................................................................
..
```

```
................................+.......+.....+........+.....................
..
................................................+.........................
..
..+..+..........................................................................
..
........+.........................+............................+.............
..
.............................+.....................+............................
..
.................................................................+........
..
.+...+.......................................+.......................+.....
.+
..++*++*++*
```

In the option file of the FTP server, this DH parameter file can then be specified with the *-tlsDHparamFile* option:

```
-tlsDHparamFile $.SYSDAT.TCP-IP-AP.053.DHPARAM
```

3.5 Using TLS/SSL

interNet Services offers the option of TLS/SSL support for FTP, TELNET, Mail reader, Mail sender and Mail server.

3.5.1 TLS/SSL support in FTP

TLS/SSL support on the FTP client and FTP server can be controlled in more than one way.

Selecting TLS/SSL support on the FTP client

The following instruments are available to you for selecting TLS/SSL support on the FTP client:

- Option file

  TLS/SSL support offers a wide range of selection options. These settings can be stored in one or more option files via options.

- FTP client commands for TLS/SSL support

  These are commands for enabling/disabling TLS/SSL support, for reading the option file, etc.

- Extended status information

  The *status* command output contains TLS-specific information.

TLS/SSL support on the FTP client is described in detail in the section “TLS/SSL support in the FTP client” on page 95.

Selecting TLS/SSL support on the FTP server

The following instruments are available to you for selecting TLS/SSL support on the FTP server:

- Option file

  TLS/SSL support offers a wide range of selection options. These settings can be stored in one or more option files via options.

- Installation command SET-FTP-TELNET-PARAMETERS

  The settings for TLS/SSL security can also be defined with the aid of additional parameters of the installation commando.

- FTP protocol commands

  The FTP server supports the FTP protocol commands AUTH, PBSZ and PROT to permit TLS/SSL security of the control and data connections to be enabled and disabled.

- Extended status information

  The STAT command output contains TLS-specific information.

TLS/SSL support on the FTP server is described in detail in the “interNet Services Administrator Guide”.

3.5.2 TLS/SSL support in TELNET

TLS/SSL support on the TELNET client and TELNET server can be controlled in more than one way.

Selecting TLS/SSL support on the TELNET client

The following instruments are available to you for selecting TLS/SSL support on the TELNET client:

- Option file

  TLS/SSL support offers a wide range of selection options. These settings can be stored in one or more option files via options.

- TELNET client commands for TLS/SSL support

  These are commands for enabling/disabling TLS/SSL support, for reading the option file, etc.

TLS/SSL support on the TELNET client is described in detail in the section “Security in the TELNET client” on page 283.

Selecting TLS/SSL support on the TELNET server

The following instruments are available to you for selecting TLS/SSL support on the TELNET server:

- Option file

  TLS/SSL support offers a wide range of selection options. These settings can be stored in one or more option files via options.

- Installation command SET-FTP-TELNET-PARAMETERS

  The settings for TLS/SSL security can also be defined with the aid of additional parameters of the installation command.

TLS/SSL support on the TELNET server is described in detail in the “interNet Services Administrator Guide”.

3.5.3 TLS/SSL support in the mail reader

TLS/SSL can be used for connecting the mail reader to the POP3 server or IMAP server in order to allow secure transmission of the password.

TLS/SSL support is set in the configuration file of the mail reader (see section “POP3/IMAP servers: SERVER parameter section” on page 367).

3.5.4 TLS/SSL support in the mail sender

TLS/SSL can be used to secure the connection of the MAILCLNT subsystem to the mail server.

TLS/SSL support is set in the configuration file of the mail sender backend. TLS/SSL support in the mail sender is described in detail in the “interNet Services Administrator Guide”.

3.5.5 TLS/SSL support in the mail server

TLS/SSL can be used to secure connections from the mail server to the mail clients or to other mail servers.

TLS/SSL support is set in the configuration file of the mail server. TLS/SSL support in the mail server is described in the “interNet Services Administrator Guide”.

3.6 Specification of a cipher suite preference list

The specification consists of one or more cipher mnemonics which are separated by a colon (:).

A cipher mnemonic can take the following forms:

- A cipher mnemonic can consist of a single cipher suite such as DES-CBC-SHA.
- A cipher mnemonic can represent:
  - a list of cipher suites which contain a particular algorithm
  - cipher suites of a particular type

  For example, SHA1 represents all cipher suites which use the digest algorithm SHA1, and SSLv3 represents all SSL Version-3 algorithms.
- Lists of cipher suites can be combined to form a single cipher mnemonic with the aid of the “+” character. This is then interpreted as a logical AND operation. Thus SHA1+DES represents all cipher suites which contain the SHA1 and DES algorithms.
- Each cipher mnemonic can optionally be prefixed by one of the characters “!”, “-” or “+”:
  - If the prefix is “!”, the relevant cipher suites are permanently deleted from the preference list. Subsequently these no longer appear in the preference list even when they are specified explicitly.
  - If the prefix is “-”, the relevant cipher suites are deleted from the preference list, but some or all of them can be added again by means of subsequent options.
  - If the prefix is “+”, the relevant cipher suites are moved to the end of the preference list. This means that no cipher suites are added to the preference list, but only existing ones moved.
  - If none of the three characters “!”, “-” or “+” is prefixed, the cipher mnemonic is interpreted as a list of cipher suites which is appended to the current preference list. If this includes a cipher suite which is already contained in the current preference list, it is ignored. It is not moved to the end of the preference list.
- The cipher mnemonic @STRENGTH can be added at any position in order to sort the current preference list according to the length of the encryption key.

Permissible cipher mnemonics

The permissible cipher mnemonics are described below.

ALL
: All cipher suites with the exception of the eNULL ciphers. The latter must be enabled explicitly.

HIGH
: Cipher suites with key lengths greater than 128 bits.

MEDIUM
: Cipher suites with a key length of 128 bits or cipher suites downgraded due to other reasons.

LOW
: Cipher suites with key lengths of 64 or 56 bits, except Export cipher suites.

EXP, EXPORT
: Export encryption algorithms including 40- and 56-bit algorithms.

EXPORT40
: 40-bit Export encryption algorithms.

EXPORT56
: 56-bit Export encryption algorithms.

eNULL, NULL
: “NULL” encryption algorithms, in other words those without encryption. As these offer no encryption and thus present a security risk, they are by default disabled and, if required, must be specified explicitly.

aNULL
: Cipher suites without authentication. This means at present the anonymous Diffie-Hellman algorithms. These algorithms are vulnerable to “man in the middle” attacks, and you are consequently advised not to use them.

kRSA, RSA
: Cipher suites with RSA key exchange.

aRSA
: Cipher suites with RSA authentication, in other words the certificates contain RSA keys.

aDSS, DSS
: Cipher suites with DSS authentication, in other words the certificates contain DSS keys.

TLSv1.2, TLSv1.0, SSLv3
: TLSv1.2, TLSv1.0 or SSLv3 cipher suites.

  Note: There exist no TLSv1.1.specific cipher suites.

DH
: Cipher suites with Diffie-Hellman key exchange, including anonymous exchange.

ADH
: Cipher suites with anonymous Diffie-Hellman key exchange.

kEDH, kDHE
: Cipher suites with ephemeral Diffie-Hellmann key negotiation including anonymous suites.

kEECDH, kECDHE
: Cipher suites with ephemeral Elliptic Curve Diffie-Hellmann key negotiation including anonymous suites.

EECDH, ECDHE
: Cipher suites with ephemeral Elliptic Curve Diffie-Hellmann key negotiation without anonymous suites.

AECDH
: Anonymous Cipher suites with Elliptic Curve Diffie-Hellmann key negotiation.

ECDH
: Cipher suites with Elliptic Curve Diffie-Hellmann key negotiation including anonymous, ephemeral and fixed ECDH.

AES128, AES256, AES
: Cipher suites with AES encryption (key length of 128 or 256 bits or one of them).

3DES
: Cipher suites with Triple DES encryption.

DES
: Cipher suites with DES encryption (no Triple DES).

RC4
: Cipher suites with RC4 encryption.

RC2
: Cipher suites with RC2 encryption.

MD5
: Cipher suites with MD5 hash function.

SHA1, SHA
: Cipher suites with SHA1 hash function.

As it is just a matter of time until feasible attacks on SHA1 appear, you should switch as soon as possible to cipher suites that use the hash functions SHA256 or SHA384, for example. However, this usually also implies switching to TLS protocol version 1.2.

SHA256, SHA384
: Cipher suites using the SHA256 and SHA384 hash function respectively for the MAC (message authentication code) computation. In the case of cipher suites using AESGCM and hence AEAD (Authenticated Encryption with Associated Data) as the MAC method, the SHA256 and SHA384 respectively in the name has a different meaning.

aECDSA
: Cipher suites using ECDSA authentication, in other words, the certificates contain ECDSA keys.

AESGCM
: Cipher suites using AES in "Galois Counter Mode (GCM)". These cipher suites are only supported by TLSv1.2.

CAMELLIA128, CAMELLIA256, CAMELLIA
: Cipher suites that use 128 bit CAMELLIA, 256 bit CAMELLIA or either 128 or 256 bit CAMELLIA.

  The selecting effect of a preference list specification can be checked with the SHOW.CIPHERLIST procedure (see page 51)

The available cipher suites are listed in the table below.

Name	Version	Key exchange	Authen-tication	Encryption	Digest	Export
DHE-RSA-AES256-SHA	SSLv3	DH	RSA	AES(256)	SHA1	
DHE-DSS-AES256-SHA	SSLv3	DH	DSS	AES(256)	SHA1	
AES256-SHA	SSLv3	RSA	RSA	AES(256)	SHA1	
DHE-RSA-AES128-SHA	SSLv3	DH	RSA	AES(128)	SHA1	
DHE-DSS-AES128-SHA	SSLv3	DH	DSS	AES(128)	SHA1	
AES-128-SHA	SSLv3	RSA	RSA	AES(128)	SHA1	
DHE-DSS-RC4-SHA	SSLv3	DH	DSS	RC4(128)	SHA1	
EDH-RSA-DES-CBC3-SHA	SSLv3	DH	RSA	3DES(168)	SHA1	
EDH-DSS-DES-CBC3-SHA	SSLv3	DH	DSS	3DES(168)	SHA1	
DES-CBC3-SHA	SSLv3	RSA	RSA	3DES(168)	SHA1	

Available cipher suites

Name	Version	Key exchange	Authen-tication	Encryption	Digest	Export
RC4-SHA	SSLv3	RSA	RSA	RC4(128)	SHA1	
RC4-MD5	SSLv3	RSA	RSA	RC4(128)	MD5	
EDH-RSA-DES-CBC-SHA	SSLv3	DH	RSA	DES(56)	SHA1	
EDH-DSS-DES-CBC-SHA	SSLv3	DH	DSS	DES(56)	SHA1	
DES-CBC-SHA	SSLv3	RSA	RSA	DES(56)	SHA1	
DES-CBC3-MD5	SSLv2	RSA	RSA	3DES(168)	MD5	
RC2-CBC-MD5	SSLv2	RSA	RSA	RC2(128)	MD5	
RC4-MD5	SSLv2	RSA	RSA	RC4(128)	MD5	
RC4-64-MD5	SSLv2	RSA	RSA	RC4(64)	MD5	
DES-CBC-MD5	SSLv2	RSA	RSA	DES(56)	MD5	
EXP1024-DHE-DSS-RC4-SHA	SSLv3	DH(1024)	DSS	RC4(56)	SHA1	export
EXP1024-RC4-SHA	SSLv3	RSA(1024)	RSA	RC4(56)	SHA1	export
EXP1024-DHE-DSS-DES-CBC-SHA	SSLv3	DH(1024)	DSS	DES(56)	SHA1	export
EXP1024-DES-CBC-SHA	SSLv3	RSA(1024)	RSA	DES(56)	SHA1	export
EXP1024-RC2-CBC-MD5	SSLv3	RSA(1024)	RSA	RC2(56)	MD5	export
EXP1024-RC4-MD5	SSLv3	RSA(1024)	RSA	RC4(56)	MD5	export
EXP-EDH-RSA-DES-CBC-SHA	SSLv3	DH(512)	RSA	DES(40)	SHA1	export
EXP-EDH-DSS-DES-CBC-SHA	SSLv3	DH(512)	DSS	DES(40)	SHA1	export
EXP-DES-CBC-SHA	SSLv3	RSA(512)	RSA	DES(40)	SHA1	export
EXP-RC2-CBC-MD5	SSLv3	RSA(512)	RSA	RC2(40)	MD5	export
EXP-RC4-MD5	SSLv3	RSA(512)	RSA	RC4(40)	MD5	export
EXP-RC2-CBC-MD5	SSLv2	RSA(512)	RSA	RC2(40)	MD5	export
EXP-RC4-MD5	SSLv2	RSA(512)	RSA	RC4(40)	MD5	export
ADH-AES256-SHA	SSLv3	DH	none	AES(256)	SHA1	
ADH-AES128-SHA	SSLv3	DH	none	AES(128)	SHA1	
ADH-DES-CBC3-SHA	SSLv3	DH	none	3DES(168)	SHA1	
ADH-DES-CBC-SHA	SSLv3	DH	none	DES(56)	SHA1	
ADH-RC4-MD5	SSLv3	DH	none	RC4(128)	MD5	
EXP-ADH-DES-CBC-SHA	SSLv3	DH(512)	none	DES(40)	SHA1	export
EXP-ADH-RC4-MD5	SSLv3	DH(512)	none	RC4(40)	MD5	export
NULL-SHA	SSLv3	RSA	RSA	none	SHA1	
NULL-MD5	SSLv3	RSA	RSA	none	MD5	

Available cipher suites

Name	Version	Key exchange	Authen-tication	Encryption	Digest	Export
ECDHE-ECDSA-AES128-SHA	SSLv3	ECDH	ECDSA	AES(128)	SHA1	
ECDHE-ECDSA-AES256-SHA	SSLv3	ECDH	ECDSA	AES(256)	SHA1	
ECDHE-ECDSA-DES-CBC3-SHA	SSLv3	ECDH	ECDSA	3DES(168)	SHA1	
ECDHE-ECDSA-RC4-SHA	SSLv3	ECDH	ECDSA	RC4(128)	SHA1	
ECDHE-ECDSA-NULL-SHA	SSLv3	ECDH	ECDSA	none	SHA1	
ECDHE-RSA-AES256-SHA	SSLv3	ECDH	RSA	AES(256)	SHA1	
ECDHE-RSA-AES128-SHA	SSLv3	ECDH	RSA	AES(128)	SHA1	
ECDHE-RSA-DES-CBC3-SHA	SSLv3	ECDH	RSA	3DES(168)	SHA1	
ECDHE-RSA-RC4-SHA	SSLv3	ECDH	RSA	RC4(128)	SHA1	
ECDHE-RSA-NULL-SHA	SSLv3	ECDH	RSA	none	SHA1	
AECDH-AES256-SHA	SSLv3	ECDH	none	AES(256)	SHA1	
AECDH-AES128-SHA	SSLv3	ECDH	none	AES(128)	SHA1	
AECDH-DES-CBC3-SHA	SSLv3	ECDH	none	3DES(168)	SHA1	
AECDH-RC4-SHA	SSLv3	ECDH	none	RC4(128)	SHA1	
AECDH-NULL-SHA	SSLv3	ECDH	none	none	SHA1	
DHE-RSA-CAMELLIA256-SHA	SSLv3	DH	RSA	Camellia(256)	SHA1	
DHE-RSA-CAMELLIA128-SHA	SSLv3	DH	RSA	Camellia(128)	SHA1	
DHE-DSS-CAMELLIA256-SHA	SSLv3	DH	DSS	Camellia(256)	SHA1	
DHE-DSS-CAMELLIA128-SHA	SSLv3	DH	DSS	Camellia(128)	SHA1	
CAMELLIA256-SHA	SSLv3	RSA	RSA	Camellia(256)	SHA1	
CAMELLIA128-SHA	SSLv3	RSA	RSA	Camellia(128)	SHA1	
ADH-CAMELLIA256-SHA	SSLv3	DH	none	Camellia(256)	SHA1	
ADH-CAMELLIA128-SHA	SSLv3	DH	none	Camellia(128)	SHA1	
ECDHE-ECDSA-AES256-GCM-SHA384	TLSv1.2	ECDH	ECDSA	AESGCM(256)	AEAD	
ECDHE-ECDSA-AES128-GCM-SHA256	TLSv1.2	ECDH	ECDSA	AESGCM(128)	AEAD	
ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA	AESGCM(256)	AEAD	
ECDHE-RSA-AES128-GCM-SHA256	TLSv1.2	ECDH	RSA	AESGCM(128)	AEAD	
AES256-GCM-SHA384	TLSv1.2	RSA	RSA	AESGCM(256)	AEAD	
AES128-GCM-SHA256	TLSv1.2	RSA	RSA	AESGCM(128)	AEAD	
DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA	AESGCM(256)	AEAD	
DHE-RSA-AES128-GCM-SHA256	TLSv1.2	DH	RSA	AESGCM(128)	AEAD	
DHE-DSS-AES356-GCM-SHA384	TLSv1.2	DH	DSS	AESGCM(256)	AEAD	

Available cipher suites

Name	Version	Key exchange	Authen-tication	Encryption	Digest	Export
DHE-DSS-AES128-GCM-SHA256	TLSv1.2	DH	DSS	AESGCM(128)	AEAD	
ADH-AES256-GCM-SHA384	TLSv1.2	DH	none	AESGCM(256)	AEAD	
ADH-AES128-GCM-SHA256	TLSv1.2	DH	none	AESGCM(128)	AEAD	
ECDHE-ECDSA-AES256-SHA384	TLSvv1.2	ECDH	ECDSA	AES(256)	SHA384	
ECDHE-ECDSA-AES128-SHA256	TLSv1.2	ECDH	ECDSA	AES(128)	SHA256	
ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH	RSA	AES(256)	SHA384	
ECDHE-RSA-AES128-SHA256	TLSv1.2	ECDH	RSA	AES(128)	SHA256	
DHE-RSA-AES256-SHA256	TLSv1.2	DH	RSA	AES(256)	SHA256	
DHE-RSA-AES128-SHA256	TLSv1.2	DH	RSA	AES(128)	SHA256	
DHE-DSS-AES256-SHA256	TLSv1.2	DH	DSS	AES(256)	SHA256	
DHE-DSS-AES128-SHA256	TLSv1.2	DH	DSS	AES(128)	SHA256	
AES256-SHA256	TLSv1.2	RSA	RSA	AES(256)	SHA256	
AES128-SHA256	TLSv1.2	RSA	RSA	AES(128)	SHA256	
ADH-AES256-SHA256	TLSv1.2	DH	none	AES(256)	SHA256	
ADH-AES128-SHA256	TLSv1.2	DH	none	AES(128)	SHA256	

Available cipher suites

4 FTP

interNet Services includes, among other things, the TCP/IP-based application FTP. The **F**ile **T**ransfer **P**rotocol is used to transfer data between computers from different vendors independently of the operating systems running on them. The FTP protocol has been standardized, in other words the server commands and the responses are defined in RFC 959 and further RFCs for newer functions. Note, however, that not all FTP functions are mandated in RFC 959, so different implementations may have some minor deviations, depending on the vendor.

FTP works in accordance with the client/server principle which, in turn, requires the existence of two complementary processes. The initiator of a connection or request is called the client, whereas the partner who receives and responds to the request is called the server. The server process, which is usually permanently started, is known as a daemon. The client process is generally initiated by an appropriate call, e.g. *ftp*.

The FTP functionality in BS2000 is split into the server and client functionality.

4.1 FTP servers in BS2000

The FTP server accepts requests from FTP clients on the local network and executes them.

The use of an FTP server on a BS2000 partner system is, of course, only possible, provided the FTP server has been powered up by the administrator.

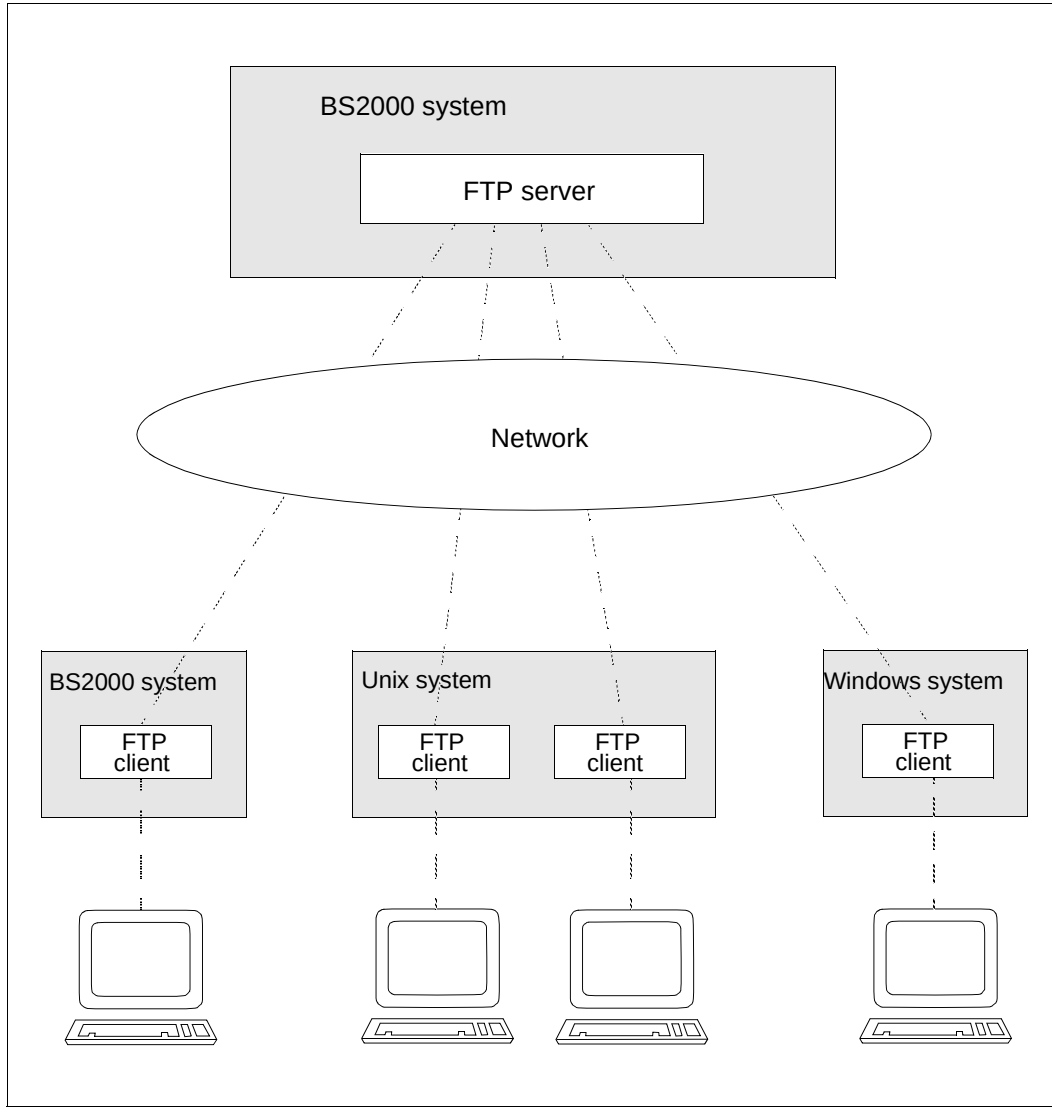


Figure 1: FTP server in BS2000

In the sections that follow, the host with the FTP server is referred to as the remote host so that the description of the server is consistent with that of the client (although from the viewpoint of the server, the local system is its own host).

Mode of operation

For each connection request from a client (*open* command), the server generates a separate BS2000 task. The data required to initiate this task (user ID, account number, password) is requested from the client. A program (FTPDC) is then initiated to execute the actual file transfer and the local file accesses initiated by the client. The DMS access rights in BS2000 (*RDPASS*, *WRPASS*, *EXPASS*, *SHARE=YES|NO*, *ACCESS=READ|WRITE*) as well as the access permissions in POSIX are both taken fully into account.

When multiple clients log on to the server at exactly the same time, a connection request may be rejected. If this occurs, the client must repeat the connection request.

Password protection

In BS2000, individual DMS files can be protected by passwords. The FTP server in BS2000 expects function calls with file names in the form

filename,C'password' or *filename,X'password'*.

Note, however, that this does not apply to the POSIX file system.

Example

A command is issued from a Unix system (file name: *ZWATON*) to overwrite the BS2000 file *ANTON*, which is protected by a read password *OTTO*:

```
put ZWATON ANTON,C'OTTO'
```

Files that are protected by different passwords (e.g. *RDPASS=C'OTTO'* and *WRPASS=C'KARL'*) cannot be addressed directly. At least one of the two passwords must have been entered earlier with the *site exec* function of the *quote* command:

```
quote site exec PASSWORD C'KARL'
```

quote site exec may only be used

- when FTP access is not controlled with FTAC and
- if it was not deactivated using the server option *-disableSiteExecCommand* (see the “interNet Services Administrator Guide”).

PASSIVE mode

PASSIVE mode enables a file transfer between two servers to be initiated from a client. The FTP client in BS2000 supports this procedure via the client command *proxy* (see page 172).

In the BS2000 FTP server this functionality is supported by the server commands PASV and EPSV. In the BS2000 FTP client PASSIVE mode can be selected using the client command *passive* (see page 168).

In PASSIVE mode, the server is made to wait at a data port for a connection request instead of actively setting up a connection itself. The connection is set up between the two server tasks via the first line address generated. If this line does not support connection between the two servers, the connection request is refused (return code 425, can't build data connection).

PASSIVE mode is also used when an FTP server is to be accessed behind a firewall as connections from the server often cause conflicts with the firewall rules. For this reason many clients use PASSIVE mode by default.

Scope of implementation

The functions of an FTP server are implemented in standardized protocol elements (server commands).

The following table contains a summary of the functions of the FTP servers in BS2000.

Server command	Server function	On servers in BS2000		Corresponding (k) / Issuing (a) client command
		(DMS)	(POSIX)	
acct	State account number	+	+	
abor	Abort file transfer	+	+	
allo	Allocate memory	(+)	(+)	
appe	Append file to existing file	+	+	append (a)
auth	Initiate TLS/SSL connection	+	+	open(a)
ccc	Enable/disable TLS security for the control connection	+	+	ccc (a)
cdup	Change to the next higher working directory (POSIX) or remove partial qualifier (DMS)	+(1)	+	cdup (a)
cmod	Enable 1:1 transfer	+	+	copymode(k)

FTP server functions

Server command	Server function	On servers in BS2000		Corresponding (k) / Issuing (a) client command
		(DMS)	(POSIX)	
cwd	Change working directory (POSIX) or remove partial qualifier (DMS)	+	+	cd (a)
dele	Delete file	+	+	delete (a)
eprt	Extended port command for IPv6	+	+	
epsv	Extended passive command for IPv6	+	+	
feat	Show new features of the server	+	+	
help	Request help	+	+	remotehelp (a)
list	List file names and directories	+	+	dir (a)
mdtm	Show date and time of last change to file	+	+	
mkd	Make directory	-	+	
mlsd	List file names and file properties	+	+	mlsd (a)
mlst	List file properties	+	+	mlst (a)
mode	Define transfer mode	+	+	mode (a,k)
nlst	List file names	+	+	ls (a)
noop	No operation	+	+	
opts	Specify options	+	+	
pass	State password	+	+	
pasv	Set PASSIVE mode	+	+	
pbsz	Assign memory for data encryption	(+)	(+)	private (a)
port	Define data connection port	+	+	
prot	Enable/disable encryption of data connection	+	+	private (a)
pwd (same as xpwd)	Show current working directory	+	+	pwd (a)
quit	Terminate session	+	+	close (a)
rest	Specify position in file where data transfer is to commence	+	+	reget (a) reput (a)
retr	Retrieve a file	+	+	get (a)
rmd (same as xrmd)	Remove directory	-	+	

FTP server functions

Server command	Server function	On servers in BS2000		Corresponding (k) / Issuing (a) client command
		(DMS)	(POSIX)	
rnfr	Rename a file, output old name	+	+	rename (a)
rnto	Rename a file, output new name	+	+	rename (a)
site	Prefix for BS2000-specific FTP server functions	+	+	
size	Output file size	+	+	
stat	Show server status information	+	+	status (k)
stor	Store file	+	+	put (a)
stou	Store files with unique names	+	+	put (a)
stru	Define transfer structure	+	+	struct (a+k)
syst	Show system information on the server	+	+	system (a)
type	Define transfer type	+	+	type (a+k)
user	Define user ID	+	+	user (a)
xcup (same as cdup)	Change to the next higher working directory (POSIX) or remove partial qualifier (DMS)	+(1)	+	
xcwd (same as cwd)	Change working directory (POSIX) or remove partial qualifier (DMS)	+	+	
xmkd	Make directory	-	+	mkdir (a)
xpwd	Show current working directory	+	+	pwd (a)
xrmd	Remove directory	-	+	rmdir (a)

FTP server functions

Key

+ The function is implemented.
(+) The function is implemented as a dummy function.
- The function is not implemented
(a) Client command issues the server command.
(k) Client command has the same meaning as the server command.
(a+k) Client command has the same meaning as the server command and issues the server command.

(1) On the BS2000 server, *cdup* and *xcup* achieve the function "Change working directory" by removing a partial qualifier.

BS2000-specific functions of the FTP server

The following table provides an overview of the BS2000-specific functions of the FTP server.

Server command	Server function	Servers in BS2000 (DMS)	(POSIX)	Corresponding (k) / Issuing (a) client command
cmod	Enable 1:1 transfer	+	+	copymode (k)
exec	Forward command to remote operating system	+	+	
file	Change DMS file attribute	+	+	file (k)
ftyp	Change file editing type	+	+	ftyp (k)
help	Information on FTP commands	+	+	help (k)
modc	Modify character string for switching between POSIX / DMS	+	+	modchar (k)
setc	Change code tables	+	+	setcode (k)
sfil	Enable/disable special EOF marker for PAM files; Enable/disable filling of blank SAM records	+	(+)	setfile (k)
svfs	Enable/disable TVFS	+	+	
exit	Define parameters for exit routines	+	+	rexit (a)

BS2000-specific FTP server functions

Key

+ The function is implemented.
(+) The function is implemented as a dummy function.
(k) Client command has the same meaning as the server command.

It is recommended that BS2000-specific FTP server functions always be called with the prefix *site*, although these functions can still also be used for compatibility reasons at the moment with the *file*, *ftyp*, *modc* and *setc* commands without a preceding *site*.

However, it is strongly recommended, particularly when creating procedures, to use the current form with a preceding *site* to prevent problems that may arise if the FTP standard is extended.

If *site* is not followed by any of the commands listed in the table, *site* is interpreted in the same way as *site exec*. The string following *site* is then forwarded as a command to the remote operating system and is interpreted as in earlier FTP server versions.

Calling the FTP server functions via the FTP partner client

For most FTP server functions, there is a corresponding command call on the partner client. If a function has no corresponding command call on the client, it can be called directly via the FTP server with the aid of the client command *quote* (see section “quote - Call server functions” on page 178).

In the case of FTP clients that do not offer any special input options for an account number (e.g. web browser), the account number can be appended to the user ID separated by a comma on the BS2000 FTP server.

There is no meaningful equivalent for the "Make directory" and "Remove directory" functions in the DMS file system on BS2000. As far as the POSIX file system on BS2000 is concerned, these functions are implemented within the framework of the POSIX support. The FTP client commands *mkdir* and *rmdir* can be used to create and remove directories in the POSIX file system.

Any server function of the partner system can always be called with the client command *quote*.

The server functions *file*, *ftyp*, *modc*, *setc*, *sfil*, *svfs*, *cmod* and *exit* are described in detail below. These functions exist only in the DMS file system in the BS2000 FTP server and must always be addressed via the client command *quote*.

- The *site exit* function defines the parameter for the exit routines (see the “interNet Services Administrator Guide”). The *exit* command can also be sent to the server via the client command *rexit* (see page 186).
  Enter the following:

  quote site exit receive:<recv-parm>

  quote site exit send:<send-parm>

  quote site exit receive:<recv-parm>!send:<send-parm>

  e.g.: *quote site exit receive:-C5!send:-D7*
  *quote site exit send:*NONE*

- The *site file* function defines the file attributes of a DMS file to be transferred on the remote system (i.e. the system on which the FTP server is running). The *site file* function corresponds to the client command *file* and is described in detail on page 133.
  Enter the following:
  quote site file <remote-file,file-operand-list>
  e.g.: *quote site file file1,fcbtype=sam*

- The *site ftyp* function defines whether the SAM files on the remote computer are to be processed as text or binary files The *site ftyp* function corresponds to the client command *ftyp* and is described in detail on page 136.
  Enter the following:
  quote site ftyp <file-processing-type>
  e.g.: *quote site ftyp binary*

- The *site modc* function defines the first character in the string for switching between the DMS file system and the POSIX file system. The *site modc* function corresponds to the client command *modchar* and is described in detail on page 161.
  Enter the following:
  quote site modc <character>
  e.g.: *quote site modc $*

- The *site setc* function is used to set the code tables for code conversion. The *site setc* function corresponds to the client command *setcode* and is described in detail on page 193.
  Enter the following:
  quote site setc <EBCDIC-table> <ISO-table>
  e.g.: *quote site setc EDF049 ISO88599*

- The *site sfil* function defines special modes of operation for transferring files and corresponds to the client command *setfile*, which is described in detail on page 194.
  Enter the following:

  quote site sfil datend on | off | lbp
  Enable/disable the special EOF marker (default: enabled) or use the new EOF marker mode Last Byte Pointer (LBP).

  quote site sfil pademptyrec on|off
  Enable/disable filling of blank SAM records (default: disabled).

  e.g.: *quote site sfil datend off*
  quote site sfil pademptyrec on

- The *site cmod* function (see the description of the client command *copymode* on page 126).

- The *site svfs* function enables/disables TVFS for the corresponding session.
  Enter the following:

  quote site svfs on | off
  Enable/disable TVFS.

  e.g.: *quote site svfs on*

FTP server commands, which support the restart capability of the FTP client

The following FTP server commands support the restart capability of the FTP client and FTP server:

– mdtm
– size
– rest

mdtm - Establish the date and time of the last file change to date

The *mdtm* command provides the date and time of the last change to a file to date:

Message: `213 <YYYYMMDDhhmmss>`

mdtm
<file>

<file>
File for which the *mdtm* command supplies the date and time of the last change.

Example

```
quote site mdtm testdatei
213 20101015204331
```

size - Establish the size of a file

The *size* command specifies how many bytes were transferred via the network when this file was transferred. The current settings for *mode*, *type*, *struct* and *ftyp* are taken into account here. The *size* command is rejected with an error code for *mode* <> *stream*.

It can occur that the *size* command is disabled for technical reasons on a BS2000 FTP server.

Message: 213 <size of file (in bytes)>

size
<file>

<file>
File for which the *size* command supplies the size.

Example

```
quote size test file
213 498665
```

rest- Specify file position where a data transfer should begin

The *rest* command specifies any byte position at whose corresponding file position (instead of file start) a file transfer initiated by a subsequent *stor* or *recv* command should begin.

rest
<position>

<position>
Byte position at whose corresponding file position the next file transfer should begin.

Command for showing the server features

The FTP server supports the *FEAT command* (RFC 2389). By default, *FEAT* reports support of the *EPSV, EPRT, SIZE*, *MDTM* and *REST STREAM* commands. Support of *SIZE* is not indicated if *SIZE* has been disabled via server option *-disableSizeCommand* (see the “interNet Services Administrator Guide”).

If TLS support of the FTP server is enabled, *FEAT* also reports support of *AUTH TLS*, *PBSZ* and *PROT*.

If TVFS is enabled globally using the FTP server option, the FEAT command reports TVFS support.

In addition, support for the MLSD/MLST commands is reported initially via MLST *type*;size*;create*;modify*;perm*;unique*;UNIX.owner*;UNIX.group*;UNIX.mode*;*. In doing so, the list next to MLST indicates which facts can be supplied by the MLSD/MLST commands using file system objects. '*' indicate the facts that are currently returned; you can use the OPTS command to change the quantity of these facts. For details on TVFS, MLSD, and MLST, see RFC 3659.

FEAT

FTAC interface

The advantages of the FTP protocol lie, among other things, in the fact that FTP client programs are standardized and hence widely used. When seen from a security aspect, however, it is not up to the usual standards prevalent in BS2000. In other words, anyone who knows your login-specific data can retrieve data from your user ID, save data to your user ID, delete data or change file attributes.
For this reason, interNet Services offers access to the FTAC interface for FTP. Access control and protection via openFT-AC has already been available for the File Transfer openFT since quite some time.

The following features are offered by FTAC to protect the BS2000 server:

– Decoupling of FTP transfer admissions and login admissions
– Access rights depending on partner systems
– User-specific access rights
– Flexible levels of access rights
– Logging of every authorization check
– Simple application

More detailed information on the FTAC support for FTP can be found in the chapter “FTAC interface” (see page 219).

4.2 SNMP subagent for FTP

The FTP server has its own subagent (FTP subagent), which is operated via a management application, the BCAM Manager.

The “SNMP Management for openNet Server and interNet Services” manual contains information on the following topics:

– Handling the BCAM Manager
– Software requirements
– Installation and deinstallation
– Starting up and shutting down the FTP subagent

Interaction between the FTP subagent and FTP server

The FTP server accesses the FTP subagent via the fixed port number 3237. Immediately after starting up, the FTP server reports to the FTP subagent, provided this subagent is also started, and provides it with the following information:

– Port number, under which the FTP subagent can access the FTP server
– Server port number for the control connection to the FTP clients

Assuming a server entity with this server port number does not already exist, the FTP server creates the relevant server entry.

Each FTP server writes its two port numbers to the SYSDAT.TCP-IP-AP.*nnn*.SNMP file at startup. If the FTP subagent is only started subsequently, it can check the SYSDAT.TCP-IP-AP.*nnn*.SNMP file for the currently active FTP server and create the relevant data structures.

If the FTP server is terminated, it deletes its entry from the SYSDAT.TCP-IP-AP.*nnn*.SNMP file.

4.3 1:1 transfer of BS2000 disk files

1:1 transfer of BS2000 disk files retains file attributes. A homogeneous interconnection is required for this, in other words the source and target systems must be BS2000 systems.

In the event of1:1 transfer, the file format attributes that remain unchanged include the following:

– Blocking factor
– Block structure (PAMKEY, DATA, NO)
– Record format
– Definitions of the ISAM key

Key contents are also sent in 1:1 transfer.

In addition to the file attributes that retain content and structure, file protection attributes (USER-ACCESS, ACCESS, BASIC-ACL, AUDIT, RETENTION-PERIOD), with the exception of passwords and GUARDS rules, can also be transferred to the target file.

With 1:1 transfer it is also possible to store BS2000 files temporarily on non-BS2000 systems and then configure them on BS2000 systems. This corresponds to transfer in Transparent mode in openFT.

– Transfer is not possible if there is a PAMKEY on NK disks or an odd blocking factor on NK4 disks. In these cases you must first of all convert these disks with the PAMCONV utility routine.
– The structure of the transferred files is not revealed. Thus it is not sensible to use code conversion routines implemented with an FTP exit routine, for example. It is recommendable to disable such exit routines during a 1:1 transfer.
– 1:1 transfer of tape files is not supported.

Enable 1:1 transfer

You enable/disable 1:1 transfer as follows:

– On the FTP client side using the *copymode* command (see page 126).
– On the FTP server side using the proprietary command CMOD, which you issue on the FTP client with *quote site cmod …*

4.4 FTP client in BS2000

Within BS2000, each FTP user opens his or her own FTP client task. By using the *open* command, the client establishes a link to the required FTP server on the remote host.

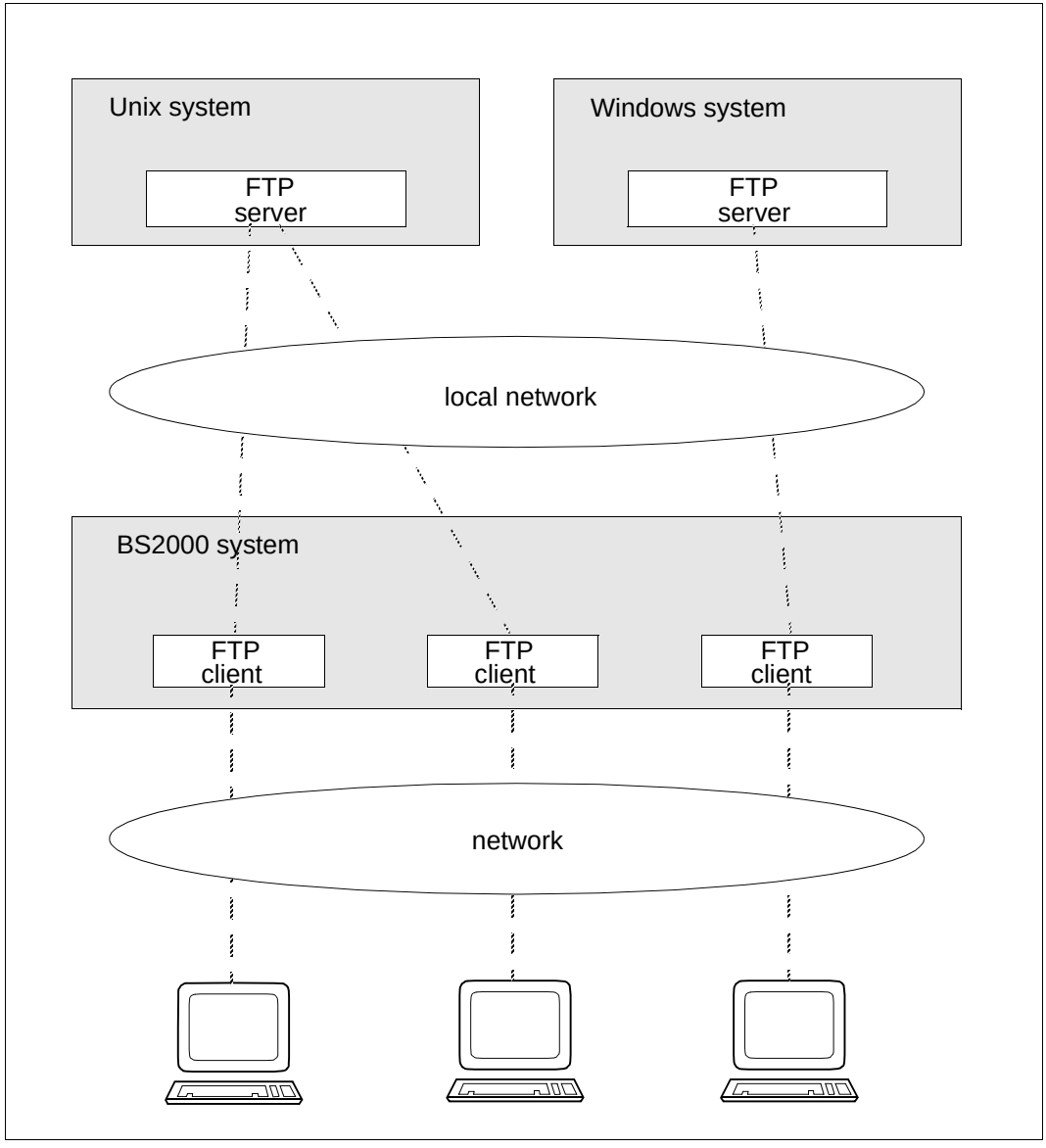


Figure 2: FTP clients in BS2000

Starting and terminating the client

You start the FTP client as follows:

START–FTP or FTP

The commands for terminating the FTP client are *quit* or *bye*.

Opening and closing a connection

Before any operation that (also) affects a remote host can be executed, a connection must be set up. This is done by using the *open* command. If an access protection mechanism exists on the remote host, FTP will first request the required entries. In BS2000, the transfer admission (i.e. the access authorization) typically consists of the values for the user ID, password and account number. These values are system-dependent, as can be seen in the following table.

If the partner is an FTAC-protected BS2000 system, the transfer admission must be specified in accordance with the FTAC setting

System	Login name	Account number	Password
BS2000 without FTAC protection	1 - 8 alphanumeric characters	1 - 8 alphanumeric characters	C string, 1 - 8 characters long or X string, 1 - 16 characters long
BS2000 with FTAC protection	$FTAC or relevant string defined for server installation		C string, 8 - 32 characters long or X string, 15 - 64 characters long
Unix system	1 - 32 characters	The Unix system does not recognize any account numbers locally	Alphanumeric characters (the length is system dependent); a distinction is made between uppercase and lowercase
Windows	1 - 36 characters	Windows does not recognize any account numbers locally	C string, 8 - 32 characters long or X string, 15 - 64 characters long
MVS	1 - 7 alphanumeric characters	Max. 40 characters including uppercase letters, digits and special characters $, @, #	1 - 8 alphanumeric characters

The connection to the remote host can be cleared with the *close* command.

In interactive mode, FTP responds with a version number and the input prompt *ftp>*. If the authorization entries following the *open* command are invalid, the connection to the remote host is kept open, provided the task switch 1 is not set. The correct entries can then be

made via the *user* command. Setting task switch 1 enables a SPIN-OFF mechanism (but see FTP client commands *jobvar* and *svar*). If the entries passed to the access protection mechanism are invalid, the FTP client is terminated. Prompting is enabled automatically on starting FTP in interactive mode and can be disabled with the *prompt* command.

In batch mode, after the *open* command, the authorization entries (user ID, password, account number) must be specified individually on continuation lines if they are required by the remote host. If these entries are invalid, the FTP client terminates. In batch mode, commands are read with RDATA and must be entered in lowercase letters. Prompting is disabled on starting FTP in batch mode.

Note that task switch 1 must be set for batch mode.

Command	Function	Page
open	Open connection to a remote host	165
close	Close connection to the remote host	125
!	Switch to BS2000 command mode	211
bye	Exit FTP	120
quit	= bye	177

Commands to control FTP

The following commands return current information on the client and server settings, command states and on trace and debug settings:

Command	Function	Page
help	Display summary information on FTP commands	142
?	= help	210
status	Display FTP STATUS information	196
remotehelp	Display information on functions of the remote FTP server	182
debug	Enable/disable DEBUG output	128
trace	Enable/disable TRACE output	203
verbose	Enable/disable server responses	208
system	Display information on FTP server	201

General information on FTP

Directories

After a connection has been successfully established, a working directory is defined on the local host as well as the remote host. If desired, you can obtain information on the current local and remote working directories with *lpwd* and *pwd*, respectively. Existing files in the local working directory can be listed with the command *ldir* or *lls*, and files in the remote working directory can be listed with *dir* or *ls*. When files are not identified by their full path names in commands, they are considered to be relative to the currently set working directory.

Local file names must comply with the conventions of the local host, and remote file names must comply with the conventions of the remote host. Special attention must be paid to the use of uppercase and lowercase characters.

Command	Function	Page
ldir	Information on local file(s)	147
lpwd	Output local working directory	149
lls	List names of files on local host	148

Information on local files and directories

Command	Function	Page
dir	Information on remote file(s)	130
mdir	Information on remote file names in multiple directories on the remote host	153
ls	List names of files on remote host	150
mls	List names of files in in multiple directories on the remote host	157
mlsd	Output information on remote files/directories in machine readable format	158
mlst	Output information on one remote file or directory in machine readable format	160
pwd	Output remote working directory	176

Information on remote files and directories

BS2000 can optionally work with two different file systems: the proprietary DMS file system and the POSIX file system (based on a Unix system). In BS2000, directories are emulated by addressing groups of files using partial qualifiers. The complete or absolute path in BS2000 corresponds to the fully-qualified file name (e.g.:*CATID:$USERID.FILENAME*). The

lpwd command can be used to check which file system is currently enabled on the local host. Setting the working directory enables you to switch between the DMS file system and the POSIX file system.

In order to access files in either the DMS or POSIX file system, you must be in the current working directory of the corresponding file system. It is not possible to access POSIX files from within the DMS file system, or vice versa. This principle must be observed for all FTP commands and functions.

Some operating systems (e.g. Unix or Windows systems) recognize a hierarchically organized system of directories.The structure of the POSIX file system corresponds to that of the Unix file system (UFS), i.e. is organized in a hierarchy. In the DMS file system, directories are emulated by addressing groups of files using partial qualifiers.

Example

```
:5:$TCPTEST.AAA.FILE1
:5:$TCPTEST.AAA.FILE2
:5:$TCPTEST.AAA.BBB.FILE3
:5:$TCPTEST.AAA.BBB.FILE4
:5:$TCPTEST.AAA.CCC.FILE5
:5:$TCPTEST.DDD.FILE6
:5:$TCPTEST.FILE7
```

Here *:5:$TCPTEST* is the main directory containing all files. *AAA* and *DDD* are subdirectories of the main directory. *BBB* and *CCC* are subdirectories of *AAA*. The part of the file specification consisting of the directory names is designated as the path name (for example, *:5:$TCPTEST.AAA.BBB* is the path name for the files *FILE3* and *FILE4*).

Switching between the DMS and POSIX file systems

Switching from the DMS to the POSIX file system, or vice versa, is achieved by entering the character string that was set for this purpose. Note that the entry is case-sensitive.

The default string is:

%POSIX / %posix to switch to the POSIX file system
%BS2000 / %bs2000 to switch to the DMS file system

The first character in this string (% character) can be changed for the local system with the client command *modchar* and for the remote system with the server function *site modc*. There is no separate client command for the server function *site modc*, so *site modc* has to be invoked with the client command *quote*.

The client command for changing the file system is either *lcd* or *cd*, depending on whether you want to change the local or remote working directory. For example, the command for switching to the POSIX file system on the local computer is *lcd %POSIX*.

A switch always takes you to the HOME directory of the target file system. In the POSIX file system, the system administrator defines the name of the HOME directory, whereas in the DMS file system, it corresponds to *:CATID:$USERID*.

TVFS (Trivial Virtual File System)

TVFS is a method described in RFC 3659 for addressing the files that can be accessed from the FTP server with Unix-like path names in which the subdirectory names are each separated by "/".

For POSIX files, the changes are therefore only minimal.

In the case of the TVFS implementation selected in the BS2000 FTP server, files in the DVS are accessed via a virtual directory /%BS2000, where the name %BS2000 is identical to the string described in the previous section, which can be used to switch from the POSIX to the DVS file system without TVFS (the version written in lowercase is not supported). In other words, the name of the virtual directory changes if the first character of the string changes with the *site modc* server function.

You can enable or disable the TVFS globally using the FTP server option (-TVFS ON/OFF) or for specific FTP sessions using the server command (*site svfs on* | *off*).

If the TVFS is enabled, this means the following for switching from POSIX files to DVS files:

– you enter either the absolute path /%BS2000 (instead of simply %BS2000 without TVFS),

– or you specify a relative path, which leads from the current POSIX directory to this virtual directory, which therefore contains %BS2000 as the last directory.

The /%BS2000 directory contains subdirectories in the form :<catid>:$<userid>, which in turn contain only the associated BS2000 file names but no further subdirectories. In other words, there is no emulation of subdirectories – contrary to the non-TVFS case – using partially qualified file names that end with a period.

If an FTP session is established, /%BS2000 initially contains only one entry with the default pubset of the respective login user ID. If the user switches to different combinations of catid and user ID using the *cwd* or *xcwd* server command, these are then cached. In other words, when listing the /%BS2000 directory, all combinations of catid/user ID visited since the beginning of the session are listed.

TVFS offers few advantages to human users; rather, the discontinuation of the emulation of subdirectories using partial qualifiers presents disadvantages. GUI FTP clients, on the other hand, work more reliably and in a more easily understandable way for the user if they can be supported by a standardized TVFS file system instead of having to operate on a file system using heuristics whose rules they often do not (fully) know. If GUI FTP clients are

used, they should be instructed to send a *site svfs on* to the BS2000 FTP server at the beginning of the FTP session. How this can be achieved in each case is described in the documentation for the respective client.

File management commands

The following file management commands are available for local and remote files and directories:

Command	Function	Page
lcd	Change local working directory	145

Modifying local files and directories

Command	Function	Page
mkdir	Create a remote working directory	156
rmdir	Remove a remote working directory	187
cd	Change remote working directory	122
delete	Delete remote file	129
mdelete	Delete multiple remote files	151
rename	Rename remote file	183
cdup	Change to the next higher directory (one level up)	124

Modifying remote files and directories

Metacharacters in file names

File names for local and remote files are operands in many FTP commands. In some commands, the file name must designate only one file; in other cases, you may also specify file groups by entering metacharacters (e.g. * and ?).

The syntax and semantics of metacharacters are not standardized and depend on the special implementation. Since special implementations are, for the most part, supported by existing operating system calls, the permitted metacharacters are identical to those generally allowed in the operating system. In most situations, the FTP server determines whether and which metacharacters are allowed, though this can also be done by the FTP client in some cases. The following table contains an overview of the metacharacters allowed in BS2000, Unix and Windows systems.

Function	BS2000 system		Unix system	Windows system
	DMS	POSIX		
Match any string (including the null string)	*	*	*	*
Match exactly one character	/	No equivalent	?	?
Match the listed strings	<s1,...>	No equivalent	No equivalent	No equivalent
Match a string that lies between strings s1 and s2 in lexical order	<s1:s2>	No equivalent	[s1-s2] (*)	No equivalent
Exclude the corresponding files	-s1 (only at the beginning of the string)	No equivalent	No equivalent	No equivalent

Metacharacters in file names
(*) only single characters are permitted

Backslash

Since the FTP client in BS2000 strips single backslashes, any backslash that is to be used as a directory separator, for example, must be entered twice in succession.

File passwords

In BS2000, individual DMS files may also be protected by file passwords. The BS2000 FTP server expects function calls with file names in the form:

```
<filename>,C'<password>' or <filename>,X'<password>'.
```

This does not apply to the DMS file system.

Example

Overwrite the BS2000 file *ANTON* protected by the read password *OTTO* from within a Unix system with the file name *ZWATON*:

```
put ZWATON ANTON,C'OTTO'
```

File conversion on transfer (not 1:1 transfer)

When transferring files from one operating system to another using FTP, two viewpoints need to be taken into account:

- Different operating systems recognize and accept different file types. For example, BS2000 recognizes, for example, SAM, ISAM and PAM files; Unix and Windows systems recognize only unstructured files.
- Different operating systems use different codes to represent characters. For example, BS2000 systems encode characters in EBCDIC; Unix and Windows systems use ASCII.

If XHCS is used, it is possible to switch between any of several ASCII/EBCDIC code conversions. The client command for switching conversion modes is *setcode*. The corresponding server command is *quote site setc* (see page 178).

The type of transfer influences the conversion of file types and codes. The following applies locally for a BS2000 FTP client:

- When the transfer type is ASCII (default value), code conversion takes place from EBCDIC to ASCII (send) or vice versa (receive). It is possible to read SAM files, ISAM files (without key) and PAM files (without PAMKEY). By default, a SAM file with a variable record length is generated, but this default can be changed with a command.
- When the transfer type is BINARY, no code conversion occurs. It is possible to read SAM files, ISAM files (without key) and PAM files (without PAMKEY). A PAM file is generated by default, but this default can also be changed with a command.
- When the transfer type is EBCDIC, no code conversion occurs. It is possible to read SAM files, ISAM files (without key) and PAM files (without PAMKEY). A SAM file with a variable record length is generated.

In addition, file attributes and the processing type can be modified by using the *quote site file* or *quote site ftyp* commands:

- *quote site file*:
  The file attributes for a file to be created can be preset by defining whether a PAM file or SAM file is to be created. All file attributes supported by C (BS2000) for STREAM I/O are also supported here.

- *quote site ftyp*:
  The processing type when writing a SAM file can also be preset. The processing type determines whether the files are to be processed as text files (i.e. organized in records or lines) or as binary files (stream of bytes) when they are written.

  – PAMKEYs are not transferred. This means that it is not possible to interchange executable files between two BS2000 systems. This restriction can be overcome by packaging executable files in PLAM libraries.

  – ISAM files lose their ISAM keys on transfer and must hence also be always packaged in PLAM libraries before being transmitted.

How you transfer BS2000 disk files while still retaining their attributes is described in the section “1:1 transfer of BS2000 disk files” on page 80.

Command	Function	Page
user	Specify user ID for remote host	206
ascii	Initiate ASCII transfer type	116
binary	Initiate BINARY transfer type	119
copymode	Enable 1:1 transfer of BS2000 disk files	126
file	Change DMS file attributes	133
ftyp	Change or query DMS file processing type	136
modchar	Modify string for switching between BS2000 and POSIX file systems	161
setcode	Change code tables	193
setfile	Enable/disable usage of special EOF markers and filling of dummy records with blanks	194
tenex	= binary	202
type	Change or query transfer type	204
mode	Change or query transfer mode	162
struct	Change or query transfer structure	198
form	Change or query transfer format	135
runique	Set unique target file name on overwriting with get	188
sunique	Set unique target file name on overwriting with put	199

Commands to control data transmission

Transferring files

The actual file transfer can be initiated for one or more files, regardless of the direction of transfer. When transferring multiple files interactively, the *prompt* command can be used to control whether or not confirmation is required for each individual file.

An abort of the file transfer initiated by the client is implemented by the FTP server via the server command *abor*. In such cases, the BS2000 FTP server reports a successful abort with the messages below. Note, however, that this abort will not function if the FTP access occurs via FTAC.

```
226 Abort successful
426 Transfer aborted. Data connection closed.
```

The FTP client in BS2000 recognizes a user´s intention to abort a file transfer by pressing the K2 key. If the K2 key is pressed repeatedly, it will no longer be possible to return to FTP.

Command	Function	Page
append	Append a local file to a remote file	115
get	Fetch a file	138
recv	= get	180
mget	Fetch multiple remote files	154
reget	Fetch a file after a transfer abort	181
put	Send a local file	175
send	= put	189
mput	Send multiple local files	163
reput	Send a local file after a transfer abort	184

Transferring files

Forwarding commands

Functions that have been implemented on the FTP server of the remote host and for which there are no corresponding calls on the own FTP client can be initiated directly by using the *quote* command. The required parameters are passed transparently, without any validation on the FTP client The *remotehelp* command can be used to determine which functions have been implemented on the FTP server of the remote host.

One possible server function that must be initiated by using *quote* is *site exec*. In the BS2000 implementation, *site exec* permits commands to be forwarded to the operating system of the remote computer. Note, however, that *site exec* cannot be used in combination with the FTAC functionality, unless different behavior has been configured using the server option *-disableSiteExecCommand* (see the “interNet Services Administrator Guide”).

Command	Function	Page
bell	Enable/disable bell	118
hash	Enable/disable transfer progress indicator	141
quote	Transfer parameters to the remote FTP server	178
glob	Enable/disable expansion of metacharacters	140
prompt	Enable/disable prompting	170
sendport	Enable/disable port command	190
settime	Set timeout value for server responses	195
exit	Define parameters for local exit routines	132
rexit	Define parameters for remote exit routines	186
jobvar	Enable/disable usage of a job variable	143
svar	Enable/disable usage of an S variable (SDF-P)	200
passive	Enable/disable PASSIVE mode	168

Other FTP functions

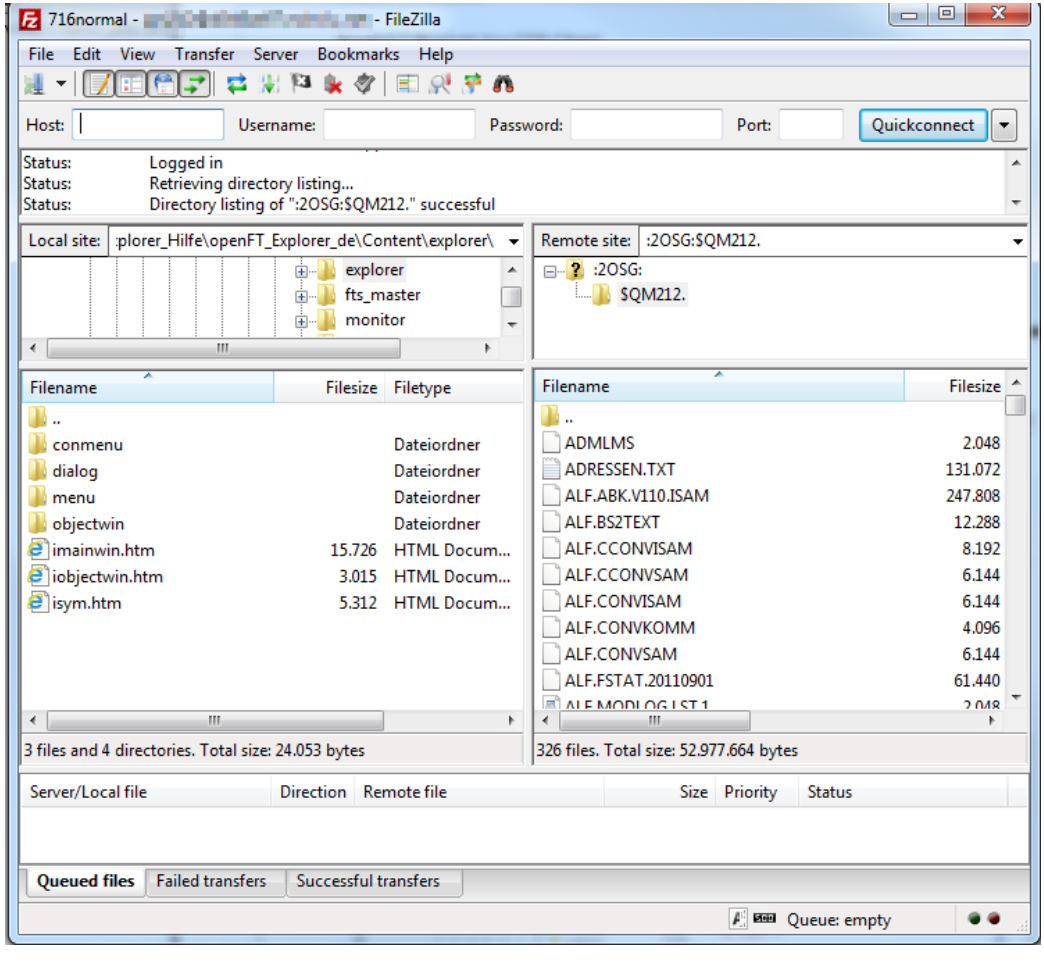
Restart capability of the FTP client

With the restart capability of the FTP client, aborted FTP transfers can be resumed at the position in the target file at which the FTP transfer was interrupted. The restart capability of the FTP client is implemented with the FTP client command *reget* (see page 181) and *reput* (see page 184).

Batch capability of the FTP client

The FTP client can be used in (batch) procedures. The batch capability of the FTP client allows the procedure to establish the command for which the error occurred. The faulty client command or the associated faulty server command and the associated error messages are stored in a previously specified job variable and/or SDF-P variable so that the calling procedure can respond suitably. The batch capability is supported by the FTP client commands *jobvar* (see page 143) and *svar* (see page 200).

Example of an FTP session between a PC and BS2000



4.5 FTP client in POSIX

If the FTP client is running under POSIX, it issues the following start message:

```
POSIX-FTP <vers> <date> <time>
```

<date> and <time> here specify the date and time when the main module was compiled. At startup time, the FTP client switches to the POSIX directory.

The FTP call with switch -n in POSIX corresponds to batch mode in BS2000.

Example

Information about remote files is to be written into a file dir.erg:

```
ftp -n <remote host> >>dir.erg <<END

user <id> <password> <account>
dir
close
bye
END
```

TLS/SSL support of the FTP client in POSIX

TLS support of the FTP client is implemented in the same way as TLS support of the FTP client in BS2000 (see the section “TLS/SSL support in the FTP client” on page 95).

The private option file has the name $HOME/.ftp.options. The file names relate to the POSIX file system. If you want to use a BS2000 file as CertificateFile, you must prefix the file name with “/BS2/”.

4.6 TLS/SSL support in the FTP client

interNet Services supports the control and - optionally - the data connection with the aid of the TLS/SSL protocol.

TLS/SSL support is implemented by the following measures:

- Option file

  TLS/SSL support offers a wide range of selection options. These settings can be stored in one or more option files via options (see the section “Parameter selection using option files” on page 96).

- FTP client commands for TLS/SSL support

  These are commands for enabling/disabling TLS/SSL support, for reading the option file, etc. (see the sections “Commands for TLS/SSL support” on page 112 and “Overview of commands (FTP client)” on page 114).

- Extended status information

  The *status* command output contains TLS-specific information (see the section “status - Display FTP status information” on page 196).

In the following, “TLS/SSL” will be referred to as “TLS” for short.

4.7 Parameter selection using option files

Settings which are particularly important for using TLS are stored in one or more option files. The option file is read in when the FTP client is started. However, you can also adapt the option setting to suit current requirements later by reading in an option file with the FTP client command *readopt* (see page 179).

Determination of the relevant option file by the FTP client

The FTP client proceeds as follows to determine the relevant option files:

1. First the FTP client searches for a centrally stored option file with the default file name $.SYSDAT.TCP-IP-AP.*nnn*.F-GOPT

   In the event of installation with IMON you can redefine the logical ID SYSDAT.F-GOPT to refer to another file which the FTP client then uses as option file.

2. Regardless of the central file named under 1.), the FTP client also searches for a user-owned option file under the name SYSDAT.TCP-IP-AP.*nnn*.FTP.OPT. If this file exists, the FTP client reads the options from there.

If an option is included in both files but with different values, the value defined in the user-owned file applies.

If settings for TLS security have been specified using the FTP client command, these have priority over those specified in the option file.

Details of the special aspects relating to TLS support of the FTP client in POSIX are provided on page 94.

Notation of the options in the option file

The various options must be entered in the option file according to the following rules:

- Each option must be in a separate line.
- If an option's arguments extend over more than one line, each line that is to be continued must be terminated with the continuation character "\".
- A line beginning with the character "#" in column 1 is ignored when the file is read in.
- The option names are not case-sensitive.

Time when the options or changes to the options become effective

After the FTP client has started up you can read in the required option file using the client command *readopt* (see page 179). You can repeat this procedure as often as you wish.

The individual options or changes to the options become effective at different times:

- With the following options the changes take effect the next time a control connection is set up:

  -protect
  -private
  -tlsCipherSuite
  -tlsKeyFile
  -tlsCertificateFile
  -tlsProtocol
  -tlsCACertificateFile
  -tlsCARevocationFile
  -tlsVerifyServer
  -tlsVerifyDepth
  -tlsRandomSeed
  -tlsUseCryptoHardware

- The following option is only evaluated once during an FTP session, namely when the OpenSSL library is loaded
  (Message: `[Loading OpenSSL library...]`):

  -tlsOpenSSLlibName

  If you wish to set up a connection to other values of these options after TLS has been initialized, proceed as follows:

  1. Terminate FTP client
  2. Adapt default option file if required
  3. Restart FTP client
  4. Before starting TLS initialization with a *protect*, *private* or *open* command:

     If required, read in a suitable option file using the *readopt* command

  Steps 2.) and 4.) are alternatives.

Description of the options

The individual options are described below.

-transferType - Select transfer type

The *-transferType* option is used to (pre)select a transfer type (see also the command “type - Change or query transfer type” on page 204).

In contrast to the *type* command, the *-transferType* option is only effective locally and only becomes active when a new connection is set up. If, however, an FTP connection already exists and the option file is read in with the *readopt* command, this connection remains unaffected by the *-transferType* option.

-transferType
ascii | **binary** | **ebcdic** | **tenex**

ascii
: Selects the transfer type ASCII. This transfer type should always be selected for text files.
ascii is the default.

binary
: Selects the transfer type BINARY. This transfer type should always be selected for binary files.

ebcdic
: Selects the transfer type EBCDIC. This transfer type should always be selected if both partners are working with EBCDIC, in other words code conversion is not required and a SAM file is to be generated in BS2000.

tenex
: Corresponds to the transfer type BINARY.

-initialCommand - Specify FTP client command

The *-initialCommand* option is used to specify an FTP client command which is executed immediately after the FTP client has been started up. Only commands should be used which do not require a connection to an FTP server to execute successfully. The *-initialCommand* option can be specified more than once to specify more than one FTP client command.

The specified command is only executed after both option files (central and local) have been read in. If the option files contain several *-initialCommand* options, the order in which the commands are executed is defined by the order in which the associated options are entered in the option files.

-initialCommand
<ftp-client-cmd>

<ftp-client-cmd>
: FTP client command which is to be executed immediately after the FTP client has started up.

-protect - TLS security for control connections

The *-protect* option defines that all control connections subsequently set up are secured using TLS. If no *-protect* option is specified, no TLS security is provided.

The *-protect* option has no influence on an existing control connection.

-protect

-private - TLS security for control and data connections

The *-private* option defines that all control and data connections subsequently set up are secured with TLS. The *-private* option has no influence on an existing control connection. However, if the existing control connection is secured by TLS, all data connections subsequently set up are also secured by TLS.

-private

-tlsRandomSeed - Initialize pseudo random number generator

The *-tlsRandomSeed* option is used to specify how the pseudo random number generator used by TLS is initialized. Efficient initialization with values that are as random and unforeseeable as possible is of decisive importance for TLS security. If the BS2000 subsystem PRNGD (**P**seudo **R**andom **N**umber **G**enerator **D**emon) is active on the system on which the FTP client is running, PRNGD is used for initialization, so the setting of the *-tlsRandomSeed* option is of practically no significance. Subsystem PRNGD is described in the “interNet Services Administrator Guide”.

-tlsRandomSeed
PROGRAM | **USER**

PROGRAM
Program-internal functions are used. These utilize above all fluctuations of the real-time clock in relation to the timer of the system CPU to generate random numbers for initialization.

USER
Part of the initialization is the same as with the specification PROGRAM. In addition, the user is repeatedly prompted to key in characters as randomly as possible and/or to press the ENTER key. The time stamp from the input is used for initialization. The characters entered are also used for initialization. Since, however, the randomness of the characters is not known and these characters can in many cases be intercepted, they are not taken into account when estimating whether there is already enough initialization material. Only the number of times the ENTER key is pressed is taken into account in this estimate.
USER is the default.

If the FTP client is operated in batch mode, it is generally recommendable to use the PROGRAM setting as no user is available to enter random numbers in batch mode.

-tlsProtocol - TLS/SSL protocol selection

OpenSSL supports Versions 3 of the SSL protocol and also Versions 1, 1.1 and 1.2 of the TLS protocol. Some of these protocols can be activated selectively using the *-tlsProtocol* option.

-tlsProtocol
[**+** | **-**] {**SSLv3** | **TLSv1** | **TLSv1.1** | **TLSv1.2** | **All** } ...

+
The protocol specified after this sign is permissible.

If neither “+” nor “-” is specified, this has the same effect as specifying “+”.

-
The protocol specified after this sign is not permissible.

SSLv3
SSL protocol Version 3

Version 3 of the SSL protocol displays some security-related deficiencies and should therefore not be used if possible.

TLSv1
TLS protocol Version 1

TLSv1.1
TLS protocol Version 1.1

TLSv1.2
TLS protocol Version 1.2

ALL
All protocols are to be enabled.

All -SSLv3 is the default.

Example

The specifications `-tlsProtocol TLSv1 TLSv1.1 TLSv1.2` and `-tlsProtocol All -SSLv3` have the same effect as long as no support of the future TLS version 1.3 is added to the FTP.

-tlsCipherSuite - Specify a cipher suite preference list

The *-tlsCipherSuite* option is used to specify a cipher suite preference list. If this option is not specified, a default preference list is used.

-tlsCipherSuite
<specification>

<specification>
Specification of a a cipher suite preference list, see section “Specification of a cipher suite preference list” on page 60.
ALL: !EXP: !ADH:!RC4 is the default.

-tlsCertificateFile - File with X.509 client certificate in PEM format

The *-tlsCertificateFile* option is used to specify a file which contains the X.509 client certificate for client authentication in PEM format. This file can also contain the client's private key. However, generally the certificate and key are stored in different files. In this case the key file is specified using the *-tlsKeyFile* option (see page 106).

-tlsCertificateFile
<file-name 1..54> | ***NONE**

<file-name 1..54>
Name of the file which contains the X.509 client certificate in PEM format.

***NONE**
No client certificate is used (and thus no client authentication, either).
*NONE is the default.

-tlsKeyFile - Specify file with client key in PEM format

The *-tlsKeyFile* option is used to specify a file which contains the private client key in PEM format.

If both an X.509 client certificate and a private client key are contained in the same file (see the *-tlsCertificateFile* option on page 105), the *-tlsKeyFile* option need not be specified.

If the client key is protected with a passphrase, this passphrase must be entered after the FTP client has started up the first time a TLS-secured FTP connection is established.

-tlsKeyFile
<file-name 1..54> | ***NONE**

<file-name 1..54>
Name of the file which contains the private client key.

***NONE**
No separate client key file is used.

The default is the file name specified in the *-tlsCertificateFile* option (see page 105).

-tlsCACertificateFile - Specify file with server authentication

The *-tlsCACertificateFile* option is used to specify a file containing the CA certificates in PEM format which are required for FTP server authentication. The individual PEM certificates are arranged sequentially in the file.

You can process the file with a text editor of your choice when you wish to add or delete certificates. The individual certificates are registered in the file as follows:

```
-----BEGIN CERTIFICATE-----
< CA certificate in Base64 encoding >
-----END CERTIFICATE-----
```

Text outside these sequences is ignored by the FTP client and can therefore be used to identify the certificates which, owing to the ASN.1/Base64 encoding, are available in non-readable form.

-tlsCACertificateFile
<file-name 1..54> | ***NONE**

<file-name 1..54>
: Name of the file containing the certificates in PEM format which are required for FTP server authentication.

***NONE**
: No file with CA certificates is specified.
*NONE is the default.

-tlsCARevocationFile - Specify file with CRL

The *-tlsCARevocationFile* option is used to specify a file which contains the CRLs (Certificate Revocation Lists) of the Certificate Authorities (CAs). (Certificates issued by a Certificate Authority can be declared invalid by publication of a Certificate Revocation List (CRL).)

-tlsCARevocationFile
<file-name 1..54> | ***NONE**

<file-name 1..54>
Name of the file which contains the CRLs of the Certificate Authorities.

***NONE**
No file with CRLs is specified.
*NONE is the default.

-tlsVerifyServer - Verify FTP server certificate (yes/no)

The *-tlsVerifyServer* option defines whether the FTP server certificate should be verified.

-tlsVerifyServer
YES | **NO**

YES
The FTP server certificate should be verified.
YES is the default.

NO
The FTP server certificate should not be verified.
This setting makes you vulnerable to “man in the middle” attacks.

-tlsVerifyDepth - Define verification depth

The *-tlsVerifyDepth* option is used to define the verification depth, in other words the maximum permissible number of certificates between the FTP server certificate and the certificate which is known to the FTP client.

Here you must note the following:

- If the value 1 (default) is specified as the maximum depth, the server certificate must have been signed directly by a Certificate Authority (CA) that the FTP client knows for it to be accepted.
- If the maximum depth is exceeded, the connection is cleared, unless verification of the FTP server certificate has been disabled with *-tlsVerifyServer NO* (see page 109).
- Specifying the depth as 0 is meaningless. In this case only self-signed certificates would be permissible.

-tlsVerifyDepth
<depth>

<depth>
Maximum permissible number of certificates between the FTP server certificate and the certificate which is known to the FTP client (including the FTP server certificate).
Default: 1

-tlsOpenSSLlibName - Define an LMS file for an OpenSSL library

The *-tlsOpenSSLlibName* option is used to define the LMS file from which the OpenSSL library should be dynamically loaded. It may be necessary to specify a name other than the default name if, for example, the OpenSSL library is also used by other products.

Dynamic loading of the OpenSSL library can be expedited with the aid of DAB using caches. If the OpenSSL library is used jointly by a number of products, the size of the DAB buffer used is reduced.

-tlsOpenSSLlibName
<openssl-libname>

<openssl-libname>
Name of the LMS file from which the OpenSSL library is to be dynamically loaded.
Default: $.SYSLNK.TCP-IP-AP.*nnn*

4.8 Commands for TLS/SSL support

A number of FTP client commands are of significance for TLS support. The table below lists these commands and includes a brief description of their functionality. Detailed descriptions of these commands can be found in the section “Overview of commands (FTP client)” on page 114.

Command	Functionality	Page
open	With enabled TLS security for the control connection: Open TLS-secured connection to a remote system.	165
private	Enable/disable TLS security for the data connection.	169
protect	Enable/disable TLS security for the control connection.	171
readopt	Read in option file.	179
status	Show whether – the data connections are secured with TLS, – a control connection exists, whether this is encrypted and, if appropriate, which algorithm is used for encryption.	196

FTP client commands in the context of TLS support

The settings for TLS/SSL security made via FTP client command have priority over those specified in the options.

4.9 Tape processing with FTP

The following formats, settings and modes are available in tape processing with FTP:

- all file formats: SAM-V, SAM-F, SAM-U, PAM
- all settings for BLKCTRL, BLKSIZE, RECSIZE
- all transfer modes: ASCII, EBCDIC, BINARY, ftyp text, textbin, binary

Tape processing is controlled by a BS2000 FILE command (see the “User Commands (SDF Format)” manual.

For a remote tape file you must prefix the FILE command you issue with *quote site ...* .

Examples

1. A SAM file is to be created on the MTC SADW48 of type T-C2. The tape already contains a file (-> FSEQ=2, as the required file is the second on the tape):

   ```
   FILE <FILE>,FCBTYP=SAM,RECFORM=F,RECSIZE=300,FSEQ=2,VOL=SADW48,DEV=T-C2
   ```

2. The file is to be created on the two empty MTCs SADW48,SADW49:

   ```
   FILE <FILE>,FCBTYP=SAM,RECFORM=FIX,RECSIZE=300,VOL=(SADW48,SADW49),DEV=T-C2
   ```

4.10 Overview of commands (FTP client)

FTP commands may be abbreviated up to the point required for unique identification. Operands are separated by spaces.

If file names occur as operands in commands, they may be specified in full (full path name). If only a relative file name is specified, the file name is extended to include the currently set working directory.

The following section describes the commands in alphabetical order.

append - Append a local file to a remote file

The *append* command transfers a file from a local host to a remote host and appends it to a file on the remote host that may already exist. In the case of DMS files of the file type SAM, the processing type can be influenced with the *ftyp* command.

append
<local-file> [<remote-file>]

<local-file>
: Name of a POSIX or DMS file on the local host that is to be transferred to the remote host. Metacharacters are not allowed.

<remote-file>
: Name of a file on the remote host (metacharacters are not allowed). If the file already exists, the local file is appended to it. If the file does not exist, a new file is created. If the *remote-file* operand is omitted, the name of the local file is used (in this case the name of the local file must correspond to the file naming conventions of the remote host). Uppercase letters in local file names are converted into lowercase. If the remote host is a BS2000 system, the server converts lowercase characters back into uppercase (see also the FTP client command *setcase* on page 192).

Example

The file *:5:$TCPTEST.MAN.FTP.C* is transferred from the local host to the remote BS2000 host, where it is appended to the file *:110:$TSOS.FTP.2*. The FTP server function PORT is called implicitly.

```
lpwd
Local directory is :5:$TCPTEST.MAN.

pwd
257 ":110:$TSOS." is current directory.

append ftp.c ftp.2
200 PORT command successful.
...
```

ascii - Set transfer type to ASCII

The *ascii* command sets the transfer type to ASCII. This type of transfer should always be chosen when transferring text files. On starting FTP, ASCII is set as the default value for the transfer type.

The transfer type can also be changed by using the *binary*, *tenex* and *type* commands. The currently set transfer type can be determined with the *status* and *type* commands. The transfer type is described in more detail on page 89.

ascii

Example

1. The current transfer type is queried.

```
type
Using binary mode to transfer files.
```

2. The transfer type is changed from BINARY to ASCII.

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

3. The current status is displayed

```
status
Connected to anlaged, port 21.
No proxy connection.
Passive Mode: off
Mode: stream; Type: ascii; Form: non-print; Structure: file
Copymode: off; Ftyp: textbin
Verbose: on; Bell: off; Prompting: on; Globbing: on
Filesystem is: BS2000
The character to change the filesystem is: %
ISO-codetable is ISO88591, EBCDIC-codetable is EDF041
Time limit for server responses: 30 secs
Time limit for file size determination: 60 secs
Store unique: off; Receive unique: off
Use EOF marker: special EOF marker (C-DATEIENDE):
Pad empty record with blank: off
Case sensitivity: OFF
Used job variable: *NONE
Used SDF-P variable: *NONE
Used receive selector: *NONE
```

```
Used send selector: *NONE
Hash mark printing: off; Use of PORT cmds: on
Protected control channel: off
Private data channel: off
Cipher: clear
```

bell - Enable/disable bell

The *bell* command has no effect in the BS2000 implementation and is only included for compatibility reasons.

bell

binary - Set transfer type to BINARY

The *binary* command sets the transfer type to BINARY. This type of transfer should always be chosen when transferring binary files. On starting FTP, ASCII is set as the default value for the transfer type.

The transfer type can also be changed by using the *binary*, *tenex* and *type* commands. The currently set transfer type can be determined with the *status* and *type* commands. The transfer type is described in more detail on page 89.

binary

Example

1. The current transfer type is queried.

```
type
Using ascii mode to transfer files.
```

2. The transfer type is changed from ASCII to BINARY.

```
binary
200 Type set to I.
```

3. A PAM file is transferred. The FTP server function PORT is called implicitly.

```
put pamela
200 PORT command successful.
...
```

4. The new transfer type is queried.

```
type
Using binary mode to transfer files.
```

bye - Exit FTP

The *bye* command terminates the FTP program. If a connection to a remote host is still open, it is cleared (implicit *close*).
The *quit* command can be entered as a synonym for *bye*.

bye

Example

```
bye
221 Goodbye.
```

ccc - Disabling TLS security for the control connection

The *ccc* command disables TLS security for the control connection. This is used mainly for FTP connections which are routed via firewalls and NAT devices as these devices must be able to read and modify certain FTP commands (PORT, PASV, EPRT, EPSV) to perform their tasks.

ccc

The *ccc* command enables you to cancel encryption of the control connection after the first time the user ID, password and, if necessary, account are transferred in order, for example, to subsequently transfer files through firewalls.

After the *ccc* command has been issued, the commands for setting the encryption of the data connections (PBSZ, PROT) are rejected, i.e. in this respect the status at the time when the *ccc* command was issued is retained. If, for instance, encryption of the data connections was enabled, the data connections continue to be encrypted.

You should use the *ccc* command only when it is absolutely necessary, because after it has run the names of the files and directories which were addressed and which could present secret information are once more visible to an attacker. A control connection in plaintext mode also means that certain attacks on an FTP server which are prevented by using TLS are possible again.

cd - Change remote working directory

The *cd* command changes the current working directory on the remote host. If a connection has been set up to a BS2000 FTP server, the *cd* command can be used to switch between the DMS and POSIX file system if a POSIX file system is available on the remote host (see page 85). The current remote working directory can be determined using the *pwd* command. Changing the file system always takes you to the HOME directory.

cd
.|**..**| <remote-directory> | **%BS2000** | **$HOME** | **%POSIX**

.
The remote directory is not changed (only provided for reasons of compatibility with other implementations).

..
The last partial qualifier (on the extreme right) is removed from the name of the remote working directory in the DMS file system.
In the POSIX file system, the directory is changed to the next higher level.

<>
Name of the new working directory.

%BS2000
Change to the DMS HOME directory. If you are already in the DMS file system, the %BS2000 specification is synonymous with $HOME.

$HOME
Change to the DMS HOME directory. In order to specify $HOME, you must already be in a DMS directory.

%POSIX
Change to the POSIX HOME directory.

Example

The remote host is a BS2000 host with a POSIX directory.

1. Query the remote working directory.

```
pwd
257 "/home/usr/tcptest" is current directory.
```

2. Change the working directory by adding the suffix *man/sam/nach.bs2000*.

```
cd man/sam/nach.bs2000
250 "/home/usr/tcptest/man/sam/nach.bs2000" is current directory now.
```

3. Query the new working directory.

```
pwd
257 "/home/usr/tcptest/man/sam/nach.bs2000" is current directory.
```

4. Change to the DMS directory.

```
cd %BS2000
250 :4:$TEST is current directory now.
```

cdup - Change to next higher directory (one level up)

The *cdup* command changes the current working directory to the next higher directory on the remote host. This can alternatively also be achieved by specifying the *cd* .. command.

cdup

close - Close connection to remote host

The *close* command clears the connection to the remote host.

close

Example

1. The status query shows that a connection to system *anlaged* exists.

```
status
Connected to anlaged, port 21.
No proxy connection.
Passive Mode: off
Mode: stream; Type: ascii; Form: non-print; Structure: file
Copymode: off; Ftyp: textbin
Verbose: on; Bell: off; Prompting: on; Globbing: on
Filesystem is: BS2000
The character to change the filesystem is: %
ISO-codetable is ISO88591, EBCDIC-codetable is EDF041
Time limit for server responses: 30 secs
Time limit for file size determination: 60 secs
Store unique: off; Receive unique: off
Use EOF marker: special EOF marker (C-DATEIENDE)
Pad empty record with blank: off
Case sensitivity: off
Used job variable: *NONE
Used SDF-P variable: *NONE
Used receive selector: *NONE
Used send selector: *NONE
Hash mark printing: off; Use of PORT cmds: on
Protected control channel: off
Private data channel: off
Cipher: clear
```

2. The *close* command clears the connection.

```
close
221 Goodbye.
```

copymode - Enable/disable 1:1 transfer of BS2000 disk files

With the *copymode* command you can enable and disable 1:1 transfer of BS2000 disk files. Further details on 1:1 transfer can be found in the section “1:1 transfer of BS2000 disk files” on page 80.

copymode
[**on** | **same** | **off**]

on
Enables 1:1 transfer (without retaining the file protection attributes). Attributes that retain content and structure, such as FCB type, block length, record length, record format, are taken over from the source file.
If no operand is specified, this corresponds to the specification `copymode on`.

same
Enables 1:1 transfer (while retaining the file protection attributes). Attributes that retain content and structure, such as FCB type, block length, record length, record format, are taken over from the source file.
With the exception of passwords and GUARDS rules, the file protection attributes (USER-ACCESS, ACCESS, BASIC-ACL, AUDIT, RETENTION-PERIOD), are also transferred to the target file.

off
Disables 1:1 transfer.
off is the default if no *copymode* statement is specified.

If you want to select *copymode* on the FTP server, specify *quote site cmod*. The parameters correspond to those of *copymode*. However, with *quote site cmod* one of the parameters on, same or off must always be specified.

Example

1. Transfer between two BS2000 systems, retaining the file protection attributes:
   - Enter `copymode same` on the FTP client
   - Send `quote site cmod same` to the FTP server
   - Start transfer of the required file

2. FTP client is on a Unix system.
   Transfer of a BS2000 file to a Unix system with subsequent transfer of the file to a BS2000 system.

   - Send `quote site cmod` ... to the BS2000 system
   - Transfer BS2000 file to the Unix system
   - Send `quote site cmod` ... to the BS2000 target system
   - Send BS2000 file to the BS2000 target system

     `type binary` must be selected so that the target system does not modify the received files.

debug - Enable/disable DEBUG output

DEBUG output is primarily used by network administrators and support staff to diagnose problems on the network. Users do not usually require DEBUG output.

debug
<debug-value>

<debug-value>
Values in the range 0 through 9 are permitted.

0 No DEBUG output

1 All messages from the FTP client to the FTP server and from the FTP server to the FTP client are output.

In addition to 1:

2 Information on file access is output.

If no operand is specified, the DEBUG output switch is toggled, i.e. if the DEBUG output was enabled, it is now disabled, and if it was disabled, it is now enabled. Values greater than 2 are treated in the same way as 2.

delete - Delete a remote file

The *delete* command is used to delete a file on the remote host. Partially qualified files on the remote host can be deleted with the *mdelete* command.

delete
<remote-file>

<remote-file>
Name of the file to be deleted on the remote host.

Example

1. The remote working directory is displayed.

```
pwd
257 ":110:$TSOS." is current directory.
```

2. The file *anton* is deleted.

```
delete anton
200 DELE command okay.
```

dir - Show information on remote file(s)

The *dir* command returns information about files on the remote host. Details on remote files can also be obtained with the *mdir*, *ls* and *mls* commands.

dir
[<remote-file>] [<local-file>]]

<remote-file>
Name of a file on the remote host. If this operand is omitted, a list of all files in the current working directory of the remote host is returned.

<local-file>
Name of a local file to which the output from the command is to be written. If this operand is not specified, the output is sent to the terminal.

If the connection to the partner was set up via an FTAC profile, the file information is only output relatively to the path name set in this profile.

Example

The remote host is a Unix system.

1. The local and remote working directories are queried.

```
lpwd
Local directory is :5:$TCPTEST.MAN.

pwd
257 "/usr/tcptest/man/sam/nach.bs2000" is current directory.
```

2. The files in the remote working directory are queried. The FTP server function PORT is called implicitly.

```
dir
200 PORT command successful.
...
-rw-rw-r-- 1 tcptest 229   Sep 7 12:55 anton.1
-rw-rw-r-- 1 tcptest 229   Sep 7 12:55 anton.2
-rw-rw-r-- 1 tcptest 229   Sep 7 12:55 anton.3
-rwx------ 1 tcptest 10505 Sep 5 18:35 berta
-rw-rw-r-- 1 tcptest 229   Sep 7 12:55 berta.1
-rw-rw-r-- 1 tcptest 229   Sep 7 12:55 berta.2
-rwx------ 1 tcptest 10505 Sep 5 18:19 caesar
```

```
-rw-rw-r-- 1 tcptest 229 Sep 7 12:55 zwaton
...
461 bytes received in 2.54 seconds (1.58E-01 Kbytes/s)
```

3. Information on the remote files *anton.** is written to the local file *anton.prot*.

```
dir anton.* anton.prot
200 PORT command okay.
...
```

exit - Define parameters for local exit routine

You can set exit routines for the FTP client with the *exit* command.

You will find further details on the FTP exit routines in the “interNet Services Administrator Guide”.

exit
[receive:<receive-selector>] [!] [send:<send-selector>]

<receive-selector>

<receive-selector> allows you to select the required action from several actions available when receiving data. For this purpose, you supply the exit routine with a string to which <receive-selector> points. You can decide freely the meanings that are to be attached to the individual strings. Only the string *NONE" has a fixed meaning: If you specify "*NONE", the exit routine is disabled.

If you do not specify <receive-selector>, no exit routine is selected.

<send-selector>

<send-selector> allows you to select the required action from several actions available when sending data. For this purpose, you supply the exit routine with a string to which <send-selector> points. You can decide freely the meanings that are to be attached to the individual strings. Only the string *NONE" has a fixed meaning: If you specify "*NONE", the exit routine is disabled.

If you do not specify <send-selector>, no exit routine is selected.

file - Define file attributes on local host

The *file* command specifies - for a non-1:1 transfer - the file attributes of a file to be transferred from the local host. *file* is mapped to the BS2000 *FILE* command. If preceded by the *quote site* command, this command can also be issued at the FTP server (see page 73).

file
<local-file> <, file-operand-list>

<local-file>
Name of the local file whose attributes are to be specified. Either a fully-qualified file name permitted in the *FILE* command or the "*" character must be used.

<file-operand-list>
The possible operands are listed in the appropriate BS2000 manuals.
If a file name is specified for *local-file*, the *file* command applies only to the next *get*, *mget*, *reget* or *recv* command that references the file name.
If "*" is specified for *local-file*, the *file* command applies only to the next *get*, *recv*, *reget* or *mget* command. The file attributes then apply to the local file name specified in the *get*, *reget*, *recv* or *mget* command.

If several *file* commands are entered, only the entries for the last *file* command apply.

If no *file* command is specified and if the file does not exist, the usual defaults apply, i.e.:

- For transfer type *ascii* (see the *type* or *ascii* command), SAM files with a variable record length are generated.
- For transfer type *binary* (see the *type* or *binary* command), PAM files are generated.
- If the file already exists, the attributes are taken from the catalog entry, and the file is overwritten.
- The *LINK* operand must not be included in the *file-operand-list*, since a link name is added automatically.

Restrictions

Only those file attributes also supported by C-RTS V2.0 for STREAM I/O (see the "C library functions for POSIX applications" manual) are supported here.

For ISAM files, only the settings *KEYPOS=5* and *KEYLEN=8* are supported for the key position and key length, respectively.

Record keys are not transmitted when ISAM files are transferred with the *put* command. The *recform* operand is ignored. Consequently, ISAM files can only be read.

How you transfer BS2000 disk files and retain their attributes is described in the section "1:1 transfer of BS2000 disk files" on page 80.

Example

1. The transfer type is set to *binary*.

```
type binary
200 Type set to I.
```

2. A *file* command is issued for *file1*.

```
file file1,fcbtype=sam
```

3. The file *file1* is created as a SAM file.

```
get file2 file1
```

form - Change or query transfer format

The *form* command sets the transfer format or returns details on the current transfer format. The currently set transfer format can also be determined with the *status* command.

form
[file]

file

Sets the transfer format to FILE.
If no operand is specified, details on the current transfer format are output on the screen.

The FTP standard provides for a number of different, optional transfer formats. Of these only the transfer format FILE is implemented in BS2000. The *form* command is included here only for compatibility with other implementations of FTP.

ftyp - Specify processing type for files on local host

The *ftyp* command defines whether SAM files on the local host are to be processed as text or binary files.

The *ftyp* command applies only to SAM files of the DMS file system. PAM files are still treated as binary files, and ISAM files as text files. POSIX files are not affected by this restriction.

If preceded by the *quote site* command, this command can also be issued at the FTP server (see page 73).

ftyp
text | **binary** | **textbin**

The default has changed since previous FTP versions, because the transfer cannot be restarted with ftyp=text (for further details, see the chapter “Frequently asked questions (FAQ)” on page 485).

text

SAM files on the local host are to be processed as text files.
With this setting, tabs (X’05’ in EBCDIC) are converted to a corresponding number of blanks when writing to a file (tab stops 1, 9, 17 ...), and newline characters (X’15’ in EBCDIC) cause a line feed (record change).
When reading, newline characters (X’15’) are added to each record read (see the “C library functions for POSIX applications” manual).

binary

SAM files on the local host are to be processed as binary files.
With this setting, data is transferred unchanged when writing to a file. No conversion of tabs (X’05’) and newline characters (X’15’) occurs. When reading, no newline characters (X’15’) are added.
This setting enables SAM files of fixed, variable and undefined record lengths to be written and read.

For records of fixed length, the last record is padded with binary zeros (if necessary). If this is not desirable, variable length records should be used.

textbin

SAM files on the local host are to be processed as binary text files.
With this setting, tabs (X’05’ in EBCDIC) are not converted to blanks when writing to a file, but newline characters (X’15’ in EBCDIC) do result in a line feed (record change).
When reading, a new line character (X’15’) is added to each record read.
This setting is defined after FTP is started.

Example

1. The file processing type is set to *binary*.

```
ftyp binary
ftyp set to binary.
```

2. The file *file1* is created without conversion of the tab and newline characters.

```
get file2 file1
```

get - Fetch a file

The *get* command is used to transfer a file from the remote host to the local host. Files may also be transferred from the remote host to the local host with the commands *mget* and *recv*. A special variant of the *get* command, the *reget* command (see page 181), is now available for supporting restart functionality.

get
<remote-file> [<local-file>]

<remote-file>
Name of a file on the remote host to be transferred to the local host.

<local-file>
Name of a POSIX or DMS file on the local host. If the file already exists, it is overwritten or, alternatively, a new file is created after adding an appropriate suffix. This behavior is controlled by *runique*. If the file does not exist, a new file is created.
If the *local-file* operand is omitted, the name of the remote file is used (in this case, the name of the remote file must correspond to the file naming conventions on the local host).

Example

The remote host is a Unix host.

1. The names of the local and remote working directories are queried.

```
lpwd
Local directory is :5:$TCPTEST.MAN.VON.UNIX.

pwd
257 /home/usr/man is current directory.
```

2. The contents of the file *anton* (in the remote working directory) are transferred to a DMS file of the same name (in the local working directory). The FTP server function PORT is called implicitly.

```
get anton
200 PORT command okay.
...
```

3. The directory is changed to the POSIX HOME directory.

```
lcd %POSIX
Local directory now /home/usr
```

4. The remote file *anton* is transferred to the local host and stored in the POSIX file system under the name */home/usr/berta*.

```
get anton berta
200 PORT command okay.
...
```

glob - Enable/disable expansion of metacharacters

The *glob* command enables or disables the expansion of metacharacters in local file names. On starting FTP, metacharacter expansion is enabled by default.

In the commands *mget*, *mput*, *mdir*, *mls* and *mdelete*, metacharacters may be specified in the local file names. If metacharacter expansion is enabled, the metacharacters are recognized as such and evaluated in accordance with the rules. If metacharacter expansion is disabled, the metacharacters are not recognized as such and are interpreted as part of the file name.

glob

Example

```
mget a*
```

If metacharacter expansion is disabled - as can be seen from the status *off* - FTP interprets the entry *a** as a fully-qualified file name and issues the error message:

```
550 a* not a plain file or does not exist
```

hash - Enable/disable transfer progress indicator

The *hash* command enables or disables the indicator that shows the progress of a file transfer. If the indicator is enabled, then after the transfer of each block, the # (hash) character is output on the screen.

On starting FTP, *hash* is disabled. When *hash* is enabled, the block length being used is output. If FTP is called in batch mode, i.e. if task switch 1 is set, then *hash* has no effect.

The current setting of *hash* can be determined with the *status* command.

hash

Example

1. Query the names of local and remote working directories.

```
lpwd
Local directory is :5:$TCPTEST.MAN.

pwd
257 "/usr/tcptest/man/sam/from.bs2000" is current directory.
```

2. Enable the indicator. The FTP server function PORT is called implicitly with the *append* command.

```
hash
Hash mark printing on (8192 bytes/hash mark).

append sam.to.sinix.anton.upd anton
200 PORT command okay.
```

3. Transfer of the file, with output of the # character after each block of 8192 bytes.

```
##################################################
226 Transfer complete.
109089 bytes sent in 12.83 seconds (8.30 Kbytes/s)
```

help - Show information on FTP commands

The *help* command displays a list of all FTP commands. Brief information is also available on each of the commands.

The *?* command also shows a list of all FTP commands. A list of all the functions of the FTP server on the remote host can be obtained by using the *remotehelp* command.

help
[<command>]

<command>
FTP command on which information is to be displayed.
If no FTP command is specified, a list of all FTP commands is shown.

Example

1. Request a list of all FTP commands.

```
help
Commands may be abbreviated.  Commands are:
!               modchar         mget            quote           set
?               file            mkdir           recv            status
append          form            mls             readopt         struct
ascii           ftyp            mlsd            reget           sunique
bell            get             mlst            remotehelp      svar
binary          glob            mode            rename          system
bye             hash            mput            reput           tenex
ccc             help            open            rexit           trace
cd              jobvar          passive         rmdir           type
cdup            lcd             private         runique         user
copymode        ldir            prompt          send            verbose
close           lls             protect         sendport
delete          lpwd            proxy           setcase
debug           ls              put             setcode
dir             mdelete         pwd             setfile
exit            mdir            quit            settime
```

2. Request brief information on mget.

```
help mget
mget get multiple files
```

jobvar - Store error information in a job variable

The *jobvar* command can be used to inform the FTP client as to whether the command return information is to be stored in a job variable.

An alternative to the storage of error information is offered by the *svar* command (see page 200).

If an error occurs when processing the *jobvar* command and if task switch 1 is set (batch operation), the FTP client is terminated with TERMJ. The batch job or the procedure is therefore only continued after the next STEP instruction.

If the provision of a job variable is activated, the behavior of commands that execute several individual actions (commands *mdir*, *mls*, *mget*, *mput*, *mdelete*) changes. In the case of these commands, command processing is aborted after the first faulty action.

In contrast, the FTP client is not terminated with TERMJ in batch mode if the admission data in the *open* command is faulty.

The JV subsystem must be activated in order to use the *jobvar* command.

jobvar
< jv-name> | ***NONE**

<jv-name>
: Name of job variable that is to supply the FTP client with the command return information after the *jobvar* command is issued. If the job variable <jv-name> does not already exist, it is created.

***NONE**
: Specification of *NONE causes a job variable to be no longer supplied with command return information from the FTP client.

Layout of job variables

Offset / Length	Field	Field description
0 / 3	Status indicator	$S: Command executed successfully $E: Command has errors $T: FTP client ended normally $A: FTP client ended abnormally with errors
3 / 1	Filler	Always "0"
4 / 4	TSN	TSN of FTP client task
8 / 4	catid	Supplied with spaces
12 / 4	Session number	System sequence number
16 / 16	User command name	Name of FTP client command
32 / 96	Command Parameter	FTP command parameter
128 / 4	FTP protocol command	Command sent from the client to the server
132 /124	FTP message	Local message or server response

lcd - Change local working directory

The *lcd* command changes the current local working directory or the file system on the local host. Changing the file system always takes you to the corresponding HOME directory. The current local working directory can be determined with the *lpwd* command. The remote working directory can be set with the *cd* command.

lcd
. | **..** | **%POSIX** | **%BS2000** | **$HOME** | <path>

.
The local working directory is not changed (included only for compatibility with other implementations).

..
The last partial qualifier (on the extreme right) is removed from the name of the local working directory in the DMS file system.
In the POSIX file system, the directory is changed to the next higher level.

%POSIX
Changes to the local POSIX HOME directory.

%BS2000
Changes to the local DMS HOME directory. In the DMS file system, the %BS2000 specification is synonymous with $HOME.

$HOME
Changes to the local DMS HOME directory. In order to specify $HOME, you must already be in a DMS directory.

<path>
If a full path name (beginning with the specification of a catalog and/or user ID) is specified in the DMS directory, the existing working directory is replaced. If the specification of the catalog and the user ID are omitted, the <old.name>.<path> is set.

The path names are not validated in the DMS file system. In the POSIX file system, <path> may be an absolute or relative POSIX path name.

Example

1. Query the local working directory.

```
lpwd
Local directory is :5:$TCPTEST.
```

2. Change the working directory by adding the partial qualifier MAN.

```
lcd man
Local directory now :5:$TCPTEST.MAN.
```

3. Change to the POSIX HOME directory.

```
lcd %POSIX
Local directory now /home/usr
```

4. Change the working directory to */home/usr/test*.

```
lcd test
Local directory now /home/usr/test
```

5. Change to the DMS HOME directory.

```
lcd %BS2000
Local directory now :5:$TCPTEST.
```

6. Change the working directory by adding a partial qualifier.

```
lcd XXX
Local directory now :5:$TCPTEST.XXX.
```

7. Change the working directory by removing a partial qualifier.

```
lcd ..
Local directory is :5:$TCPTEST.
```

ldir - List information on local files

The *ldir* command returns information about files on the local host. Details on local files can also be obtained by using the *lls* command.

ldir
[<local-file>] [, <fstatus-operand-list>]

<local-file>

ldir is mapped to the BS2000 command *FSTATUS* in the DMS file system.
<*local-file*> must be a fully or partially qualified file name permitted in the *FSTATUS* command.
ldir is mapped to the *ls -l* command in the POSIX file system. <*local-file*> must be a legal fully or partially qualified file name.
If no local file is specified, information on all files in the current working directory is displayed.

<fstatus-operand-list>

fstatus-operand-list represents any other operands permitted in the *FSTATUS* command. The *FSTATUS* command is described in the “User Commands (ISP Format)” manual.

Example

1. Query the local working directory.

```
lpwd
Local directory is :5:$TCPTEST.MAN.
```

2. Display information on DMS files with the SAM file attribute in the working directory.

```
ldir ,fcbtype=sam
0000003 :5:$TCPTEST.MAN.ANTON.PROT
0000003 :5:$TCPTEST.MAN.P.TRC
0000003 :5:$TCPTEST.MAN.P.USR
0000003 :5:$TCPTEST.MAN.SAM.NACH.MSDOS.ANTON.UPD
0000003 :5:$TCPTEST.MAN.SAM.NACH.MSDOS.BERTA.1
0000003 :5:$TCPTEST.MAN.SAM.NACH.SINIX.ANTON
0000006 :5:$TCPTEST.MAN.SAM.VON.SINIX.BERTA
0000003 :5:$TCPTEST.MAN.SAM.VON.SINIX.BERTA.1
0000003 :5:$TCPTEST.MAN.SAM.VON.SINIX.BERTA.2
0000006 :5:$TCPTEST.MAN.SAM.VON.SINIX.CAESAR
:5: PUBLIC: 30 FILES. RES= 99, FREE= 51, REL= 0 PAGES
```

lls - list file names on local host

The *lls* command lists the file names on the local host.

lls
[<local-file>] [, <fstatus-operand-list>]

<local-file>
lls is mapped in the to the BS2000 command *FSTATUS* in the DMS file system.
<*local-file*> must be a fully or partially qualified file name that would be permitted in the *FSTATUS* command.
In the POSIX file system, *lls* is mapped to the *ls* command. <*local-file*> must be a fully or partially qualified file name.
If no local file is specified, information on all files in the current working directory is displayed.

<fstatus-operand-list>
fstatus-operand-list represents any other operands permitted in the *FSTATUS* command. The *FSTATUS* command is described in the “User Commands (ISP Format)” manual.

Example

1. Query the local working directory (DMS file system).

```
lpwd
Local directory is :5:$TCPTEST.
```

2. Request a list of the DMS files in the local directory.

```
lls sam.nach.*
SAM.NACH.MSDOS.ANTON.UPD
SAM.NACH.MSDOS.BERTA.1
SAM.NACH.MSDOS.BERTA.2
SAM.NACH.SINIX.ANTON
SAM.NACH.SINIX.BERTA.1
SAM.NACH.SINIX.BERTA.2
SAM.NACH.SINIX.CAESAR
```

lpwd - Show local working directory

The *lpwd* command returns the name of the current local working directory. The local working directory is set by means of the *lcd* command.

lpwd

Example

1. Query the local working directory.

```
lpwd
Local directory is :5:$TCPTEST.
```

2. Change the local working directory by adding the partial qualifier *MAN.SAM*.

```
lcd man.sam
Local directory now :5:$TCPTEST.MAN.SAM.

lpwd
Local directory is :5:$TCPTEST.MAN.SAM.
```

3. Change the working directory in the POSIX file system.

```
lcd %POSIX
local directory now /home/usr/tcp.

lpwd
local directory is /home/usr/tcp.
```

ls - list file names on remote host

The *ls* command lists file names on the remote host. Information on remote files can also be obtained by using the *dir*, *mdir* and *mls* commands.

ls
[<remote-file>] [<local-file>]]

<remote-file>
Name of a file on the remote host. If this operand is omitted, a list of all the files in the current working directory of the remote host is returned.

<local-file>
Name of the local file to which the output from the command is to be written. If this operand is not specified, the output is sent to the terminal.

Example

The remote host is a Unix host.

1. Query the name of the working directory.

```
pwd
257 "/usr/tcptest" is current directory.
```

2. List the files in the working directory on the screen. The FTP server function PORT is called implicitly.

```
ls
200 PORT command okay.
...
FIL.files
FIL.tools
FTP
TELNET
TU
man
pool
backup
...
```

3. List the remote files designated by *FIL.** and pipe the output to the local file *fil.prot*.

```
ls FIL.* fil.prot
200 PORT command okay.
...
```

mdelete - Delete multiple remote files

The *mdelete* command is used to delete files from the remote host. The prompting function is enabled and disabled using the *prompt* command. The *delete* command can also be used to delete files from the remote host.

mdelete
<remote-file> [<remote-file>] [<remote-file>]

<remote-file>
Name of the file to be deleted from the remote host. Multiple file names may be specified.
If the prompting function is enabled, FTP checks before deleting each file whether the file is really to be deleted.

Example

The remote host is a Unix host.

1. Query the name of the remote working directory.

```
pwd
257 "/usr/tcptest/man/sam/von.msdos" is current directory.
```

2. Delete the files *anton.** and *berta.**; the prompting function is enabled.

```
mdelete anton.* berta.*
mdelete anton.2 (y/n/q)?

y
200 DELE command okay.
mdelete anton.3 (y/n/q)?

n
mdelete anton.upd (y/n/q)?

y
200 DELE command okay.
mdelete berta.1 (y/n/q)?

y
200 DELE command okay.
mdelete berta.2 (y/n/q)?

n
```

3. Disable the prompting function.

```
prompt
Interactive mode off.
```

4. Delete the remaining files in the remote working directory; the prompting function is disabled.

```
mdelete *
200 DELE command okay.
```

mdir - Show information on remote files

The *mdir* command returns information on files on the remote host. Details on remote files can also be obtained by using the *dir*, *ls* and *mls* commands.

mdir
[<remote-file>] [<remote-file>] [<remote-file>] ... - | <local-file>

<remote-file>
Name of a file on the remote host. Multiple file names may be specified in a single call.

<local-file>
Name of the local file to which the output from the command is to be written.

-
The information is to be displayed on the screen.

If no operands are specified, they are requested interactively.

If the connection to the partner was set up via an FTAC profile, the file information is only output relatively to the path name set in this profile.

Example

The remote host is a Unix host.

1. Query the names of local and remote working directories.

```
lpwd
Local directory is :5:$TCPTEST.MAN.

pwd
257 "/usr/tcptest/man/sam/nach.bs2000" is current directory.
```

2. Output information on the remote files *berta*.* and *anton*.* to the local file *SEVERAL.PROT*. The FTP server function PORT is called implicitly.

```
mdir berta.* anton.* several.prot
200 PORT command okay.
150 ASCII data connection for berta.* (89.16.100.0,1187).
226 Transfer complete.
100 bytes received in 0.71 seconds (1.36E-01 Kbytes/s)
200 PORT command okay.
150 ASCII data connection for anton.* (89.16.100.0,1188).
226 Transfer complete.
150 bytes received in 1.64 seconds (8.90E-02 Kbytes/s)
```

mget - Fetch multiple remote files

The *mget* command is used to transfer files from the remote host to the local host. Files can also be transferred from the remote to the local host by using the *get*, *reget* and *recv* commands.

mget
<remote-file> [<remote-file>] [<remote-file>] ...

<remote-file>
: Name of the file on the remote host to be transferred to the local host. Multiple file names may be specified.
The files are assigned the same names on the local host as they have on the remote host. The file names must therefore comply with the rules for the local host as well as with those of the remote host.
If the prompting function is enabled (see *prompt*), FTP requests confirmation before transferring each file.

Example

The remote host is a Unix host.

1. Query names of local and remote working directories.

```
lpwd
Local directory is :5:$TCPTEST.MAN.SAM.VON.SINIX.

pwd
257 "/usr/tcptest/man/sam/nach.bs2000" is current directory.
```

2. All files from the remote working directory are transferred. The FTP server function PORT is called implicitly. The prompting function is disabled in this example (see *prompt*).

```
mget *
200 PORT command okay.
150 ASCII data connection for anton (89.16.100.0,1192).
226 Transfer complete.
242 bytes received in 0.06 seconds (3.69 Kbytes/s)
200 PORT command okay.
150 ASCII data connection for anton.1 (89.16.100.0,1193).
226 Transfer complete.
242 bytes received in 0.06 seconds (3.75 Kbytes/s).
.
further transfers
.
200 PORT command okay.
150 ASCII data connection for caesar (89.16.100.0,1199).
226 Transfer complete.
10845 bytes received in 0.78 seconds (13.56 Kbytes/s)
```

mkdir - Create a remote directory

The *mkdir* command is used to create a new directory on the remote host. This command is not valid for the DMS file system.

mkdir
<remote-directory>

<remote-directory>
Fully qualified name of the directory on the remote host.

mls - list file names in multiple directories on remote host

The *mls* command returns information on files on the remote host. Details on remote files can also be obtained with the *dir*, *mdir* and *ls* commands.

mls
[<remote-file>] [<remote-file>] [<remote-file>] ... - | <local-file>

<remote-file>
Name of a file on the remote host. Multiple file names may be specified in a single call.

<local-file>
Name of the local file to which the output from the command is to be written.

- The information is to be displayed on the screen.

If no operands are specified, they are requested in a dialog.

Example

The remote host is a Unix host.

1. The names of the local and remote working directories are queried.

```
lpwd
Local directory is :5:$TCPTEST.MAN.

pwd
257 "/usr/tcptest/man/sam/nach.bs2000" is current directory.
```

2. The files designated by *berta*, *berta.** and *anton.** are listed in the local file *lists.prot*. The FTP server function PORT is called implicitly. The prompting function is disabled.

```
mls berta berta.* anton.* lists.prot
200 PORT command okay.
150 ASCII data connection for berta (89.16.100.0,1184).
226 Transfer complete.
6 bytes received in 0.05 seconds (9.93E-02 Kbytes/s)
200 PORT command okay.
150 ASCII data connection for berta (89.16.100.0,1185).
226 Transfer complete.
16 bytes received in 0.21 seconds (7.37E-02 Kbytes/s)
200 PORT command okay.
150 ASCII data connection for anton.* (89.16.100.0,1186).
226 Transfer complete.
24 bytes received in 0.14 seconds (1.58E-01 Kbytes/s)
```

mlsd - list file names and file properties on the remote host

The *mlsd* command lists file names and properties on the remote host using the MLSD FTP server command, which returns data in a machine-readable format (see RFC 3659). It is therefore not aimed quite so much at human users (although it does provide these users with more information for DVS files than the *dir* command), but it is especially suited for use within procedures.

The scope of output can be modified using the OPTS FTP server command. For more information, see *mlst* command.

mlsd
[<remote-directory> [<local-file>]]

<remote-directory>

Name of a directory on the remote host. If this operand is omitted, a list of all the files in the current working directory of the remote host is returned.

<local-file>

Name of the local file to which the output from the command is to be written. If this operand is not specified, the output is sent to the terminal.

Example

The remote host is a BS2000 host.

1. Query the name of the working directory.

```
pwd
257 ":2OSG:$TSOS." is current directory.
```

2. List the files in the working directory on the screen. The FTP server function PORT is called implicitly.

```
mlsd
200 PORT command okay
. . .
type=file;perm=fr;size=2048;create=20161220175301;modify=20161220175301;
SYSENT.TCP-IP-AP.053.FTPD
type=file;perm=fr;size=2048;create=20161216132350;modify=20161216132350;
SYSENT.TCP-IP-AP.053.START
type=file;perm=fr;size=2048;create=20161216132414;modify=20161216132414;
SYSENT.TCP-IP-AP.053.STARTF
type=file;perm=fr;size=2048;create=20161216132352;modify=20161216132352;
SYSENT.TCP-IP-AP.053.STARTT
```

```
type=file;perm=fr;size=2048;create=20161216132405;modify=20161216132405;
SYSENT.TCP-IP-AP.053.TELNETD
. . .
```

mlst - list file properties on the remote host

The *mlst* command lists the file properties on the remote host using the MLST FTP server command, which returns data in a machine-readable format (see RFC 3659). It is therefore not aimed quite so much at human users, but it is especially suited for use within procedures. Unlike the *mlsd* command, information is not returned via a separate data connection, but via the control connection, because it only returns information on a single file in each case.

The scope of output can be modified using the OPTS FTP server command (for details, see RFC 3659): A *quote OPTS MLST;* means that *mlsd* and *mlst* list only the file name(s). After a *quote OPTS MLST modify;size;*, the size, date of change, and file name are listed.

mlst
[<remote-file>]

<remote-file>

Name of a file on the remote host.

Example

The remote host is a BS2000 host.

1. Query the name of the working directory.

```
pwd
257 ":2OSG:$TSOS." is current directory.
```

2. Query the file properties of a file.

```
mlst SYSENT.TCP-IP-AP.053.FTPD
250-Listing SYSENT.TCP-IP-AP.053.FTPD
type=file;perm=fr;size=2048;create=20161220175301;modify=20161220175301;
SYSENT.TCP-IP-AP.053.FTPD
250 End
```

3. Reduce the output scope.

```
quote OPTS MLST modify;size;
200 MLST OPTS size;modify;
```

4. Query the file properties again.

```
mlst SYSENT.TCP-IP-AP.053.FTPD
250-Listing SYSENT.TCP-IP-AP.053.FTPD
size=2048;modify=20161220175301; SYSENT.TCP-IP-AP.053.FTPD
250 End
```

modchar - Modify character string

In order to switch between the local DMS file system and the local POSIX file system with the *lcd* command, you need to specify a defined string (see page 145). If problems are encountered when using the standard strings %BS2000 and %POSIX (e.g. because a POSIX directory of the same name exists), the first character of the string can be modified with *modchar*.

The current string can be queried with the *status* command.

The relevant setting can be made on a BS2000 FTP server with the *quote site modc* command (see page 73).

modchar
<character>

<character>
: Character to be substituted for the first character in the strings to switch between the DMS and POSIX file systems.

Example

1. Query the local working directory.

```
lpwd
Local directory is :4:$TEST.
```

2. Switch from the DMS HOME directory to the POSIX HOME directory.

```
lcd %POSIX
Local directory now /home/usr.

lpwd
Local directory is /home/usr.
```

3. The first character in the currently set string to switch between the DMS and POSIX file systems is changed to #.

```
modchar #
The prefix has been set to #.
```

4. Switch from the POSIX HOME directory to the DMS HOME directory.

```
lcd #BS2000
Local directory now :4:$TEST.
lpwd
Local directory is :4:$TEST.
```

mode - Change or query transfer mode

The *mode* command defines the transfer mode or returns details on the currently set transfer mode. If no operand is specified, the current transfer mode is displayed on the screen. Details on the currently set transfer mode can also be obtained with the *status* command.

The FTP standard defines a number of optional transfer modes. In BS2000, only the *stream* and *block* transfer modes have been implemented.

mode
[**stream** | **block**]

stream
Sets the transfer mode to STREAM.

block
This operand enables files to be transferred without changing the existing record structure. In order to specify *block*, the following conditions must be satisfied:

– The transfer type must be EBCDIC or ASCII, and
– the file attribute FCBTYPE=SAM must be set.

mput - Send multiple local files

The *mput* command is used to transfer files from the local host to the remote host. Files can also be transferred to a remote host by using the *put*, *reput*, *append* and *send* commands.

mput
<local-file> [<local-file>] [<local-file>] ...

<local-file>
: Name of a file on the local host to be transferred to the remote host. Multiple files may be specified in a single call.
The files are assigned the same names on the remote host as on the local host. The file names must therefore comply with the file name conventions of the local host as well as those of the remote host. By default uppercase characters in the name of the local file are converted to lowercase. However, you can disable this setting using the client command *setcase* (see page 192).
If the prompting function is enabled (see the *prompt* command), FTP requests confirmation before transferring each file. If a file transfer is unsuccessful, the following files are also not transferred.

Example

The remote host is a Unix host.

1. Query the names of local and remote working directories.

```
lpwd
Local directory is :5:$TCPTEST.MAN.SAM.NACH.SINIX.

pwd
257 "/usr/tcptest/man/sam/nach.bs2000" is current directory.
```

2. Transfer the files in the local directory designated by the entries *anton*, *anton.** and *berta.**. The FTP server function PORT is called implicitly. The prompting function is disabled in this example. See the description of the *prompt* command on page 170.

```
mput anton anton.* berta.*

200 PORT command okay.
150 ASCII data connection for anton (89.16.100.0,1201).
226 Transfer complete.
242 bytes sent in 0.03 seconds (6.05 Kbytes/s)
200 PORT command okay.
150 Opening data connection for anton.upd (89.16.100.0,1202).
.
further transfers
```

```
.
226 Transfer complete.
242 bytes sent in 0.04 seconds (5.76 Kbytes/s)
200 PORT command okay.
150 ASCII data connection for berta.2 (89.16.100.0,1207).
226 Transfer complete.
242 bytes sent in 0.04 seconds (5.90 Kbytes/s)
```

open - Open the connection to a remote host

The *open* command sets up a connection to a remote host. Either the name or the internet address of this host must be known. The host must either be part of the local network or be accessible via a gateway. The names and addresses of the hosts that can be reached via the *open* command can be obtained from the administrator of the local network.

If TLS security for the control connection was enabled using the *protect* or *private* command, this results in the behavior of the *open* command being modified (see page 167).

open
<ipadr> | <remote-host> | local-host | loopback [<port>]

<ipadr>
: Internet address (IPv4 or IPv6 address) of the remote host to which the connection is to be made.

- An IPv4 address must be specified in the usual “decimal-dotted” notation.
- An IPv6 address must be specified in the usual hexadecimal notation with a colon (:).

<remote-host>
: Symbolic name of the remote host to which the connection is to be made.

local-host
: Symbolic name reserved for the user’s own host (usually meaningful only for test purposes). A separate Internet address is generated for *localhost* that is not identical to the address of the user’s own host in the local network. This address is known only to the local computer; other computers cannot use it.

loopback
: *loopback* stands for the Internet address by means of which the local computer can actually be accessed in the LAN.

<port>
: Port number of the FTP server. The FTP server is assigned port number 21 by default. The specification of a port number is required, for example, when multiple FTP servers are run on one host.

If an access control mechanism has been implemented on the remote host (for example, in BS2000 and in Unix systems), FTP determines the required authorization data (user ID, account number, password) interactively. Only the data that is really needed is requested (e.g. Unix systems do not require an account number, so the corresponding request is omitted).

If the authorization data is entered incorrectly, the connection to the remote host remains open when working in interactive mode. The *user* command can then be used to enter the appropriate data.

If task switch 1 is set, then the required authorization data must be entered in batch mode or interactive mode on separate lines (in the correct sequence) following the *open* command. If the authorization data is invalid, the connection is cleared. FTP is terminated with TERMJ, i.e. the batch job or the procedure is continued only after the next STEP instruction. See also the client commands *jobvar* on page 143 and *svar* on page 200.

The admission data must be specified in accordance with the conventions of the partner system. Please observe the table on page 82.

If no password is required, enter *'*NONE'* (uppercase letters).

Example

The remote host is a Linux host.

1. Call the FTP client.

```
/START-FTP
%  BLS0523 ELEMENT 'FTP', VERSION 'V05.3A00', TYPE 'L' FROM LIBRARY
':HPC3:$TSOS.SYSLNK.TCP-IP-AP.053' IN PROCESS
% BLS0524 LLM 'FTP', VERSION 'V05.3A00' OF '2016-12-05 19:29:27' LOADED
%  BLS0551 COPYRIGHT (C) 2016 FUJITSU TECHNOLOGY SOLUTIONS GMBH. ALL
RIGHTS RESERVED
BS2000-FTP Vers V05.3A00 Dec  5 2016 18:41:20
Some options read from global option file,
option file name: $.SYSDAT.TCP-IP-AP.053.F-GOPT
ftp>
```

2. Open a connection to the host systemd.

```
open systemd
Connected to anlaged, port21.
220 anlaged (vsFTPd 2.0.5)
```

3. Prompt for a user ID (name).

```
Name (systemd:TCPTEST):
USERID
331 Password required for tcptest.
```

4. Prompt for a password.

```
Password (systemd:tcptest):
PASSWORD
230 Login successful.
```

Behavior of the *open* command when TLS security for the control connection already exists

When TLS/SSL security for the control connection is enabled via the *protect* or *private* command or the corresponding options, the *open* command behaves as follows:

1. If the TLS subsystem has not yet been initialized the message `[Initialising TLS]` is issued and, if the BS2000 subsystem PRNGD is not yet active, you yourself must ensure the pseudo random number generator is initialized (see the section “protect - Enable/disable TLS security for the control connection” on page 171).
2. If a private key (for client certification) was specified in the option file, the passphrase for the private key is queried when the connection to the server is set up if this key was stored in encrypted form. For security reasons we strongly recommend that you store your private key in encrypted form.

   The private key is loaded only when the first TLS-secured *open* command is issued, which means that the passphrase need not be entered again in the event of further *open* commands in the same FTP session.
3. The message `[Starting SSL/TLS negotiation ...]` is issued. If no errors occur in this TLS negotiation, the FTP server certificate data is displayed:
   - name of the certificate owner
   - name of the certificate issuer
   - validity period
4. If the host name of the FTP server which is specified in the *open* command is not identical to the name entered in the certificate, a corresponding warning is issued and you are asked if you want to continue. You should only decide to continue if you are certain that the server designated by the certificate is really the required server. Otherwise you could become the victim of a so-called “man in the middle” attack: In a “man in the middle” attack the attacker places himself/herself between server and client and pretends to the client to be the required server.

   If the host name and certificate owner are the same or if you have given the go-ahead despite differences in names, the usual prompt is displayed for entering the user name for the FTP server: The further procedure is the same as for an *open* command without TLS security.

passive - Enable/disable PASSIVE mode

The *passive* command enables or disables PASSIVE mode. PASSIVE mode has now become the default setting in many implementations and is used to reach the internet from a local network via a firewall. Such firewalls often suppress active connection setup into the network protected by the firewall.

Details on PASSIVE mode are provided in the “interNetServices Administrator Guide”.

passive
[**on** | **off**]

on
PASSIVE mode is enabled.
If no operand is specified, this corresponds to the specification `passive on`.

off
PASSIVE mode is disabled.
off is the default if no passive statement is specified.

In PASSIVE mode all subsequent file transfer commands are handled with active client and passive child, in other words by means of PASV and EPSV commands. This applies until a negative acknowledgment is received by the server for a PASV or EPSV command, for example when a connection to the server has been terminated.

The following applies for the EPSV command on the child side: If you specify the ALL parameter in the EPSV command, all subsequent PORT and EPRT commands are rejected with “522 PORT command not successful” or “522 EPRT command not successful”.

The *status* command allows you to check whether PASSIVE mode is set:

```
ftp> status
Connected to PGAB0021, port 21.
No proxy connection.
Passive Mode: on
...
```

Each time you enter the *passive* command the latest setting is output.

private - Enable/disable TLS security for the data connection

The *private* command can be used to change the switch for TLS security for the data connection.

You can inquire the current setting for this option using the *status* command.

Data connections are only secured by TLS if the control connection is also secured by TLS. The *private* command is thus rejected if it is issued when there is a control connection that does not have TLS security.

private
[**on** | **off**]

on
All data connections subsequently established are secured with TLS.

off
Data connections subsequently established are not secured with TLS.

If no argument is specified, the switch changes from *on* to *off* or from *off* to *on* depending on the initial value.

If no control connection secured by TLS exists, *protec on* is implemented explicitly if *private on* is set.

prompt - Enable/disable prompting

When using *mdir*, *mget*, *mls*, *mput* and *mdelete*, multiple file operations can be executed with a single call. If prompting is enabled, you will be asked before each file operation whether you really want the operation executed. The *prompt* command enables or disables prompting. If prompting is enabled, it is disabled by the *prompt* command, and vice versa. On starting FTP, prompting is enabled by default. The current setting of the prompt option can be determined with the *status* command.

prompt

If prompting is enabled, the possible responses to the prompts are as follows:

- y The action is executed.
- n The action is not executed. The command is not aborted, but continues with the next subaction.
- q The action is not executed. The entire command is aborted.

Inputs other than *n* or *q* are interpreted as *y*.

Example

See the *mdelete* command on page 151.

protect - Enable/disable TLS security for the control connection

The *protect* command can be used to change the switch for TLS security for the control connection.

The first time TLS security is enabled the message `[Initializing TLS ...]` is issued and you may then be requested to key in some characters as randomly as possible or merely to press the ENTER key (see the option “-tlsRandomSeed - Initialize pseudo random number generator” on page 102). This initializes the pseudo random number generator. You may again be requested to key in characters as randomly as possible or merely to press the ENTER key until enough material is available for initialization.

You can inquire the current setting for this option using the *status* command.

The *protect command* has no influence on any existing control connection.

protect
[**on** | **off**]

on
All control connections subsequently established are secured with TLS.

off
Control connections subsequently established are not secured with TLS.

If no argument is specified, the switch changes from *on* to *off* or from *off* to *on* depending on the initial value.

proxy - Controlling a connection to two remote systems

The *proxy* command controls a connection to two remote systems for transferring files between these two remote systems. A requirement here is that the second remote system supports the PASV or EPSV command.

proxy
<ftp-command>

<ftp-command>
Specifies an FTP client command:

– In order to establish the control connection to the second system, the first <ftp-command> must be the *open* command.

– When you enter `proxy help` the other FTP commands are shown which can be executed on the secondary connection.

The following commands behave differently if they are prefixed with *proxy*:

- *get* and *mget* transfer files from the first server to the second server.
- *put*, *mput* and *append* transfer files from the second server to the first server.

The figure below sketches the basic procedure for transferring a file <file> between two remote servers A and B. C is the client.

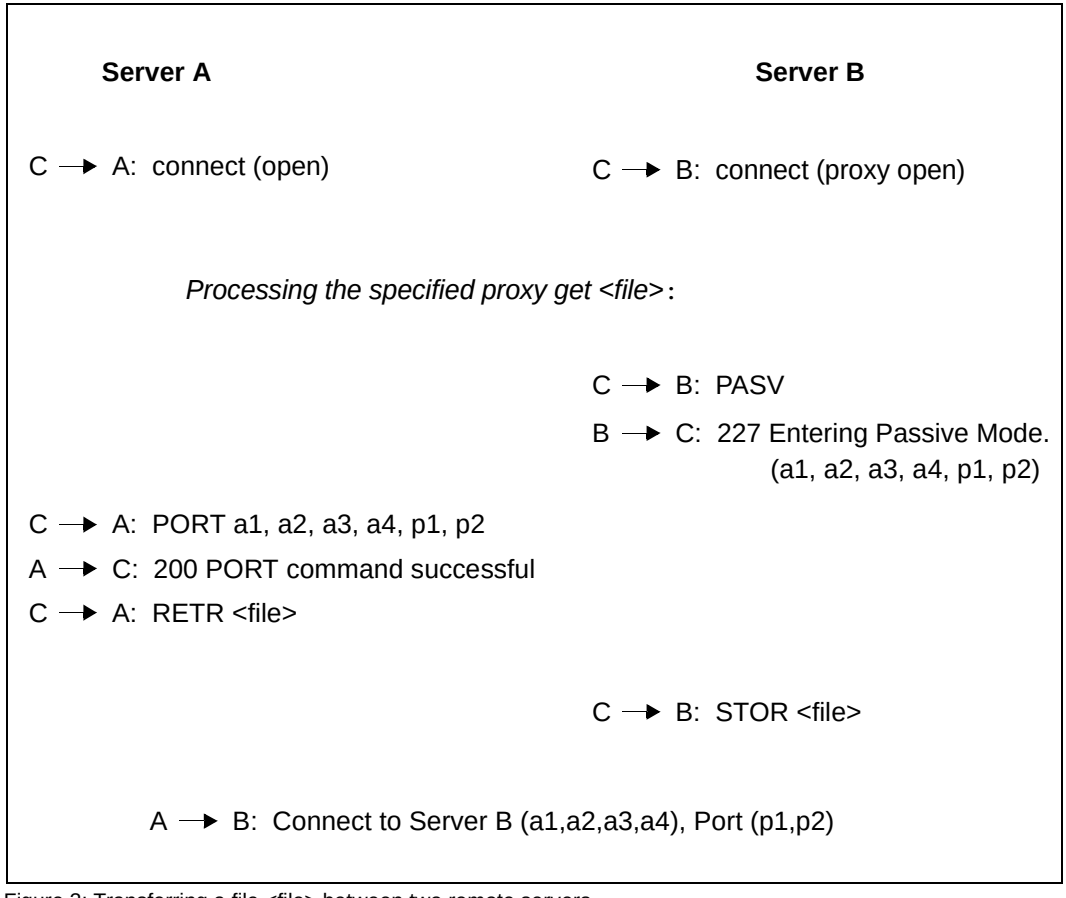


Figure 3: Transferring a file <file> between two remote servers

Example

All files of the ID *userid1* on the remote *system1* are to be transferred to the ID *userid2* on the remote *system2*.

```
ftp> open system1        Open control connection to 1st server
userid1                  .
.                        .
.                        .
.                        .
ftp> proxy open system2 Open control connection to 2nd server
userid2                  .
.                        .
.                        .
.                        .
ftp> proxy?              Which proxy commands are supported?
ftp> proxy mget*         Transfer filessystem1 -> system2
ftp> proxy ls            Check result
ftp> proxy close         Close secondary control connection
```

The *status* command returns the following output in this case:

```
ftp> status
Connected to PGAB0021, port21.
Connected for proxy commands to system2.
Passive Mode: off
...
```

put - Send a local file

The *put* command is used to transfer a file from the local host to the remote host. Files may also be transferred to a remote host with the *mput*, *append* and *send* commands. A special variant of the *put* command, the *reput* command (see page 184), is now available for supporting restart functionality.

put
<local-file> [<remote-file>]

<local-file>
Name of a POSIX or DMS file on the local host to be transferred to the remote host.

<remote-file>
Name of a file on the remote host. If the file already exists, it is overwritten or, alternatively, a new file is created after adding an appropriate suffix. This behavior is controlled by *sunique*. If the file does not exist, a new file is created.
If the *remote-file* operand is omitted, the name of the local file is used (in which case the name of the local file must also comply with the file naming conventions of the remote host). By default uppercase characters in the name of the local file are converted to lowercase. However, you can disable this setting using the client command *setcase* (see page 192).

Example

The remote host is a Unix host.

1. Query the names of local and remote working directories.

```
lpwd
Local directory is :5:$TCPTEST.

pwd
257 "/usr/tcptest/man/sam/nach.bs2000" is current directory.
```

2. Transfer the contents of the local file *MAN.SAM.TO.SINIX.ANTON* to the remote file *anton*. The FTP server function PORT is called implicitly.

```
put man.sam.nach.sinix.anton anton
200 PORT command okay.
...
```

pwd - Show remote working directory

The *pwd* command shows the name of the currently set remote working directory. The remote working directory can be set by using the *cd* command.

pwd

Example

The remote host is a Unix host.

Query the name of the remote working directory.

```
pwd
257 "/usr/tcptest/man/sam/nach.bs2000" is current directory.
```

quit - Exit FTP

The *quit* command terminates the FTP program. If a connection to a remote host is still open, it is cleared (implicit *close*). The *bye* command can be entered as a synonym for *quit*.

quit

Example

```
quit
221 Goodbye.
```

quote - Call server functions

The *quote* command can be used to call FTP server functions on the remote host when there is no corresponding FTP client command on the local host.

One example of this is the *site exec* function, which is available from the BS2000 FTP server, but for which there is no command in FTP. *site exec* can be used to execute any BS2000 commands on the server, provided

- the FTAC functionality is not being used on the server
- the *site exec* function has not been disabled via the server option *-disableSiteExecCommand* (see the “interNet Services Administrator Guide”).

Since the functions that can be called using *quote* depend on the server, a more detailed description cannot be offered here. The functions of the BS2000 server are described in the section “FTP servers in BS2000” (see page 67).

The functions implemented in the server can be determined by using the *remotehelp* command. The *help* command returns a list of the commands available on the local FTP client.

quote
<argument> [<argument>] [<argument>]

The *quote site setc* command can be used to change the code conversion table setting on the BS2000 server. The command syntax is the same as for *setcode* (see page 193).

Example

The remote host is a BS2000 host.

1. The *quote CWD* command is used to change the directory on the remote server.

```
quote CWD test
200 ":110:$TSOS.TEST." is current directory now.
```

2. The *quote PWD* command is used to check the current file system on the remote FTP server.

```
quote PWD
257 ":110:$TSOS.TEST." is current directory.
```

readopt - Read in option file

An option file is read in with the *readopt* command. This command supplements/replaces the options which were read from the global and local option files when the FTP client was started (if these files existed at startup time). You can execute the *readopt* command as often as you wish, for example to form option records which are distributed over different files, or to switch between various option records.

The *readopt* command should, if possible, not be executed for an existing FTP connection. Depending on the contents of the option file this could result in inconsistencies between the settings in the local and the remote systems.

Please note that for some TLS options any change made after TLS initialization remains ineffective. If you wish to work with changed TLS options, you must terminate and then restart the FTP client.

readopt
<option-file 1..54>

<option-file 1..54>
Name of the option file.

recv - fetch a file

The *recv* command is used to transfer a file from the remote host to the local host.

recv
<remote-file> [<local-file>]

See the description of the *get* command on page 138.

reget - Fetch a file with restart support

The *reget* command is a special variant of the *get* command (see page 138) and supports the restart functionality of the FTP client.

You must ensure that the same settings are chosen for *mode*, *type*, *struct* and *ftyp* as for the aborted *get* call.

As with the *get* command, the *reget* command transfers a file from the remote host to the local host. However, the *reget* command checks before starting a transfer whether the target file (local file) already exists.

If the target file already exists, two different scenarios can be distinguished:

- Execution of the *reget* command assumes that the file has been partially transferred and continues the transfer from the point corresponding to the current end of the target file, if the following conditions are fulfilled:
  - The target file is smaller and was produced later than the source file.
  - The FTP server supports the server commands *mdtm*, *size* and *rest* which are required for checking and positioning (see page 76).
  - The transfer medium set is "stream" (see *mode* command on page 162).
  - The *ftyp* setting is *textbin*.
  - The *setfile* setting is *datend: on*, *pademptyrec: off*
  - *runique* is disabled.
- Otherwise, the execution of the *reget* command corresponds to the execution of *get*, i.e. the source file is transferred completely.

When the command is executed the local file is not checked for correct file content or correct file attributes.

reget
<remote-file> [<local-file>]

<remote-file>
Name of a file on the remote host, which is to be transferred to the local host.

<local-file>
Name of a POSIX or DMS file on the local host. If the file does not exist, it is created and the *reget* command behaves in the same way as the *get* command.
If the *local-file* operand is omitted, the name of the remote file is used (in this case the name of the remote file must correspond to the file naming conventions of the local host).

remotehelp - Get help on functions of the remote FTP server

The *remotehelp* command returns a list of all functions implemented on the FTP server of the remote host. The individual server functions can be specified as an operand of the command. Server functions for which there are no corresponding commands on the FTP client can be called by using the *quote* command. A list of all commands on the local FTP client can be obtained with the *help* command.

remotehelp
<server-fct>

<server-fct>
FTP server function for which information is to be output.
If no operand is specified, a list of all functions recognized is displayed.

Example

The remote host is a BS2000 host.

List of the FTP server functions. Functions marked with an * (asterisk) are not implemented.

```
remotehelp
214-The following commands are recognized (* =>'s unimplemented).
 USER     EPRT     APPE     MRCP*    XCWD     MKD      XCUP     OPTS     MODC
 PASS     EPSV     MLFL*    ALLO     LIST     XMKD     STOU     AUTH     SETC
 ACCT     TYPE     MAIL*    RNFR     NLST     RMD      SYST     PBSZ     MLSD
 REIN*    STRU     MSND*    RNTO     SITE     XRMD     REST     PROT     MLST
 QUIT     MODE     MSOM*    ABOR     STAT     PWD      MDTM     CCC
 PORT     RETR     MSAM*    DELE     HELP     XPWD     SIZE     FILE
 PASV     STOR     MRSQ*    CWD      NOOP     CDUP     FEAT     FTYP
214 Direct comments to TSOS at BCAMVM06.
remotehelp site
214-The following SITE commands are recognized (* =>'s unimplemented).
 CMOD     FILE     HELP     SETC     EXIT
 EXEC     FTYP     MODC     SFIL     SVFS
214 Direct comments to TSOS at BCAMVM06.
```

rename - Rename remote files

The *rename* command changes the name of a file on the remote host.

rename
<remote-file1> <remote-file2>

<remote-file1>
Old name of the file.

<remote-file2>
New name of the file.
Any existing file is generally overwritten, but not in the DMS directory.

Example

The remote host is a Unix host.

1. Query the name of the remote working directory.

```
pwd
257 "/usr/tcptest/man/sam/to.bs2000" is current directory.
```

2. Rename the remote file *anton* to *zwaton*.

```
rename anton zwaton
350 File exists, ready for destination name
250 RNTO command successful.
```

reput - Send a local file with restart support

The *reput* command is a special variant of the *put* command (see page 175) and supports the restart functionality of the FTP client.

You must ensure that the same settings are chosen for *mode*, *type*, *struct* and *ftyp* as for the aborted *put* call.

As with the *put* command, the *reput* command transfers a file from the local to the remote host. However, the *reput* command checks before starting a transfer whether the target file (file on server) already exists.

If the target file already exists, two different scenarios can be distinguished:

- Execution of the *reput* command assumes that the file has been partially transferred and continues the transfer from the point corresponding to the current end of the target file, if the following conditions are fulfilled:
  - The target file is smaller and was produced later than the source file.
  - The FTP server supports the server commands *mdtm*, *size* and *rest* (see page 76) which are required for checking and positioning.
  - The transfer medium set is "stream" (see *mode* command on page 162).
  - The *ftyp* setting is "textbin".
  - The *setfile* setting is "datend: on, pademptyrec: off"
  - *sunique* is disabled.
- Otherwise, the execution of the *reput* command corresponds to the execution of *put*, i.e. the source file is transferred completely.

When the command is executed the local file is not checked for correct file content or correct file attributes.

reput
<local-file> [<remote-file>]

<local-file>
Name of a POSIX or DMS file on the local host, which is to be transferred to the remote host.

<remote-file>
Name of a POSIX or DMS file on the remote host. If the file does not exist, it is created and the *reput* command behaves in the same way as the *put* command.

If the <remote-file> operand is omitted, the name of the local file is used (in this case the name of the local file must correspond to the file naming conventions of the remote host). Uppercase letters in the local file name are converted to lowercase (see also the FTP command *setcase* on page 192).

rexit - Define parameters for remote exit routines

The FTP client allows you to set user-defined BS2000 FTP server exits with the *rexit* (remote exit) command.

You will find further details on FTP server exit routines in the “interNet Services Administrator Guide”.

rexit
[receive:<receive-selector>] [!] [send:<send-selector>]

<receive-selector>

<receive-selector> allows you to select the required action from several actions available when receiving data (from the server’s perspective). For this purpose, you supply the exit routine with a string to which <receive-selector> points. You can decide freely the meanings that are to be attached to the individual strings. Only the strings “*” and “*NONE” have a fixed meaning:

– If you specify "*" for <receive-selector>, the exit routine is called again with the <receive-selector> selected in the FTP server option *-U*. If no <receive-selector> is selected in this server option, the specification of “*” has the same meaning as “*NONE”.

– If you specify "*NONE", the exit routine is disabled.

<send-selector>

<send-selector> allows you to select the required action from several actions available when sending data (from the server’s perspective). For this purpose, you supply the exit routine with a string to which <send-selector> points. You can decide freely the meanings that are to be attached to the individual strings. Only the strings “*” and “*NONE” have a fixed meaning:

– If you specify "*" for <send-selector>, the exit routine is called again with the *<send-selector>* selected in the FTP server option *-U*. If no *<send-selector>* is selected in this server option, the specification of “*” has the same meaning as “*NONE”.

– If you specify “*NONE”, the exit routine is disabled.

rmdir - Remove a remote directory

The *rmdir* command is used to remove an empty directory on the remote host.

rmdir
<remote-directory>

<remote-directory>
Fully qualified or relative name of a directory on the remote host.

Example

The remote host is a Unix host.

1. List the contents of the current working directory. The FTP server function PORT is called implicitly.

```
dir
200 PORT command okay.
...
-rwx------ 1 tcptest 10505 Sep 5 18:20 alaska
-rw-rw-r-- 1 tcptest 229 Sep 7 13:56 anton
-rw-rw-r-- 1 tcptest 229 Sep 7 13:56 anton.3
-rw-rw-r-- 1 tcptest 229 Sep 7 13:56 anton.upd
-rwx------ 1 tcptest 10505 Sep 5 18:20 georgia
drwxrwxrwx 2 tcptest 176 Sep 7 12:55 nach.bs2000
drwxrwxrwx 2 tcptest 176 Sep 7 13:51 von.bs2000
drwxrwxrwx 2 tcptest 32 Sep 7 14:40 von.msdos
...
```

2. Remove the (empty) directory *from.msdos*.

```
rmdir from.msdos
250 RMD command successful.
```

runique - Enable/disable unique storage of local files

The *runique* command controls the handling of files being stored on the local host, i.e. when working with *get*, *mget* or *recv*. The *runique* command toggles the current setting on or off and remains in effect until the next call to *runique*.

On starting FTP, *runique* is disabled by default.

runique disabled: If there is already an existing file of the same name, it is overwritten, provided the appropriate write authorization exists.

runique enabled: If the file to be transferred already exists, the existing file name is extended with a suffix (.1 to .99), and a new file is created. If the suffixes from .1 to .99 have already been used for a particular file name, an error message is issued, and the transfer is aborted.

runique

send - Send a local file

The *send* command is used to transfer a file from the local host to the remote host.

send
<local-file> [<remote-file>]

See the *put* command on page 175.

sendport - Enable/disable port command

The *sendport* command defines whether or not FTP should call the PORT function in the remote host's server at the start of a file transfer. If the PORT function is enabled, the *sendport* command disables it, and vice versa. On starting FTP, the PORT function is enabled.

Normally the PORT function must be implemented in each server. However, there are servers which do not behave in a standardized manner and which do not have the PORT function implemented. Such servers will respond to a PORT command with the error message:

```
550: Command not understood
```

or

```
502: PORT command not implemented
```

In this case, the PORT function must be disabled.

sendport

set - Set and display variables

With the *set* command variables can be set and displayed, which control the behaviour of the FTP client.

Currently with this command only the variable *sizeCmdTimeLimit* is modified, which controls the behaviour of the *reget* and *reput* command.

A *reget* or *reput* command determines among other things the size of the local file for finding a restart point for the data transfer.

Especially when the transfer type is not binary, the determination of the file size can be very time consuming, so the usage of *reget*/*reput* yields barely a time advantage.

With the variable *sizeCmdTimeLimit* the duration of the size determination can be limited, i.e. when the limit value is exceeded the size determination is cancelled and as a result instead a *reget*/*reput* command the corresponding *get*/*put* command is executed.

The current valid value can be queried with a parameterless call of the *set* command or with the *status* command.

set
`sizeCmdTimeLimit` <value>

sizeCmdTimeLimit <value>
: Maximum duration of the size determination in seconds.
The default is 60 seconds.

setcase - Uppercase/lowercase for file names in the target system

The *setcase* command defines the use of uppercase/lowercase letters in the target system for the file names of the files transferred with *put* and *mput*.

setcase
[**all** | **posix** | **off**]

all
: In the event of transfer from BS2000 and POSIX to the target system, uppercase and lowercase letters in the file names remain unchanged. For transfer from BS2000, this means the file names are written in uppercase.
If no operand is specified, this is equivalent to the specification `setcase all`.

posix
: Uppercase and lowercase letters in the file names remain unchanged only when transfer is from POSIX to the target system. Names of files which are transferred from BS2000 to the target system are written in lowercase letters.

off
: The file names in the target system are written in lowercase letters.
off is the default value is no *setcase* statement is specified.

setcode - Change code tables

The *setcode* command is used to change the current code tables with which the FTP client converts from EBCDIC to ISO characters (extended ASCII character set). The services of XHCS are utilized for this purpose, so only code tables that have been entered as compatible in XHCS may be specified here (see the “XHCS” manual). If a connection is set up between two BS2000 systems, it is important to ensure that both partners use the same code tables.

The current code table settings can be checked with the *status* command.
Default settings: EDF041 for EBCDIC
ISO88591 for ASCII

setcode
<ebcdic-table> <iso-table>

<ebcdic-table> <iso-table>

A code conversion table is generated with XHCS for conversion between EBCDIC and ISO codes. This conversion table is used by the FTP client for all code conversions.

CAUTION!

It is essential to ensure that the code tables are specified in the correct sequence; otherwise, invalid conversion tables will be created, and the file transfer will not be possible.

Example

```
setcode EDF045 ISO88595
```

Remarks

The conversion table between the EBCDIC table EDF045 and the ISO table ISO88595 is made available via XHCS. The usable conversion tables are defined in XHCS. From this conversion table, the corresponding table for conversion in the opposite direction is created, and the previously valid conversion table is overwritten.

setfile - Enable/disable file marker

The *setfile* command allows you to define the behavior of FTP for file transfer in particular cases:

- Mark the end of a file
  There are two ways to mark the exact end of a PAM file:
  - The customary method uses a special string for this, which contains "C-FILEEND".
  - The new method called Last Byte Pointer (or LBP for short) uses information that is stored in the file catalog entry. The file itself is not modified. If the file is processed further by programs that have difficulty handling the marking with the special string, the appending of this marker must be disabled or the LBP method used instead.
- Records without content (dummy records) can be assigned a blank by FTP so that EDT accounts for the records when they are output on the terminal.

setfile
[datend on | off] | lbp| [pademptyrec on | off]

datend on | off | lbp
: Enables/disable the use of the EOF marker "C-FILEEND" or enables the use of the new EOF marking method LBP (lpb).
on is the default.

pademptyrec on | off
: Enables/disables the insertion of a blank in dummy records.
off is the default.

If you specify *setfile* without parameters, FTP outputs the current settings for *datend* and *pademptyrec*.

If you want to set *datend* and *pademptyrec* on the FTP server, enter *quote site sfil*. The parameters correspond to the *setfile* parameters.

The *status* command (see page 196) likewise provides information on the *datend* and *pademptyrec* settings.

settime - Set time limit for server responses

The *settime* command can be used to modify the time limit for server responses (the default setting is 30 sec). If a server response that is expected by the client is not received within the defined timeout, the client issues the message:

```
time limit for server response exceeded.
```

The currently set time limit can be determined with the *status* command.

settime <sec>

status - Display FTP status information

The *status* command returns information on the current connection, any TLS security for this connection and various locally set parameters. The command also displays information on the currently set code tables, the character string for switching between the DMS and POSIX file systems, and the current file system (DMS or POSIX).

status

Example

```
status
Not connected.
No proxy connection.
Passive Mode: off
Mode: stream; Type: ascii; Form: non-print; Structure: file
Copymode: off; Ftyp: textbin
Verbose: on; Bell: off; Prompting: on; Globbing: on
Filesystem is: BS2000
The character to change the filesystem is: %
ISO-codetable is ISO88591, EBCDIC-codetable is EDF041
Time limit for server responses: 30 secs
Time limit for file size determination: 60 secs
Store unique: off; Receive unique: off
Use EOF marker: special EOF marker (C-DATEIENDE)
Pad empty record with blank: off
Case sensitivity: Off
Used job variable: *NONE
Used SDF-P variable: *NONE
Used receive selector: *NONE
Used send selector: *NONE
Hash mark printing: off; Use of PORT cmds: on
Protected control channel: off
Private data channel: off
Cipher: (not connected)
```

The last three lines show that TLS security for the control and data connections is disabled and that currently no control connection is established.

Two further possible status displays on TLS security are listed below:

- Status display for non-secured control connection:

```
Protected control channel: off
Private data channel: off
Cipher: clear
```

- Status display for a control connection secured with 3-DES:

```
Protected control channel: on
Private data channel: off
Cipher: DES-CBC3-SHA (168 bits)
```

In these examples TLS security for the data connections is disabled. This can be recognized from the following display: `Private data channel: off`

In the case of "`Private data channel: on`", too, TLS security for the data connections is only implemented if the control connection is also secured.

struct - Change or query transfer structure

The *struct* command defines the transfer structure or displays details of the current transfer structure. The FTP standard provides for a number of different, optional transfer structures, of which only FILE and RECORD are implemented in BS2000. If no operand is specified, the current transfer structure is displayed on the screen.

On starting FTP, *struct* is set to *file* by default.

The currently defined transfer structure can also be determined with the *status* command.

struct
[**file** | **record**]

file
Sets the transfer structure to FILE.

record
This entry enables files to be transferred while retaining the record structure. In order to specify *record*, the following conditions must be satisfied:

– The transfer type must be EBCDIC or ASCII, and
– The file attributes RECFORM=V and FCBTYPE=SAM must be set.

sunique - Store remote files uniquely

The *sunique* command controls the handling of files being stored on the remote host, i.e. when working with *put*, *mput* or *send*. The *sunique* command toggles the current setting on or off and the defined setting remains in effect until *sunique* is called again.

On starting FTP, *sunique* is disabled by default.

sunique disabled: If there is already an existing file of the same name, it is overwritten, provided the appropriate write authorization exists.

sunique enabled: If the file to be transferred already exists, it is recreated using a unique name.

If the partner is a BS2000 FTP server the unique name is created by appending a suffix (.1 to .99). If the suffixes from .1 to .99 have already been used, an error message is issued and the transfer is aborted. Other FTP servers may have other conventions for creating the unique name, see documentation of the respective FTP server.

sunique

svar - Store error information in an SDF-P variable

You can use the *svar* command to inform the FTP client whether it is to store command return information in an SDF-P variable.

The *jobvar* command (see page 143) offers an alternative to storing error information.

If an error occurs when editing the *svar* command and if task switch 1 is set (batch mode), the FTP client is terminated with TERMJ. The batch job or procedure is then only continued after the next STEP instruction.

If the provision of an S variable is activated, the behavior of commands that execute several individual actions (commands *mdir*, *mls*, *mget*, *mput*, *mdelete*) changes. In the case of these commands, command processing is aborted after the first faulty action. In contrast, the FTP client is not terminated with TERMJ in batch mode if the admission data in the *open* command is faulty.

svar
<sv-name> | ***NONE**

<sv-name>
: Name of the SDF-P variable (S variable) which the FTP client is to supply with the command return information after the *svar* command is issued. This S variable is created again for this purpose. If an S variable with the name <sv-name> already exists, it is deleted first.

***NONE**
: The provision of S variables is halted by entering *NONE.

Layout of S variables

Element	Element description
<sv-name>.STATUS	$S: Command executed successfully $E: Command has errors $T: FTP client ended normally $A: FTP client ended abnormally with errors
<sv-name>.USERCMD	Name of FTP client command
<sv-name>.CMDPARAM	FTP command parameter
<sv-name>.PROTCMD	Command sent from the FTP client to the FTP server
<sv-name>.MESSAGE	Local message or server response

system - Display server information

The *system* command is used to request system information from the server, which then issues the following message:

```
215: <operating system>  <additional information>
```

system

tenex - Set transfer type to BINARY

The *tenex* command sets the transfer type to BINARY. This transfer type should always be selected when transferring binary files. The *binary* command is equivalent to the *tenex* command.

After starting FTP, the transfer type is set to ASCII by default.

tenex

trace - Enable/disable SOCKET trace output

SOCKET trace output is primarily used by the network administrator and the support staff to diagnose problems on the local network. Normal users do not usually need any SOCKET trace output.

If SOCKET trace output is enabled, FTP displays various types of diagnostic information on the screen.

trace
[<trace-value>]

<trace-value>
The permitted values are in the range 0 through 10. Higher parameter values display more information.

0 No TRACE output (disable TRACE output).

1 - 10 Displays information on SOCKET traces.

If no operand is specified, the TRACE output setting is toggled, i.e. if TRACE output was enabled, it is disabled, and if it was disabled, it is enabled (*trace-value*=1). Parameter values greater than 1 imply the output of all TRACE information at the lower levels.

type - Change or query transfer type

The *type* command sets the transfer type (ASCII, BINARY, TENEX or EBCIDC). If no operand is specified, the currently set transfer type is displayed on the screen.

On starting FTP, the transfer type is set to ASCII by default.

The transfer type can also be changed by using the *ascii* and *binary* commands. The transfer type currently set can be determined with the *status* command. The possible transfer types are described in more detail from page 89. A table for ASCII/EBCDIC conversion can be found in the “XHCS” manual.

When a connection is cleared and a new connection is set up, the *type* command must be issued again for a transfer type other than ASCII.

type
[**ascii** | **binary** | **ebcdic** | **tenex**]

ascii
Sets the transfer type to ASCII. This transfer type should always be selected for text files, unless EBCDIC is in question.

binary
Sets the transfer type to BINARY. This transfer type should always be selected for binary files.

ebcdic
Sets the transfer type to EBCDIC. This transfer type should be selected if both partners are working with EBCDIC, i.e. of no code conversion is desired and a SAM file is to be generated in BS2000.

tenex
Corresponds to the transfer type BINARY.

Example

1. Query the transfer type.

```
type
Using ascii mode to transfer files.
```

2. Set the transfer type to BINARY.

```
type binary
200 Type set to I.
```

3. Query the transfer type.

```
type
Using binary mode to transfer files.
```

user - Specify user ID on remote host

You can use the *user* command to enter the admission data at a later stage in the following cases:

- If an access control mechanism is implemented on the remote host.
- Task switch 1 is not set or the storage of error information (commands *jvar*, *svar*) is enabled.
- The required admission data was incorrectly entered in the *open* command.

user
[<userid> [password> [<account>]]]

<userid>
User ID on the remote host.

<password>
Password on the remote host.
Passwords are specified in accordance with the conventions of the partner system. For BS2000 partners, they may be specified as C-strings (c'...' or C'...') or as hexadecimal numbers (X'...', with X always uppercase). BS2000 accepts the password in the formats <password> and <'password'>. If no password is required, enter "*NONE".

<account>
Account number on the remote host.

If any additional authorization data is required (password, account number), it is requested in interactive mode.

Example

The remote host is a Linux host.

1. Setting up a connection to systemd using the *open* command.

```
open systemd
Connected to anlaged.
220 anlaged FTP Server (vsFTPd 2.0.5).
ready.
```

2. Selection of the login ID *tcptest*.

```
Name (systemd:TCPTEST):
tcptest
```

3. Entry of an invalid password.

```
331 Password required for tcptest.
Password (systemd:tcptest):
maewest

530 Login failed.
Login failed.
```

4. Repetition of the login.

```
user tcptest
```

5. Specification of the correct password; successful login.

```
331 Password required for tcptest.
Password:
Kr!fm3(z
230 Login successful.
```

verbose - Enable/disable server responses

The *verbose* command is used to enable or disable the display of responses from the FTP server. If the output of responses is enabled, the *verbose* command disables it, and vice versa. On starting FTP, *verbose* is enabled by default.

The current setting for *verbose* can be determined by using the *status* command.

verbose

Example

The remote host is a Unix host.

1. Query the name of the remote working directory with *verbose* enabled.

```
pwd
257 "/usr/tcptest/man/sam/nach.bs2000" is current directory.
```

2. Transfer multiple files (with server responses). The FTP server function PORT is called implicitly. The *prompt* function is disabled.

```
mget *
200 PORT command okay.
150 ASCII data for anton (89.16.100.0,1192).
226 Transfer complete.
242 bytes received in 0.06 seconds
(3.69 Kbytes/s)
200 PORT command okay.
150 ASCII data for anton.1 (89.16.100.0,1193).
226 Transfer complete.
150 ASCII data for caesar (89.16.100.0,1199).
226 Transfer complete.
10845 bytes received in 0.78 seconds (13.56 Kbytes/s)
```

3. Rename a file (with server responses).

```
rename anton zwaton
350 File exists, ready for destination name
200 RNTO command okay.
```

4. Disable the server responses.

```
verbose
Verbose mode off.
```

5. *pwd*, *mget* and *rename* commands without server responses.

```
pwd
mget *
rename anton zwaton
```

? - Show information on FTP commands

The *?* command returns a list of all FTP commands. It is also possible to obtain brief information on each of these FTP commands. The *?* command is synonymous with *help*.

?
<command>

See the description of the *help* command on page 142.

! - Switch to BS2000 or POSIX command mode

Switching to the BS2000 command mode (BS2000-FTP)

The *!* command enables you to enter BS2000 commands during an FTP session. If no operand is specified, the system switches to BS2000 command mode. You can return to FTP by issuing the BS2000 command *RESUME*. Alternatively, if you specify *!* with a BS2000 command as the operand, you will be automatically returned to the FTP client after execution of the command.

It is not possible to switch to POSIX command mode. However, POSIX commands can be entered by switching beforehand to the POSIX directory using the "lcd %POSIX" command.

!
[<bs2000-command>]

<bs2000-command>
: Any BS2000 command, except for the commands *LOAD*, *EXEC*, *LOGOFF* and *ABEND*, which may not be entered either directly or indirectly, since they would terminate FTP or the task in which FTP is running.
To avoid inconsistencies in the catalog, please make sure that you always use the FTP command *file* instead of the BS2000 FILE command.

Example

Issue the BS2000 command *sta p* from FTP:

```
! sta p
NAME TSN TYPE SIZE CURR-CMD PROGRAM-NAME
PETER 1447 3 DIALOG 87 EXECUTE :5:$TSOS.FTP

ftp>

!

x%BKPT AT xxxxxx

/sta p
NAME TSN TYPE SIZE CURR-CMD PROGRAM-NAME
PETER 1447 3 DIALOG 87 EXECUTE :5:$TSOS.FTP

/

resume
ftp>
```

Switching to POSIX command modem (POSIX-FTP)

Entering the *!* command enables you to enter POSIX commands during an FTP session. If no operand is specified, the system switches to POSIX command mode and subsequently returns to FTP by issuing the POSIX command *exit*. If you enter *!* with a POSIX command as the operand, you will automatically be returned to the FTP client after execution of the command.

!
[<POSIX-command>]

<posix-command>
Any POSIX command.

4.11 FTP C subroutine interface YAPFAPI

The C subroutine interface of FTP is closely based on the interactive interface and is implemented by the C function *YAPFAPI*. The calling program transfers the FTP command to *YAPFAPI* as a character string. As a result, *YAPFAPI* returns FTP server message(s) and local error information to corresponding buffers.

If the calling program does not want to write the output of a *dir* or *ls* command to file but, for example, wishes to interpret it directly, it uses the *dirCallBackFct* parameter (see below) to transfer the address of a corresponding callback function with the following signature:

```
void dirCallBackFct(char *buffer, int bufLen);
```

buffer here is a pointer to a buffer containing part of the output of a *dir* or *ls* command. *buflen* specifies the length of the buffer contents.

If client authentication is used for TLS-secured channels, the calling program can transfer a passphrase for the private key.

When linking the program which calls the FTP subroutine interface, users of the FTP subroutine interface using the standard C/C++ compiler should link the FTPAPI module in from the following library:

SYSLIB.TCP-IP-AP.*nnn* (for /390 servers)

SKUOML.TCP-IP-AP.*nnn* (for x86 servers)

In addition, the external reference to the user-defined exit routine must be resolved (see the “interNet Services Administrator Guide”). If you do not need an exit routine of your own, you can use dummy module EXITFTP from the following library for this purpose:

SYSLNK.TCP-IP-AP.*nnn* (for /390 servers)

SKMLNK.TCP-IP-AP.*nnn* (for x86 servers)

As the FTP subroutine module is produced from C source texts, C runtime libraries (CRTE) must also be linked in.

Function prototype of YAPFAPI

The function prototype of the subroutine call is as follows:

```
void YAPFAPI(struct YAPFAPI_pl_mdl *param)
```

param is a pointer to a variable of the type `struct YAPFAPI_pl_mdl`.

Description of the structure YAPFAPI_pl_mdl

Data type `struct YAPFAPI_pl_mdl` which is described below is declared in the header file YAPFAPI.H in the library SYSLIB.TCP-IP-AP.*nnn*.

```
struct YAPFAPI_pl_mdl {
   struct {
      int version;                /* Interface version                    */
      char *cmd;                  /* Contains command string              */
      char *serverMsg;            /* Buffer for server messages           */
      int maxServerMsgLen;        /* Length of buffer for server messages */
      char *localMsg;             /* Buffer for local messages            */
      int maxLocalMsgLen;         /* Length of buffer for local messages  */
      int combineMessages;        /* Write all messages into one buffer   */
      void (*dirCallBackFct)(char*, int);  /* Call back function */
      char *passPhrase;           /* Password for private key             */
      int passPhraseLen;          /* Length of password for private key   */
   } in_data;

   struct {
      int msgNumber;         /* Message number of last message from server */
      int serverMsgLen;      /* Length of messages from server             */
      int localMsgLen;       /* Length of local message                    */
      int rc;                /* Summary return code                        */
   } out_data;
}
/* Values for rc */
#define YAPFAPIrcOk                        0    /* Ok                      */
#define YAPFAPIrcVersionError              1    /* Wrong interface version */
#define YAPFAPIrcInitError                 2    /* Initialization Error    */
#define YAPFAPIrcLocalError                3    /* Local error             */
#define YAPFAPIrcFatalLocalError           4    /* Fatal local error       */
#define YAPFAPIrcRemoteError               5    /* Remote error            */
```

Description of the structure elements

version
Interface version. This parameter must be assigned the value 1.

cmd
Pointer to the null terminated command string which contains the command in the same syntax as required by the interactive interface (see the section “Overview of commands (FTP client)” on page 114).

serverMsg
Pointer to a buffer provided by the calling program to receive the server messages.

maxServerMsgLen
Length of the buffer to which *serverMsg* points. If the buffer size is not sufficient to accommodate all the messages, the excess messages are either output in fragments or not at all.

localMsg
Pointer to a buffer provided by the calling program to receive local messages.

maxLocalMsgLen
Length of the buffer to which *serverMsg* points. If the buffer size is not sufficient to accommodate all the messages, the excess messages are either output in fragments or not at all.

combineMessages
If the calling program specifies a value other than 0 for *combineMessages*, both server messages and local messages are made available in the buffer referenced by *localMsg*. In this case no separate buffer need be provided for server messages.

dirCallBackFct
Address of a callback function. If no callback function is to be used, NULL must be specified.

passPhrase
Pointer to a buffer which contains the passphrase for the private key.

passPhraseLen
Length of the passphrase.

msgNumber
The number of the last server message is returned in *msgNumber* as per RFC 959 (FTP).

serverMsgLen
The integer value returned in *serverMsgLen* shows how many characters have currently been transferred to the buffer specified by *serverMsg*.

localMsgLen
The integer value returned in *localMsgLen* shows how many characters have currently been transferred to the buffer specified by *localMsg*.

rc
The complete return code is returned in *rc*.

Sample program for the use of the FTP subroutine interface

A sample program is shown below which uses the FTP subroutine interface.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "yapfapi.h"

void printDir(char*, int);

main(int argc, char *argv[])
{
   struct YAPFAPI_pl_mdl param;
   char line[200];
   char serverMsg[2000];
   char localMsg[2000];
   FILE *fp_in;

   fp_in = fopen("(SYSDTA)", "r");

   for (;;) {
      printf("ftp> ");
      if (fgets(line, sizeof line, fp_in) == 0)
         break;
      param.in_data.version = 1;
      param.in_data.cmd = line;
      param.in_data.serverMsg = serverMsg;
      param.in_data.maxServerMsgLen = sizeof(serverMsg);
      param.in_data.localMsg = localMsg;
      param.in_data.maxLocalMsgLen = sizeof(localMsg);
      param.in_data.combineMessages = 1;
      param.in_data.dirCallBackFct = &printDir;
      param.in_data.passPhrase = "ABCDEFGH";
      param.in_data.passPhraseLen = 8;
      YAPFAPI(&param);
      printf("Ret: msgNumber:    %d\n", param.out_data.msgNumber);
      if (param.out_data.serverMsgLen != 0)
         printf("Ret: serverMsgLen: %d\n", param.out_data.serverMsgLen);
      printf("Ret: localMsgLen:  %d\n", param.out_data.localMsgLen);
      if (param.out_data.serverMsgLen != 0) {
         serverMsg[param.out_data.serverMsgLen] = '\0';
         printf("Ret: serverMsg: \n");
         printf("%s", serverMsg);
         printf("Ret: serverMsgEnd\n");
      }
      localMsg[param.out_data.localMsgLen] = '\0';
      printf("Ret: localMsg: \n");
      printf("%s", localMsg);
```

```
        printf("Ret: localMsgEnd\n");
        printf("Ret: rc:        %d\n", param.out_data.rc);
        if (param.out_data.rc == YAPFAPIrcInitError)
           break;
      }
}

void printDir(char* buffer, int bufLen)
{
   printf("%.*s", bufLen, buffer);
}
```

5 FTAC interface

The FTAC function of openFT offers you the dual option of making your BS2000 server as secure as possible and as secure as required. FTAC stands for “File Transfer Access Control”. In order to use FTAC on a BS2000 server, openFT Version with openFT-AC is required.

The FTAC functions described in this chapter refer to openFT V12.1. If a different openFT version is used on your BS2000 server, the functional scope may differ.

A complete description of openFT-AC is provided in the following manual below which is available for the various openFT versions:

– openFT (BS2000) - Command Interface
  This manual contains a complete description of the FTAC functionality, the FTAC commands, and also a description of the associated CSV outputs and OPS variables.
  This manual contains all the FTAC messages.

5.1 FTAC functionality

The following features are offered by FTAC to protect the BS2000 server:

– Decoupling of transfer admissions and login admission
– Access rights depending on the partner system
– User-specific access rights
– Flexible access right levels
– Logging of every authorization check
– Simple application

5.1.1 Features of the FTAC function

For file transfer, a distinction is made between various functions. For access protection, the file transfer function being executed by the system is decisive. At first glance, there are only two such functions:

– sending a file and
– receiving a file.

Sending a file involves transmitting data from the system to be protected, while receiving a file involves the transfer of data into that system. For reasons of data security, it is, however, also important to know who requested a function on the system being protected. In FT terminology, this person is referred to as the “initiator” or “submitter” of the FT request.

A distinction is made in FTAC between two groups of initiators:

– initiators on the system being protected (outbound requests)
– initiators on partner systems (inbound requests)

Since the functionality used by FTP is restricted to inbound requests, the following transfer functions can be differentiated:

– **Inbound send**
– **Inbound receive**

FTP partner systems (FTP clients) also have the option of using file management functions to view directory or file attributes in their local system (BS2000 FTP server), to modify file attributes and to delete files and directories. This results in a further function:

– **Inbound file management**

File management, in contrast to the other functions, includes several different request options which, in turn, are partially linked to the functions *Inbound send* and *Inbound receive*:

- An FTP client can delete local files, provided the basic function *Inbound receive* is permitted.
- An FTP client can display the attributes of local files, provided the basic function *Inbound send* is permitted.
- An FTP client can display and delete directories, provided the basic function *Inbound file management* is permitted.
- An FTP client can modify the attributes of local files and create and rename directories, provided both the basic functions *Inbound receive* and *Inbound file management* are permitted.

The protection mechanisms offered by the FTAC function are primarily achieved by using admission sets and admission profiles.

5.1.2 Admission set

The admission set contains the basic specifications of which FTP functions are allowed. An admission set applies to exactly one user ID in BS2000. When access is attempted under this user ID, FTAC checks for compliance with the values given in the admission set. If desired, you can also restrict or extend the specifications of the admission set by using admission profiles or privileges, respectively. Admission profiles enable you to permit one or more individual inbound functions for specific partners. You can view admission sets at any time and modify them as required to meet your current needs.

Following the installation of openFT-AC, the entries in the standard admission set initially apply to all user IDs. As the FTAC administrator, you should modify this standard admission set after installation so that it provides the necessary protection for the majority of user IDs. If individual user IDs need greater protection, you can, of course, also create specially customized admission sets for such cases.

5.1.3 FT profile

The FT profile (also called an admission profile) defines the transfer admission and the associated access rights. The transfer admission is the actual key to accessing the BS2000 server via FTP and should therefore be treated with the same care as a password. The transfer admission must be specified in transfer requests instead of a login admission. Anyone who knows the transfer admission will have access to your user ID on the BS2000 server, but unlike the login authorization, will not be free to do as he or she pleases. You can decide which functions are to be permitted by specifying the access rights for the transfer admission. This enables you to control which files can be accessed, for example, and under what conditions. In the most extreme case, you could effectively restrict access to your user ID so severely that only one file can be accessed via only one profile. With the appropriate settings, an FT profile could be used simultaneously for both openFT and FTP.

For each file transfer request, FTAC checks whether the entries in the request conflict with the entries in the FT profile. If such a conflict exists, the FTP request is rejected, and a general error message appears on the client system. This prevents potential intruders from determining the definition of the FT profile in sequential steps on a trial and error basis. A log record that describes the precise cause of the error is created on the BS2000 server.

The following diagram illustrates the sequence for access checks with FTAC.

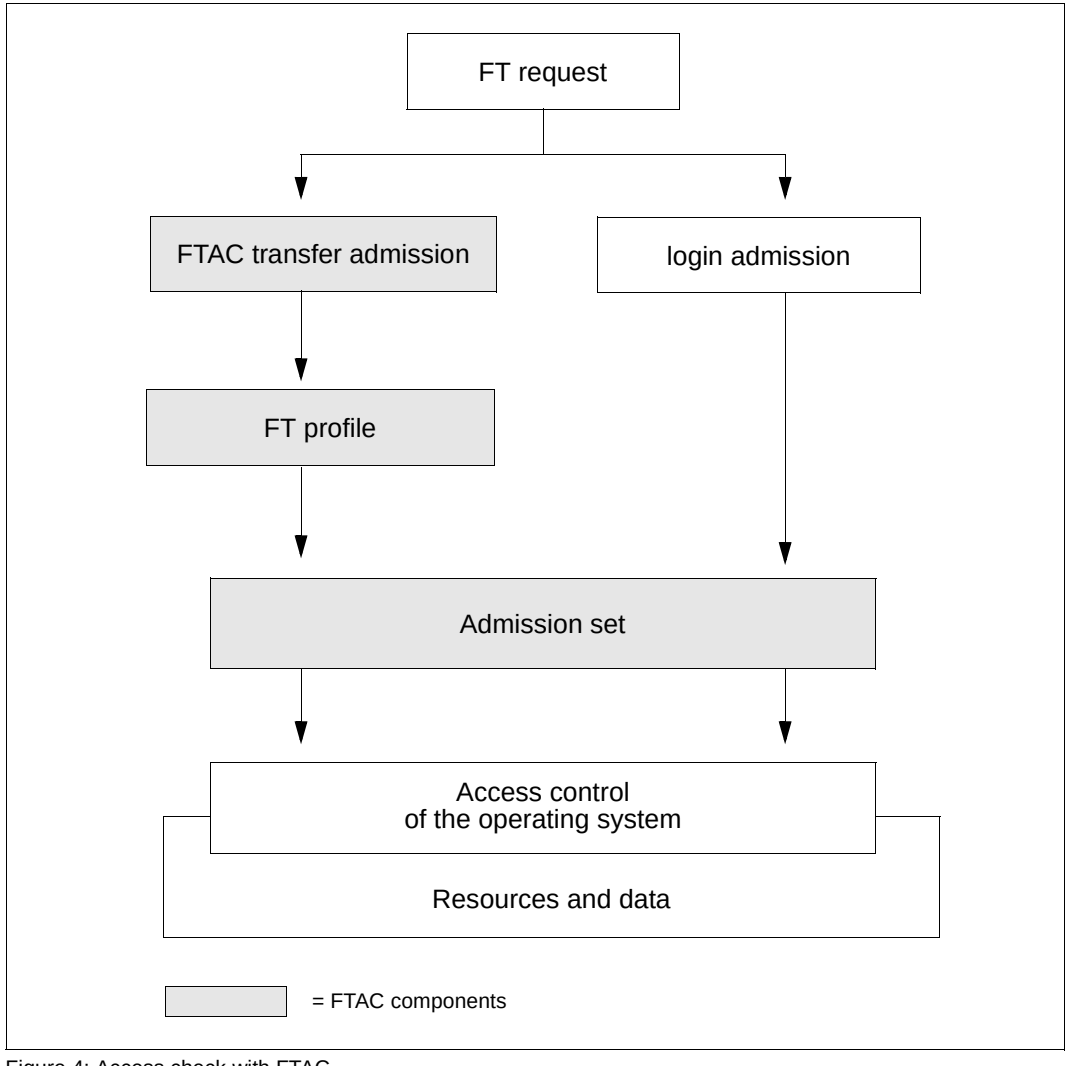


Figure 4: Access check with FTAC

An admission profile includes the following:

- a transfer admission. This transfer admission must be unique. If a request is to work with the FT profile, this transfer admission must be specified. FTAC will then only permit those access rights for the request which are defined in the FT profile. In order to uniquely assign the responsibility for requests, it is recommended that a transfer admission be assigned to exactly one person.
- if necessary, specification of the partner systems that may access this FT profile.
- details of the parameters that may be used in a request and their scope. This enables the access rights to be restricted for each person who uses this FT profile for FTP.
- if necessary, details on whether and when the FT profile can be used.
- a file name prefix. This prefix contains a part of the path name. The user of the profile can then only navigate below the specified path name. For example, if C:\USR\HUGO\ is specified as a file name prefix on a Unix system, the user of this profile will only be granted access to directories under the path C:\USR\HUGO\. This prevents users from accessing locked directories by entering ". .".

You can store many different FT profiles.

The following operations can be performed on FT profiles at any time:

- **Modify**
  and thus adapt the profile to current requirements.
- **Lock**
  In this case, a request with the locked profile is rejected on account of the invalid transfer admission. If you want to use the profile again, you must first unlock the FT profile.
- **Delete**
  You should limit the number of your FT profiles by deleting profiles which you no longer need.
- **Privilege** (system-dependent)
  In special cases, FT profiles can also utilize a function that has been locked in an admission set. In order to do this, the FT profile must be assigned a privilege by the FTAC administrator.

You can also display information on your FT profiles at any time.

5.1.4 Effects of an FT profile

The following table shows the possible restrictions to the access rights in an FT profile in the left column, with the entries needed for the FT request with respect to the partner system in the right column:

Entry in the FT profile	Entries in the FT request
Transfer admission	The transfer admission must be specified. Enter *$FTAC* as the user ID or a different character string defined by the system administrator. Use the transfer admission as the password. If an admission profile is defined with transfer admission “anonymousFTP”, access is also possible using the user ID “FTP” or “ANONYMOUS”. The password in this case must be a mail address. (it is then simply checked to verify if “@” is included.)
Transfer direction restricted	The value specified here must be the opposite of the entry in the FT profile. If the profile contains the transfer direction “From partner”, the remote system may only send data to the local system; with “To partner”, only read access is permitted on the local system.
Partner systems specified	The request can only be issued by partner systems entered in the profile. This option may only be used if the names of the FTP clients are guaranteed to be no longer than 8 characters. If DNS is activated, the client name supplied by DNS is used, otherwise the entry belonging to the IP address in the SOCKET-HOST-TABLE is relevant (see “BCAM” manual). If no corresponding entry is available, the entry in the PROCESSOR-TABLE is used. In the case of local access, “LOOPBACK” is used as the name.
Prefix for file name specified	Only part of the file name is present in the request. FTAC supplements this entry with the prefix defined in the profile to obtain the complete file name. The entry of absolute file names, or exiting a directory with “..” is prohibited by FTAC.
Follow-up processing	This FTAC function is reserved for openFT and is not used for FTP.
Syntax restriction	The request is executed only if it does not violate this syntax restriction.

Exporting admissions

You can export complete admissions as well as individual admission sets and profiles to a file (export) and then import them from the file when required.

5.1.5 Monitoring the FTP server with FTAC

The FTAC function also checks the server commands on the BS2000 FTP server.
The following table contains a list of the server commands that are checked by FTAC and the corresponding client commands which use them.

Server command	Corresponding BS2000 FTP client command
retr	*get*, *recv*, *mget*
stor	*put*, *send* (if *sunique=off*), *mput*
stou	*put*, *send* (if *sunique=on*)
appe	*append*
rfnr	*rename*
dele	*delete*, *mdelete*
site file	*quote site file*
cwd	*cd*
xcwd	*cd* (same function as *cwd*)
cdup	*cdup*
xcup	*cdup* (same function as *cdup*)
list	*dir*, *mdir*
nlst	*ls*, *mls*, *mget*, *mdelete*
site exec	*quote site exec*
mkd	*mkdir*
mlsd	*mlsd*
mlst	*mlst*
xmkd	*mkdir* (same function as *mkd*)
rmd	*rmdir*
xrmd	*rmdir* (same function as *rmd*)
pwd	*pwd*
xpwd	*pwd* (same function as *pwd*)
size	*quote size*, *reget*, *reput*
mdtm	*quote mdtm*, *reget*, *reput*

List of FTP server commands checked via FTAC

5.2 FTAC command interface

The FTAC command interface offers:

- a functional command overview provides a quick orientation of which commands are available for which tasks,
- information on entering commands,
- explanations of the return codes,
- a detailed description of all FTP commands relevant for the user in alphabetical order.

5.2.1 Functional command overview

The following overview shows the appropriate FTAC user commands for individual tasks. The use of openFT-AC for the BS2000 FTP server is a prerequisite for the use of these commands.

Commands for the FTAC admission sets

FTAC users can view the standard admission set and their own admission sets. They can also modify their own admission sets and display partner systems.

Modify admission set	MODIFY-FT-ADMISSION-SET	page 243
Show admission set	SHOW-FT-ADMISSION-SET	page 260

Commands for the FTAC admission profiles

FTAC users can create, modify, show and delete admission profiles for their own user IDs.

Create admission profile	CREATE-FT-PROFILE	page 230
Delete admission profile	DELETE-FT-PROFILE	page 241
Modify admission profile	MODIFY-FT-PROFILE	page 247
Show admission profile	SHOW-FT-PROFILE	page 272

Command for the logging function

Show logging records	SHOW-FT-LOGGING-RECORDS	page 263

5.2.2 Entering FTAC commands

Please remember the following when entering commands:

- You must insert commas to separate the individual operands of a command, e.g.

  ```
  /TRANSFER-FILE TRANSFER-DIRECTION=TO,PARTNER=CENTRE,LOCAL-PARAMETER=...
  ```

- Quotes within quotes must be doubled. This rule applies to all value assignments that have to be specified in quotes.
- If there is no default value marked (by underscoring) for an operand, then it **must** be specified with a valid value (mandatory operand).
- You may abbreviate your entries for commands and operands, provided you ensure that your entries retain their uniqueness. You can also use positional parameters if you wish. Short forms and long forms can be mixed at will. Certain abbreviated forms of keywords and a number of positional operands are guaranteed for openFT. This means that these options will also be offered in the same format in later versions. To be “on the safe side”, you should therefore form the habit of entering these commands in their abbreviated form. Make sure that you always use the guaranteed abbreviated forms in procedures, since this will ensure that they execute correctly even in subsequent versions. The recommended abbreviations are used in the examples shown in this chapter, and the possible abbreviations are listed in the individual command formats.
- If a structure is preceded by an introductory operand (e.g. *BS2000 is an introductory operand in “REM=*BS2000(...)”), then the opening parentheses must immediately follow this operand. Introductory operands may be omitted if there is no risk of ambiguity.
- The asterisk preceding keywords may be omitted if there is no risk of ambiguity. Please note that this is not a guaranteed abbreviation.

If positional operands are explicitly permitted in the syntax, values may be specified interchangeably in both positional and keyword form.

In this case, the following rules must be observed:

- The first value specified as a positional operand is assigned to the first operand, the second value to the second operand, and so on.
- For each operand omitted before a positional operand, a comma must be entered.
- If one operand is assigned one value specification each in positional and keyword form, the last specification in the structural parentheses applies. This is always the specification in keyword form, since the positional operand must always come first.

- The positional operands must be listed first. For the sake of clarity, duplicate value assignments should be avoided.
- Since it is possible that the sequence of operands could be changed in later versions, only keyword parameters should be used within procedures.

5.2.3 Command return codes

The openFT-AC supply return codes that you can query when using SDF-P. Each return code consists of a subcode1 (SC1), a subcode2 (SC2) and the maincode (MC).

Subcode1

Subcode1 represents the error class and is a decimal number.

A distinction is made between the following error classes:

– No error:
  the value of subcode1 is 0.
– Syntax error:
  the value of subcode1 is between 1 and 31, inclusive.
– Internal error (system error):
  the value of subcode1 is 32.
– Errors not assigned to any other class:
  the value of subcode1 is between 64 and 127, inclusive. If the value of subcode 1 is in this range, the maincode must be evaluated to determine the appropriate action.
– Command cannot be executed at this time:
  the value of subcode1 is between 128 and 130, inclusive.

Subcode2

Subcode2 either contains information supplementary to that in subcode1 or is equal to 0.

Maincode

The maincode corresponds to the message key of the SYSOUT message. You can use the /HELP-MSG-INFORMATION command to obtain detailed information. The chapter on *Messages* contains details on the meanings of the messages and the actions to be taken in response.

See the “User Commands (SDF Format)” manual for a detailed description of the command return codes.

Each command description below shows which command return codes are possible for the command in question and explains the meanings of the return codes.

5.2.4 CREATE-FT-PROFILE - Create admission profile

Every FTAC user can create his own admission profile for his own user ID with the CREATE-FT-PROFILE command. Admission profiles predefined by the FTAC administrator must be activated by the user with MODIFY-FT-PROFILE (see from page 247) before they can be used.

If you call *HELP* for the SDF command syntax shown below, you may also see some operands that are not indicated here. This is because only the operands relevant for FTP are described in this section.

CREATE-**FT-PROF**ILE - showing operands relevant for FTP

NAME = <alphanum-name 1..8>

,**TRANS**FER-**AD**MISSION = ***NOT-SPEC**IFIED / <alphanum-name 8..32>(...) /
<c-string 8..32 with-low> (...) / <x-string 15..64>(...) / ***SEC**RET

<alphanum-name 8..32>(...) / <c-string 8..32 with-low>(...) / <x-string 15..64>(...)

VALID = ***YES** / ***NO**

,**USAGE** = ***PRIVATE** / *__PUB__LIC

,**EXP**IRATION-**DATE** = ***NOT-REST**RICTED / <date 8..10>

,**PRIV**ILEGED = ***NO**

,**IGN**ORE-**MAX-LEV**ELS = ***NO** / ***YES** / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

,**IN**BOUND-**SEND** = ***NO** / ***YES**

,**IN**BOUND-**REC**EIVE = ***NO** / ***YES**

,**IN**BOUND-**MAN**AGEMENT = ***NO** / ***YES**

,**USER-AD**MISSION = ***OWN** / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

USER-IDENTIFICATION = ***OWN** / <name 1..8>

,**ACCOUNT** = ***OWN** / ***NOT-SPEC**IFIED / <alphanum-name 1..8>

,**PASS**WORD = ***OWN** / <c-string 1..8> / <c-string 9..32> / <x-string 1..16> / ***NONE** / ***SEC**RET

,**INIT**IATOR = ***REM**OTE

,**TRANS**FER-**DIR**ECTION = ***NOT-REST**RICTED / **FROM**-PARTNER / **TO**-PARTNER

,**PART**NER = ***NOT-REST**RICTED / list-poss(50): <text 1..200 with-low>

,**MAX-PART**NER-**LEV**EL = ***NOT-REST**RICTED / <integer 0..100>

,**FILE-NAME** = ***NOT-REST**RICTED / ***EXP**ANSION(...)

,***EXP**ANSION(...)

PREFIX = <filename 1..53> / <partial-filename 2..53> / <c-string 1..511 with-low>

,**FILE-PASS**WORD = ***NOT-REST**RICTED / ***NONE** / <c-string 1...4> / <x-string 1...8> / <integer -2147483648...2147483647> / ***SEC**RET

,**WRITE**-MODE = ***NOT-REST**RICTED / **NEW**-FILE / **REP**LACE-FILE / **EXT**END-FILE

,**FT-FUNC**TION = ***NOT-REST**RICTED / list-poss(4): ***TRANS**FER-**F**ILE / ***MOD**IFY-**F**ILE-**ATTR**IBUTES / ***READ-DIR**ECTORY / ***FILE**-**PROC**ESSING

Operands

NAME=<alphanum-name 1..8>
With NAME, the admission profile is given a name. This name must be unique among all admission profiles on your user ID. If an admission profile with this name already exists, FTAC rejects the command with the message:

```
FTC0100 FT profile already exists
```

The command SHOW-FT-PROFILE (see page 272ff) can be used to view the already existing names. To obtain this information, the command SHOW-FT-PROFILE can be entered without operands.

TRANSFER-ADMISSION=
With TRANSFER-ADMISSION, you define transfer admission. If this transfer admission is entered in an FTP-LOGON instead of the LOGON admission, then the access rights are defined in this admission profile apply. This transfer admission must be unique in the entire openFT system so that there are no conflicts with other transfer admissions defined by other FTAC users for other access rights.

If the transfer admission that you have selected has already been assigned, FTAC rejects the command with the message:

```
FTC0101 Transfer admission already exists
```

TRANSFER-ADMISSION=*NOT-SPECIFIED
This entry is used to set up a profile without a transfer admission. Such a profile remains inaccessible until a valid transfer admission has been assigned.

TRANSFER-ADMISSION=<alphanum-name 8..32>(...) / <c-string 8..32 with-low>(...) / <x-string 15..64>(...)
The character string must be entered as the transfer admission in the FTP request. The alphanumeric entry in stored internally in lowercase letters.

VALID=*YES
The transfer admission is valid.

VALID=*NO
The transfer admission is not valid. With this entry, users can be denied access to the profile.

USAGE=*PRIVATE
Access to your profile is denied for security reasons if someone with another user ID makes a repeated attempt to specify the same TRANSFER ADMISSION that you have already used.

USAGE=*PUBLIC
Access to your profile is not denied if another user happens to “discover” your TRANSFER-ADMISSION. “Discovery” means that another user ID attempted to specify the same TRANSFER ADMISSION twice. This is rejected for security reasons.

EXPIRATION-DATE=*NOT-RESTRICTED
The use of this transfer admission is not subject to a time restriction.

EXPIRATION-DATE=<date 8..10>
The use of the transfer admission is only possible until the given date. The entry must be made in the form YYYY-MM-DD or YY-MM-DD.

TRANSFER-ADMISSION=*SECRET
The system prompts you to enter the transfer admission; however, this does not appear on the screen. The operands VALID, USAGE and EXPIRATION-DATE can also be secretly entered in this case.

PRIVILEGED=*NO
The admission profile is not privileged.
FTP requests that are processed with a privileged admission profile are not subject to the restrictions set for MAX-ADM-LEVEL in the admission set. Only the FTAC administrator is allowed to assign a privileged status to profiles.

IGNORE-MAX-LEVELS=
With IGNORE-MAX-LEVELS, you can determine for which of the six basic functions the restrictions of the admission set should be ignored. The user’s MAX-USER-LEVELS can be exceeded in this way. The MAX-ADM-LEVELS in the admission set can only be effectively exceeded with an admission profile which has been designated as privileged by the FTAC administrator. The FTAC user can set up an admission profile for himself for special tasks (e.g. sending a certain file to a partner system with which he normally is not allowed to conduct an FTP transfer), which allows him to exceed the admission set. This profile must be explicitly assigned a privileged status by the FTAC administrator.
If you enter IGNORE-MAX-LEVELS=*YES, the settings for **all** the basic functions are ignored. If you wish to ignore the admission set for **specific** basic functions, you need to do this with the operands explained below.

IGNORE-MAX-LEVELS=*NO
FTP requests that are processed with the admission profile are subject to the restrictions of the admission set.

IGNORE-MAX-LEVELS=*YES
*YES allows you to communicate with partner systems whose security level exceeds the specifications of the admission set. If your profile does not have privileged status, you can only disregard the MAX-USER-LEVELS in the admission set, not the MAX-ADM-LEVELS. The current MAX-USER-LEVELS and MAX-ADM-LEVELS settings can be determined by using the command SHOW-FT-ADMISSION-SET (see example on page 261).

IGNORE-MAX-LEVELS=*PARAMETERS(...)
The following operands can be used to selectively deactivate the default settings for the individual basic functions.

INBOUND-SEND=*NO
The maximum security level that can be reached with the basic function “inbound send” is determined by the admission set.

INBOUND-SEND=*YES
For the basic function “inbound send”, you can use this admission profile to disregard the MAX-USER-LEVELS. If your profile is privileged, you are also not held to the restrictions of the MAX-ADM-LEVELS. In addition, the partial component “display file attributes” of the basic function “inbound file management” can be used.

INBOUND-RECEIVE=*NO
The maximum security level that can be reached with the basic function “inbound receive” is determined by the admission set.

INBOUND-RECEIVE=*YES
With this profile, you can disregard your settings for “inbound receive” in the MAX-USER-LEVELS. If your profile is privileged, you are also not held to the restrictions of the MAX-ADM-LEVELS. In addition, the following partial components of the basic function “inbound file management” can be used:
– delete files, as long as the file attributes are set accordingly,
– modify file attributes, if the basic function “inbound file management” was admitted in the admission set or in the admission profile.

INBOUND-MANAGEMENT=*NO
The maximum security level that can be reached with the basic function “inbound file management” is determined by the admission set.

INBOUND-MANAGEMENT=*YES
For the basic function “inbound file management”, you can use this admission profile to disregard the MAX-USER-LEVELS. If your profile is privileged, you are also not held to the restrictions of the MAX-ADM-LEVELS. The partial component “modify file attributes” of the basic function “inbound file management” only functions if the basic function “inbound receive” was permitted in the admission set or admission profile.

USER-ADMISSION=
With USER-ADMISSION, the user specifies the user ID under which the profile is to be saved. FTP requests that work with this admission profile access the given user ID in the local system.

USER-ADMISSION=*OWN
For USER-IDENTIFICATION and ACCOUNT, the specifications for your user ID and your account number are taken from your LOGON authorization. A BS2000 password is only taken from your LOGON authorization when an FTP request accesses the admission profile.

USER-ADMISSION=*PARAMETERS(...)
You can also enter the individual components of the user ID. This allows you to keep FTP requests which work with this admission profile under a different account number, for example. Alternatively, a password can be set in the admission profile. FTP requests which work with this admission profile will then only function if their current LOGON password corresponds to the preset password.

USER-IDENTIFICATION=*OWN / <name 1..8>
With USER-IDENTIFICATION, you enter your user ID in BS2000. Both entries have the same effect.

ACCOUNT=
With ACCOUNT, you enter the account number under which an FTP request is to be accounted when working with this admission profile.

ACCOUNT=*OWN
The account number is taken from your LOGON authorization.

ACCOUNT=<alphanum-name 1..8>
An FTP request should be accounted under the account number specified when it accesses this admission profile. You can enter any account number associated with your user ID.

PASSWORD=
With PASSWORD, you enter the BS2000 password associated with your user ID.

PASSWORD=*OWN
When an FTP request refers to this admission profile, FTAC uses the BS2000 password valid for your user ID at that moment. This prevents you from having to modify the admission profile if the BS2000 password is changed.

PASSWORD=*NONE
No BS2000 password is required for the user ID.

PASSWORD=<c-string 1..8> / <x-string 1..16>
When an FTP request accesses the admission profile, the password specified is compared with the current LOGON password. If the two do not correspond, the FTP request is rejected.

PASSWORD=*SECRET
The system prompts you to enter the password. The entry does not appear on the screen.

INITIATOR=*REMOTE
Since FTP requests are always treated as *REMOTE, the admission profiles for FTP must always have the *REMOTE setting. If the same profile is also used for openFT, the setting (*LOCAL,*REMOTE) would also be allowed, for example.

TRANSFER-DIRECTION=
With TRANSFER-DIRECTION, you determine which transfer direction may be used with this admission profile. The transfer direction is always seen from the viewpoint of the BS2000 FTP server on which the admission profile was defined.

TRANSFER-DIRECTION=*NOT-RESTRICTED
With this admission profile, data can be transferred from the client to the server, and vice versa.

TRANSFER-DIRECTION=*FROM-PARTNER
With this admission profile, data can only be transferred from the client to the server. It is not possible to display file attributes/directories (partial components of “inbound file management”), i.e. the following server commands are not permitted: *cdup*, *xcup*, *cwd*, *xcwd*, *list*, *mlsd*, *mlst*, *nlst*, *pwd*, *xpwd*, *retr*.

TRANSFER-DIRECTION=*TO-PARTNER
With this admission profile, data can only be transferred from the server to a client system. It is not possible to modify file attributes or delete files (partial components of “inbound file management”), i.e. the following server commands are not permitted: *appe*, *dele*, *site file*, *mkd*, *xmkd*, *rmd*, *xrmd*, *rnfr*, *stor, stou*.

PARTNER=
With PARTNER, you can specify that this admission profile is to be used only for FTP requests that are processed by a certain client system.

PARTNER=*NOT-RESTRICTED
The scope of this admission profile is not restricted to FTP requests with certain partner systems.

PARTNER=list-poss(50): <text 1..200 with-low>
The admission profile only permits those FTP requests that are processed with the specified client systems. A maximum of 50 client systems may be specified.
The total length of all the partners may not exceed 1000 characters. You may specify the name from the partner list or the address of the partner system, see also „openFT (BS2000) - Command Interface“. It is recommended, to use the name from the partner list. The format shown in the long form of the logging output provides an indication of how a partner address should be entered in an FTAC profile.

MAX-PARTNER-LEVEL=
With MAX-PARTNER-LEVEL, a maximum security level can be specified. In the case of FTP requests, the client system is assigned a security level specified by the system administrator or a default security level of 100.

MAX-PARTNER-LEVEL=*NOT-RESTRICTED
If FTP requests are processed with this admission profile, then the highest accessible security level is determined by the admission set.

MAX-PARTNER-LEVEL=<integer 0..100>
When you set a value for MAX-PARTNER-LEVEL that is less than the security level specified by the system administrator or the default value of 100, you (temporarily) prevent access to the admission profile for FTP requests.

FILE-NAME=
With FILE-NAME, you determine which files or library members under your user ID may be accessed by FTP requests that use this admission profile.

FILE-NAME=*NOT-RESTRICTED
The admission profile permits unrestricted access to all files of the user ID.

FILE-NAME=*EXPANSION
(PREFIX=<full-filename 1..53> / <partial-filename 2..53> /
<c-string 1..511 with-low>)
This entry can be used to restrict access to a number of files which all begin with the same prefix. If a file name is entered in an FTP request that uses this admission profile, FTAC places the prefix defined with EXPANSION in front of this file name. The FTP request is then permitted to access the file *PrefixFilename*.
It is not possible to switch between the POSIX and DMS file systems. If the prefix contains a "/" or begins with ".", only the POSIX file system can be accessed. In all other cases, only the DMS file system can be accessed.

Example

PREFIX=DAGOBERT.; an FTP request in which the file name BOURSE is specified accesses the file DAGOBERT.BOURSE.

Please note that the part of a DMS file name specified in the FTP command still has to be of the type <full-filename>.

FILE-PASSWORD=
With FILE-PASSWORD, you can enter a password for files into the admission profile. The FTAC functionality then only permits access to files which are protected with this password and to unprotected files. When a FILE-PASSWORD is specified in an admission profile, the password may no longer be specified in an FTP request which uses this admission profile. This allows you to grant users on remote systems access to certain files without having to divulge the file passwords.

FILE-PASSWORD=*NOT-RESTRICTED
The admission profile permits access to all files. If a password is set for a file, then it must be specified in the FTP request.

FILE-PASSWORD=*NONE
The admission profile only permits access to files without file passwords.

FILE-PASSWORD=<c-string 1..4> / <x-string 1..8> /
<integer -2147483648..2147483647>
The admission profile only permits access to files which are protected with the specified password and to unprotected files. The password which has already been specified in the profile may not be repeated in the FTP request.

FILE-PASSWORD=*SECRET
The system prompts you to enter the password. The entry does not appear on the screen.

WRITE-MODE=
With WRITE-MODE, you determine the write mode that applies to this FTP request. WRITE-MODE is only effective if the receive file is on the same system on which the admission profile was defined. In FTP commands, the write mode is not specified explicitly, but it is an implicit part of the FTP command:

appe	*EXTEND-FILE
stor, *rnfr*, *site file*, *dele*, *rmd*, *xrmd*	*REPLACE-FILE
stou	*NEW-FILE

WRITE-MODE=*NOT-RESTRICTED
In an FTP request which accesses this admission profile, all FTP write modes may be used without restrictions.

WRITE-MODE=*NEW-FILE
The *dele*, *rmd* and *xrmd* FTP commands are not permitted.

WRITE-MODE=*REPLACE-FILE
The *stou* FTP command is not permitted.

WRITE-MODE=*EXTEND-FILE
The *stor*, *stou*, *dele*, *rmd* and *xrmd* FTP commands are not permitted.

FT-FUNCTION=
This operand enables you to restrict the validity of the profile to certain FTP functions (=file transfer and file management functions).

FT-FUNCTION=*NOT-RESTRICTED
The full scope of FTP functions is available.

FT-FUNCTION=(*TRANSFER-FILE, *MODIFY-FILE-ATTRIBUTES, *READ-DIRECTORY, *FILE-PROCESSING)

The following functions are available:

***TRANSFER-FILE**
The admission profile may be used for the “transfer files”, “view file attributes” and “delete files” functions.

The following server commands are not permitted:

list, *nlist*, *pwd*, *xpwd*, *cwd*, *xcwd*, *cdup*, *xcup*, *rnfr*, *size*, *mdtm*

***MODIFY-FILE-ATTRIBUTES**
The admission profile may be used for the “view file attributes” and “modify file attributes” functions.

The following server commands are not permitted:

retr, *stor*, *appe*, *stou*, *dele*, *list*, *mlsd*, *mlst*, *nlist*, *pwd*, *xpwd*, *cwd*, *xcwd*, *cdup*, *xcup*, *size*, *mdtm*

***READ-DIRECTORY**
The admission profile may be used for the “view directories” and “view file attributes” functions.

The following server commands are not permitted:

retr, *stor*, *appe*, *stou*, *dele*

***FILE-PROCESSING**
The admission profile may be used for the “pre-processing” and “post-processing” file transfer function. The “transfer files” function must also be permitted.
The *FILE-PROCESSING specification is of relevance only for FTAC profiles without a filename prefix. Otherwise the first character of the filename prefix determines whether only normal data transfer (no pipe symbol |) or only pre-processing and post-processing (pipe symbol |) are to be possible with this FTAC profile.

Example

Dagobert Duck wants to create an admission profile for the following purpose:

Donald Duck, employee at the Duck Goldmine, has his own BS2000 system. He has to transfer monthly reports on a regular basis to his boss Dagobert‘s computer, DAGODUCK, using FTP. The file needs to have the prefix MONTHLYREPORTS. Since Dagobert’s admission set does not permit any “nbound” requests, he needs to assign the profile a privileged status (he is allowed to do this, since he is an FTAC administrator).

The command required to created such an admission profile is as follows:

```
/CREATE-FT-PROFILE NAME=GOLDMOBE,                                          -
/                  TRANSFER-ADMISSION='monthlyreportfortheboss',           -
/                  PRIVILEGED=*NO,                                         -
/                  IGNORE-MAX-LEVELS=*YES,                                 -
/                  TRANSFER-DIRECTION=*FROM-PARTNER,                       -
/                  FILE-NAME=*EXPANSION(PREFIX=MONTHLYREPORTS.),           -
/                  WRITE-MODE=*REPLACE-FILE
```

The short form of this command is:

```
/CRE-FT-PROF GOLDMOBE,TRANS-AD='monthlyreportfortheboss',                  -
/            PRIV=*YES,IGN-MAX-LEV=*YES,TRANS-DIR=*FROM,                   -
/            FILE-NAME=*EXP(PREF=MONTHLYREPORTS.),WRITE=*REPL
```

Donald Duck, who keeps the monthly report of Goldmine on his BS2000 system in the file NOTHINGBUTLIES, can use the following FTP command of the BS2000 FTP client to send it to the central computer DAGODUCK:

```
ftp> open DAGODUCK
Connected to DAGODUCK, port 21.
220 DAGODUCK FTP server ... ready.
Name (DAGODUCK:DONADUCK):

*$FTAC
331 Send your FTAC transfer admission as password
Password (DAGODUCK:$FTAC):

*monthlyreportfortheboss
230 $FTAC login ok, access restrictions apply.

ftp> put NOTHINGBUTLIES GOLDMINE
200 PORT command successful.
150 Opening data connection for GOLDMINE (139.25.24.2,4102).
22595 bytes sent in 0.06 seconds (3.6e+02 Kbytes/s)
226 Transfer complete.(SAM-IO)

ftp> bye
221 Goodbye.
```

Command return codes

(SC2)	SC1	Maincode	Meaning
0	0	FTC0051	A user ID with the same name already exists in the system.
0	0	FTC0056	Transfer admission is denied.
0	64	FTC0100	An FT profile with the same name already exists.
0	64	FTC0101	An FT profile with the same Transfer-Admission already exists.
0	64	FTC0150	The access password is missing.
0	64	FTC0153	The owner identification entered is not the own user ID.
0	64	FTC0157	No authorization to create the profile.
0	64	FTC0172	The specified User Admission does not exist in the system.
0	64	FTC0173	The specified Processing Admission does not exist in the system.
0	64	FTC0178	The specified partner name occurs several times.
0	64	FTC0182	The maximum length for partner names has been exceeded.
0	64	FTC0200	The total length of the two follow-up processing commands is too long.
0	64	FTC0255	A system error has occurred.

SC1/2 = Subcode 1/2 in decimal form
More information can be found in the manual “openFT - Command Interface”.

5.2.5 DELETE-FT-PROFILE - Delete admission profile

With the command DELETE-FT-PROFILE, you can delete all admission profiles for which you are the owner. You should go through your admission profiles on a regular basis and delete all those which are no longer required so that they don't take up memory.
With SHOW-FT-PROFILE (see from page 272), you can view the profiles and decide which ones are no longer needed.

If you call *HELP* for the SDF command syntax shown below, you may also see some operands that are not indicated here. This is because only the operands relevant for FTP are described in this section.

DELETE-**FT-PROF**ILE - showing operands relevant for FTP
NAME = ***ALL** / <alphanum-name 1..8> / *STD , **SEL**ECT-PARAMETER = ***OWN** / ***PAR**AMETERS(...) ***PAR**AMETERS(...) **TRANS**FER-**AD**MISSION = ***ALL** / ***NOT-SPEC**IFIED / <alphanum-name8..32> / <c-string 8..32 with-low> / <x-string 15..64> / ***SEC**RET , **OWN**ER-**ID**ENTIFICATION = ***OWN** / <name 1..8>

Operands

NAME=
With NAME, you can access the admission profile to be deleted by its name.

NAME=*ALL
With *ALL, you can delete all admission profiles. The FTAC user can delete all of his admission profiles with this operand, unless he selects a special profile with SELECT-PARAMETER.

NAME=<alphanum-name 1..8>
You want to delete the admission profile with the specified name.

NAME = *STD
Deletes the default admission profile for your own user ID.

SELECT-PARAMETER=
With SELECT-PARAMETER, you can enter selection criteria for the admission profiles to be deleted.

SELECT-PARAMETER=*OWN
With *OWN, you delete your own admission profiles.

SELECT-PARAMETER=*PARAMETERS(...)
With this structure, you can enter individual selection criteria.

TRANSFER-ADMISSION=
With TRANSFER-ADMISSION, you can use the transfer admission of an admission profile as a selection criterion for deletion.

TRANSFER-ADMISSION=*ALL
You want to delete admission profiles irrespective of the TRANSFER-ADMISSION.

TRANSFER-ADMISSION=*NOT-SPECIFIED
You want to delete admission profiles for which no transfer admission is specified.

TRANSFER-ADMISSION=<alphanum-name 8..32> /
<c-string 8..32 with-low> / <x-string 15..64>
You want to delete the admission profile that is accessed with this transfer admission. The FTAC user can only enter the transfer admissions of his/her own admission profiles.

TRANSFER-ADMISSION=*SECRET
The system prompts you to enter the transfer admission.The entry does not appear on the screen.

OWNER-IDENTIFICATION =*OWN / <name 1..8>
OWNER-IDENTIFICATION permits the FTAC user to delete his/her own admission profiles under this user ID. Both entries have the same effect.

Command return codes

(SC2)	SC1	Maincode	Meaning
1	0	FTC0053	No FT profile exists with these criteria.
0	64	FTC0150	The access password is missing.
0	64	FTC0153	The owner identification entered is not the own user ID.
0	64	FTC0255	A system error occurred.

SC1/2 = Subcode 1/2 in decimal form
More information can be found in the manual “openFT - Command Interface”.

5.2.6 MODIFY-FT-ADMISSION-SET - Modify admission set

The MODIFY-FT-ADMISSION-SET command can be used to modify the admission set for your user ID. You may access two components of the admission set:

- You can define a password to be entered for almost all subsequent FTAC commands (exception: the /SHOW... commands). This prevents other users who are working with your user ID from entering FTAC commands.

  It is not possible to have an FTAC password output. If an FTAC user forgets his/her FTAC password, only the FTAC administrator can delete or modify the password.

- You can modify the limiting values for the maximum security levels (the MAX-USER-LEVELS) that can be accessed from your user ID. The limiting values set by the FTAC administrator (MAX-ADM-LEVELS) cannot be disabled.
  The MAX-USER-LEVELS only work if they are higher (i.e. more restrictive) than the MAX-ADM-LEVELS.

If you call *HELP* for the SDF command syntax shown below, you may also see some operands that are not indicated here. This is because only the operands relevant for FTP are described in this section.

MODIFY-**FT-ADM**ISSION-SET - showing operands relevant for FTP

USER-IDENTIFICATION = ***OWN** / <alphanum-name 1...8>

, **SEL**ECT-PARAMETER = ***ALL**

, **MAX-LEV**ELS = ***UNCH**ANGED / ***STD** / <integer 0...100> / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

INBOUND-**SEND** = ***UNCH**ANGED / ***STD** / <integer 0...100>

, **IN**BOUND-**REC**EIVE = ***UNCH**ANGED / ***STD** / <integer 0...100>

, **IN**BOUND-**MAN**AGEMENT = ***UNCH**ANGED / ***STD** / <integer 0...100>

Operands

USER-IDENTIFICATION=
Specifies the user ID whose admission set is to be modified.

USER-IDENTIFICATION=*OWN
The admission set for the user ID under which you are currently working is to be modified.

USER-IDENTIFICATION=<alphanum-name 1..8>
The admission set for this user ID is to be modified. As an FTAC user, you can only enter your own user ID here.

SELECT-PARAMETER=*ALL
In later FTAC versions, it will be possible to enter additional selection criteria here.

MAX-LEVELS=
With this operand, you determine which security level(s) can be reached with which basic functions from the user ID of this admission set. You may optionally set one security level for all basic functions or different security levels for each basic function. The MAX-USER-LEVELS for this admission set are defined by the FTAC user; the MAX-ADM-LEVELS are defined by the FTAC administrator. FTAC runs authorization checks on the basis of the lowest specified security level in each case. FTAC users may reduce but not exceed the values set for them by the FTAC administrator.

MAX-LEVELS=*UNCHANGED
The security levels defined in this admission set are to remain unchanged.

MAX-LEVELS=*STD
With this setting, you specify that the values of the default admission set are to be used for for this admission set. This causes the admission set to be deleted from the admission file. This is possible even if the user ID has already been deleted.

MAX-LEVELS=<integer 0..100>
With this value, you can set a maximum security level for all basic functions. A value less than the security level specified by the system administrator or the default value of 100 means that no FTP access checked via FTAC is possible on this user ID until further notice (i.e. until the admission set is modified again).

MAX-LEVELS=*PARAMETERS(...)
With this structure, you can set a maximum security level for each of the basic functions. FTP partners always have the security level specified by the system administrator, which is 100 by default.

INBOUND-SEND=
Sets the maximum security level for the basic function “inbound send”. All partner systems with this security level or lower can request files from the owner of the admission set.

INBOUND-SEND=*UNCHANGED
The value for INBOUND-SEND remains unchanged.

INBOUND-SEND=*STD
For INBOUND-SEND, the value from the default admission set is used.

INBOUND-SEND=<integer 0..100>
For INBOUND-SEND, this maximum security level is entered in the admission set. A value less than the security level specified by the system administrator or the default value of 100 means that INBOUND-SEND is not possible on this ID. In this case, the *retr* server command is not permitted.

INBOUND-RECEIVE=
Sets the maximum security level for the basic function “inbound receive”. All partner systems with this security level or lower may send files to the owner of the admission set.

INBOUND-RECEIVE=*UNCHANGED
The value for INBOUND-RECEIVE remains unchanged.

INBOUND-RECEIVE=*STD
For INBOUND-RECEIVE, the value from the default admission set is used.

INBOUND-RECEIVE=<integer 0..100>
For INBOUND-RECEIVE, this maximum security level is entered in the admission set. A value less than the security level specified by the system administrator or the default value of 100 means that INBOUND-RECEIVE is not possible on this ID. In this case, the following server commands are not permitted: *stor*, *stou*, *appe*, *rnfr*, *dele*, *site file*.

INBOUND-MANAGEMENT=
Sets the maximum security level for the basic function “inbound file management”. All partner systems with this security level or lower may include the modification of file attributes and the querying of directories as part of their FTP request.

INBOUND-MANAGEMENT=*UNCHANGED
The value for INBOUND-MANAGEMENT remains unchanged.

INBOUND-MANAGEMENT=*STD
For INBOUND-MANAGEMENT, the value from the default admission set is used.

INBOUND-MANAGEMENT=<integer 0..100>
For INBOUND-MANAGEMENT, this maximum security level is entered in the admission set. A value less than the security level specified by the system administrator or the default value of 100 means that INBOUND-MANAGEMENT is not possible on this ID. In this case, the following server commands are not permitted: *cwd*, *xcwd*, *list*, *mlsd*, *mlst*, *nlst*, *mkd*, *xmkd*, *rmd*, *xrmd*, *pwd*, *xpwd*, *cdup*, *xcup*, *rnfr, size.*

Example

Donald needs information on his admission sets:

```
/SHOW-FT-ADMISSION-SET
```

Short form:

```
/SHOW-FT-ADM
```

He receives the following output:

```
   %                MAX. USER LEVELS              MAX. ADM LEVELS        ATTR
   % USER-ID  OBS  OBR  IBS  IBR  IBP  IBF  OBS  OBR  IBS  IBR  IBP  IBF
   % DUCKTAIL 100  100  100  100  100  100   80   80  100  100   80  100
```

Donald now wants to deny access to the basic function “inbound send” and thus prevent files from being read on his user ID.

```
/MODIFY-FT-ADMISSION-SET MAX-LEVELS=*PARAMETERS(INBOUND-SEND=99)
```

The short form of this command is:

```
/MOD-FT-ADM MAX-LEV=(IN-SEND=99)
```

To verify the new setting, Donald now displays his admission again:

```
/SHOW-FT-ADM
   %                MAX. USER LEVELS              MAX. ADM LEVELS        ATTR
   % USER-ID  OBS  OBR  IBS  IBR  IBP  IBF  OBS  OBR  IBS  IBR  IBP  IBF
   % DUCKTAIL 100  100   99  100  100  100   80   80  100  100   80  100
```

Command return codes

(SC2)	SC1	Maincode	Meaning
0	0	FTC0050	The set security level exceeds the administrator‘s limit and will remain invalid until the administrator´s limit is raised accordingly.
0	64	FTC0150	Authorization password is missing.
0	64	FTC0151	Only the administrator or owner is permitted to make this modification.
0	64	FTC0152	The user ID entered is not the user‘s own.
0	64	FTC0175	The operand “NEW-PASSWORD” may not be entered for *STD.
0	64	FTC0176	The user ID entered does not exist in the system.
0	64	FTC0255	A system error occurred.

SC1/2 = Subcode 1/2 in decimal form
More information can be found in the manual “openFT - Command Interface”.

5.2.7 MODIFY-FT-PROFILE - Modify admission profile

The command MODIFY-FT-PROFILE can be used by any FTAC user to modify his/her admission profile.

In a privileged admission profile, an FTAC user can only modify the operands TRANSFER-ADMISSION and PRIVILEGED.

As soon as an admission profile is modified, the timestamp is also updated. The timestamp is output with SHOW-FT-PROFILE INF=*ALL (LAST-MODIF). The timestamp is also updated if you do not change the properties of the profile, i.e. if you enter MODIFY-FTPROFILE with the parameter NAME, but no other parameters.

If you call *HELP* for the SDF command syntax shown below, you may also see some operands that are not indicated here. This is because only the operands relevant for FTP are described in this section.

MODIFY**-FT-PROF**ILE - showing operands relevant for FTP

NAME = ***ALL** / ***STD** / <alphanum-name 1..8>

,**SEL**ECT-PARAMETER = ***OWN** / ***PAR**AMETERS(...)

***PAR**AMETERS(...)
- **TRANSF**ER**-AD**MISSION = ***ALL** / ***NOT-SPEC**IFIED / <alphanum-name 8..32> / <c-string 8..32 with-low> / <x-string 15..64> / ***SEC**RET
- ,**OWN**ER**-ID**ENTIFICATION = ***OWN** / <name 1..8>

,**NEW-NAME** = ***OLD** / <alphanum-name 1..8>

,**TRANS**FER**-AD**MISSION = ***UNCH**ANGED / ***NOT-SPEC**IFIED / ***OLD-ADM**ISSION(...) / <alphanum-name 8..32>(...) / <c-string 8..32 with-low>(...) / <x-string 15..64>(...) / ***SEC**RET

***OLD-ADM**ISSION(...)

<alphanum-name 8..32>(...) / <c-string 8..32 with-low>(...) / <x-string 15..64>(...)
- **VAL**ID = ***YES** / ***NO** / ***UNCH**ANGED
- ,**USAGE** = ***PRIVATE** / ***PUB**LIC / ***UNC**HANGED
- ,**EXP**IRATION**-DATE** = ***NOT-REST**RICTED / <date 8..10> / ***UNCH**ANGED

,**PRIV**ILEGED = ***UNCH**ANGED / ***NO**

,**IGN**ORE**-MAX-LEV**ELS = ***UNCH**ANGED / ***NO** / ***YES** / ***PAR**AMETERS(...)

***PAR**AMETERS(...)
- ,**IN**BOUND**-SEND** = ***UNCH**ANGED / ***NO** / ***YES**
- ,**IN**BOUND**-REC**EIVE = ***UNCH**ANGED / ***NO** / ***YES**
- ,**IN**BOUND**-MAN**AGEMENT = ***UNCH**ANGED / ***NO** / ***YES**

```
,USER-ADMISSION = *UNCHANGED / *OWN / *PARAMETERS(...)

  *PARAMETERS(...)
    | USER-IDENTIFICATION = *OWN / <name 1..8>
    | ,ACCOUNT = *OWN / *NOT-SPECIFIED / <alphanum-name 1..8>
    | ,PASSWORD = *OWN / *NOT-SPECIFIED / <c-string 1..8> / <x-string 1..16> / *NONE / *SECRET

,INITIATOR = *UNCHANGED / *REMOTE

,TRANSFER-DIRECTION = *UNCHANGED / *NOT-RESTRICTED / FROM-PARTNER / TO-PARTNER

,PARTNER = *UNCHANGED / *NOT-RESTRICTED / *ADD(...) / *REMOVE(...) /
                        list-poss(50): <text 1..200 with-low>

  *ADD(...)
    | NAME = list-poss(50): <text 1..200 with-low>

  *REMOVE(...)
    | NAME = list-poss(50): <text 1..200 with-low>

,MAX-PARTNER-LEVEL = *UNCHANGED / *NOT-RESTRICTED / <integer 0..100>

,FILE-NAME = *UNCHANGED / *NOT-RESTRICTED / *EXPANSION(...)

  *EXPANSION(...)
    | PREFIX = <full-filename 1..53> / <partial-filename 2..53> / <c-string 1..511 with-low>

,FILE-PASSWORD = *UNCHANGED / *NOT-RESTRICTED / *NONE / <c-string 1...4> / <x-string 1...8> /
                        <integer -2147483648...2147483647> / *SECRET

,WRITE-MODE = *UNCHANGED / *NOT-RESTRICTED / *NEW-FILE / *REPLACE-FILE / *EXTEND-FILE

,FT-FUNCTION = *UNCHANGED / *NOT-RESTRICTED / list-poss(4):
                      *TRANSFER-FILE / *MODIFY-FILE-ATTRIBUTES / *READ-DIRECTORY /
                        *FILE-PROCESSING
```

Operands

NAME=
With NAME, you determine the name of the admission profile to be modified.

NAME=*ALL
Use this to modify all your admission profiles at the same time.

NAME=<alphanum-name 1..8>
Use this to modify the admission profile with this name.

NAME = *STD
Changes the default admission profile for your user ID.

SELECT-PARAMETER=
With SELECT-PARAMETER, you can specify a transfer admission. You then modify the admission profile that is accessed with this transfer admission.

SELECT-PARAMETER=*OWN
Use this to modify your own admission profile.

SELECT-PARAMETER=*PARAMETERS(...)
With this structure, you can specify the selection criteria for the profiles you want to modify.

TRANSFER-ADMISSION=
Entering the TRANSFER-ADMISSION here makes it a selection criterion for the admission profiles that you want to modify.

TRANSFER-ADMISSION=*ALL
All your admission profiles are modified, irrespective of the transfer admission.

TRANSFER-ADMISSION=*NOT-SPECIFIED
Only admission profiles without a defined transfer admission are to be modified.

TRANSFER-ADMISSION=<alphanum-name 8..32> /
<c-string 8..32 with-low> / <x-string 15..64>
The admission profile with this transfer admission is to be modified.

TRANSFER-ADMISSION=*SECRET
The system prompts you to enter the transfer admission, but does not display your entry on the screen.

OWNER-IDENTIFICATION=*OWN / <name 1..8>
OWNER-IDENTIFICATION permits the FTAC user to modify his/her own admission profile. Both entries have the same effect.

NEW-NAME=
With NEW-NAME, you assign your admission profile a new name (or not).

NEW-NAME=*OLD
The name of the admission profile remains unchanged.

NEW-NAME=<alphanum-name 1..8>
This is the new name of the admission profile. This name must be unique among all the admission profiles on your user ID. If an admission profile with this name already exists, FTAC rejects the command with the following message:

```
FTC0100 FT profile already exists
```

The command SHOW-FT-PROFILE (see from page 272) can be used to obtain information on the already assigned names. It is sufficient to enter SHOW-FT-PROFILE without operands for this information.

TRANSFER-ADMISSION=
With TRANSFER-ADMISSION, you can modify the transfer admission associated with an admission profile. You must ensure that the transfer admission is unique within your openFT (BS2000) system. If the transfer admission you have selected already exists, FTAC rejects the command with the following message:

```
FTC0101 Transfer admission already exists
```

TRANSFER-ADMISSION=*UNCHANGED
The transfer admission remains unchanged.

TRANSFER-ADMISSION=*NOT-SPECIFIED
No transfer admission is set, and existing transfer admissions, if any, are invalidated. This blocks the profile.

TRANSFER-ADMISSION=*OLD-ADMISSION(...)
The transfer admission itself remains unchanged, but the options may be changed (in contrast to the entry TRANSFER-ADMISSION=*UNCHANGED). A description of the values in brackets (VALID=, USAGE= and EXPIRATION-DATE=) is given below.

TRANSFER-ADMISSION=<alphanum-name 8..32>(...) /
<c-string 8..32 with-low>(...) / <x-string 15..64>(...)
The character string must be entered as a transfer admission in the FTP request. The alphanumeric input is always stored in lowercase letters.

VALID=*UNCHANGED
The value remains unchanged.

VALID=*YES
The transfer admission is valid.

VALID=*NO
The transfer admission is not valid. The profile can be blocked with this entry.

USAGE=*UNCHANGED
The value remains unchanged.

USAGE=*PRIVATE
Access to your profile is denied for security reasons whenever another user ID makes a repeated attempt to specify the same TRANSFER ADMISSION that you have already used.

USAGE=*PUBLIC
Access to your profile is not denied if another user happens to “discover” your TRANSFER-ADMISSION. “Discovery” means that another user ID tried to specify the same TRANSFER ADMISSION twice. This is rejected for security reasons.

EXPIRATION-DATE=*UNCHANGED
The value remains unchanged.

EXPIRATION-DATE=*NOT-RESTRICTED
The use of this transfer admission is not subject to a time restriction.

EXPIRATION-DATE=<date 8..10>
The use of the transfer admission is only possible until the given date. The entry must be made in the form YYYY-MM-DD or YY-MM-DD.

TRANSFER-ADMISSION=*SECRET
The system prompts you to enter the transfer admission, but does not display your entry on the screen. The operands VALID, USAGE and EXPIRATION-DATE can also be secretly entered in this case.

PRIVILEGED=
With PRIVILEGED, the FTAC administrator can privilege the admission profile of any FTAC user. FTP requests that are processed with a privileged status are not subject to the restrictions for MAX-ADM-LEVEL in the admission set.
The FTAC user can only revoke an assigned privileged status.

PRIVILEGED=*UNCHANGED
The status of this admission profile remains unchanged.

PRIVILEGED=*NO
With *NO, you can revoke the privileged status.

IGNORE-MAX-LEVELS=
With IGNORE-MAX-LEVELS, you can determine for which of the six basic functions the restrictions of the admission set should be ignored. The MAX-ADM-LEVELS in the admission set can only be effectively exceeded with an admission profile that has been designated as privileged by the FTAC administrator. The FTAC user can set up an admission profile for himself/herself for special tasks (e.g. sending a certain file to a partner system with which he/she is normally not allowed to conduct FTP), which allows him/her to exceed the admission set. This profile must be explicitly assigned a privileged status by the FTAC administrator.
If you enter IGNORE-MAX-LEVELS=*YES, the settings for all the basic functions are ignored. If you want to ignore the admission set for specific basic functions, you will need to do this with the operands explained further below.

IGNORE-MAX-LEVELS=*UNCHANGED
With this admission profile, you can access the same security levels as before the modification (unless you have revoked the privileged status with PRIVILEGED=*NO).

IGNORE-MAX-LEVELS=*NO
FTP requests that are processed with the admission profile are subject to the restrictions of the admission set.

IGNORE-MAX-LEVELS=*YES
*YES allows you to communicate with partner systems whose security level exceeds the specifications of the admission set. If your profile does not have privileged status, you can disregard only the MAX-USER-LEVELS in the admission set, but not the MAX-ADM-LEVELS.
The current MAX-USER-LEVELS and MAX-ADM-LEVELS settings can be determined by using the command SHOW-FT-ADMISSION-SET (see example on page 261).

IGNORE-MAX-LEVELS=*PARAMETERS(...)

INBOUND-SEND=*UNCHANGED
The maximum security level that can be reached with the basic function “inbound send” remains unchanged.

INBOUND-SEND=*NO
The maximum security level that can be reached with the basic function “inbound send” is determined by the admission set.

INBOUND-SEND=*YES
For the basic function “inbound send”, you can use this admission profile to disregard the MAX-USER-LEVELS. If your profile is privileged, you are also not held to the restrictions of the MAX-ADM-LEVELS. In addition, the partial component “display file attributes” of the basic function “inbound file management” can be used.

INBOUND-RECEIVE=*UNCHANGED
The maximum security level that can be reached with the basic function “inbound receive” remains unchanged.

INBOUND-RECEIVE=*NO
The maximum security level that can be reached with the basic function “inbound receive” is determined by the admission set.

INBOUND-RECEIVE=*YES
With this profile, you can disregard your settings for “inbound receive” in the MAX-USER-LEVELS. If your profile is privileged, you are also not held to the restrictions of the MAX-ADM-LEVELS. In addition, the following partial components of the basic function “inbound file management” can be used:
– delete files, as long as the file attributes are set accordingly,
– modify file attributes, if the basic function “inbound file management” was admitted in the admission set or in the admission profile.

INBOUND-MANAGEMENT=*UNCHANGED
The maximum security level that can be reached with the basic function “inbound file management” remains unchanged.

INBOUND-MANAGEMENT=*NO
The maximum security level that can be reached with the basic function “inbound file management” is determined by the admission set.

INBOUND-MANAGEMENT=*YES
For the basic function “inbound file management”, you can use this admission profile to disregard the MAX-USER-LEVELS. If your profile is privileged, you are also not held to the restrictions of the MAX-ADM-LEVELS. The partial component “modify file attributes” of the basic function “inbound file management” only works if the basic function “inbound receive” was permitted in the admission set or admission profile.

USER-ADMISSION=
With USER-ADMISSION, you enter the user ID under which the modified profile is to be saved. FTP requests using this profile access the user ID entered on the local system. If the FTAC administrator has created an admission profile for a user, the user can enter the ACCOUNT and PASSWORD in the operand USER-ADMISSION by using the command MODIFY-FT-PROFILE (since he/she is the only person who should know them) before the profile is used (see command CREATE-FT-PROFILE, page 230).

USER-ADMISSION=*UNCHANGED
The USER-ADMISSION of this admission profile remains unchanged.

USER-ADMISSION=*OWN
For USER-IDENTIFICATION and ACCOUNT, the specifications for your user ID and your account number are taken from your LOGON authorization. A BS2000 password is only taken from your LOGON authorization when an FTP request accesses the admission profile.

USER-ADMISSION=*PARAMETERS(...)
You can also enter the individual components of the user ID. This allows you to keep FTP requests which use this admission profile under a different account number, for example. Alternatively, a password can be set in the admission profile. FTP requests which use this admission profile will then only function if their current LOGON password corresponds to the preset password.

USER-IDENTIFICATION=
USER-IDENTIFICATION identifies your user ID in BS2000.

USER-IDENTIFICATION=***OWN**
Your user ID is taken from your LOGON authorization.

USER-IDENTIFICATION=<name 1..8>
<name 1..8> is the user ID with which the profile is to be associated.

ACCOUNT=
With ACCOUNT, you enter the account number under which an FTP request is to be accounted when working with this admission profile.

ACCOUNT=*OWN
The account number is taken from your LOGON authorization.

ACCOUNT=*NOT-SPECIFIED
The account number is to be specified by the owner of the admission profile. This function enables the FTAC administrator to set up profiles for user IDs without knowing the corresponding account numbers.

ACCOUNT=<alphanum-name 1..8>
With ACCOUNT, you specify the account number under which an FTP request is to be accounted when it accesses this admission profile. You can specify any account number associated with the user ID.

PASSWORD=
With PASSWORD, you enter the BS2000 password associated with your user ID.

PASSWORD=*OWN
When an FTP request refers to this admission profile, FTAC uses the BS2000 password valid for your user ID at that moment. This prevents you from having to modify the admission profile if the BS2000 password is changed.

PASSWORD=*NOT-SPECIFIED
The password is first specified by the owner of the admission profile. This function permits the FTAC administrator to set up profiles for foreign user IDs.

PASSWORD=*NONE
No BS2000 password is required for the user ID.

PASSWORD=<c-string 1..8> / <x-string 1..16>
When an FTP request accesses the admission profile, the specified password is compared with the current LOGON password. If the two do not correspond, the FTP request is rejected.

PASSWORD=*SECRET
The system prompts you to enter the password. The entry does not appear on the screen.

INITIATOR=
With INITIATOR, you determine if initiators from local and/or remote systems are permitted to use this admission profile for their FTP requests.

INITIATOR=*UNCHANGED
The settings in this admission profile remain unchanged.

INITIATOR=*REMOTE
Since FTP requests are always treated as *REMOTE, the admission profiles for FTP must always have the *REMOTE setting. If the same profile is also used for openFT, the setting (*LOCAL,*REMOTE) would also be allowed, for example.

TRANSFER-DIRECTION=
With TRANSFER-DIRECTION, you determine which transfer direction may be used with this admission profile. The transfer direction is always seen from the viewpoint of the BS2000 FTP server on which the admission profile was defined.

TRANSFER-DIRECTION=*UNCHANGED
The settings in this admission profile remain unchanged.

TRANSFER-DIRECTION=*NOT-RESTRICTED
With this admission profile, data can be transferred from the client to the server, and vice versa.

TRANSFER-DIRECTION=*FROM-PARTNER
With this admission profile, data can only be transferred from the client to the server. It is not possible to display file attributes/directories (partial components of “inbound file management”), i.e. the following server commands are not permitted: *cdup*, *xcup*, *cwd*, *xcwd*, *list*, *mlsd*, *mlst*, *nlst*, *pwd*, *xpwd*, *retr*, *size*, *mdtm*.

TRANSFER-DIRECTION=*TO-PARTNER
With this admission profile, data can only be transferred from the server to a client system. It is not possible to modify file attributes or delete files (partial components of “inbound file management”), i.e. the following server commands are not permitted: *appe*, *dele*, *site file*, *mkd*, *xmkd*, *rmd*, *xrmd*, *rnfr*, *stor*, *stou*.

PARTNER=
With PARTNER, you can specify that this admission profile is to be used only for FTP requests that are processed by a certain client system.

PARTNER=*UNCHANGED
Any existing PARTNER in the admission profile remains unchanged.

PARTNER=*NOT-RESTRICTED
The scope of this admission profile is not restricted to FTP requests with certain partner systems.

PARTNER=list-poss(50): <text 1..200 with-low>
The admission profile only permits those FTP requests which are processed with the specified client systems. A maximum of 50 client systems can be specified. You may specify the name from the partner list or the address of the partner system, see also „openFT (BS2000) - Command Interface“. It is recommended, to use the name from the partner list.

PARTNER=*ADD(list-poss(50): <text 1..200 with-low>)
With this specification, you can add elements to an existing list of partner systems. A maximum of 50 client systems can be specified.

PARTNER=*REMOVE(list-poss(50): <text 1..200 with-low>)
With this specification, you can remove elements from an existing list of partner systems. A maximum of 50 client systems can be specified.

MAX-PARTNER-LEVEL=
With MAX-PARTNER-LEVEL, a maximum security level can be specified. In the case of FTP requests, the client system is always assigned a security level specified by the system administrator or the default security level of 100. MAX-PARTNER-LEVEL works in conjunction with the admission set. When non-privileged admission profiles are used, the access check is executed on the basis of the smallest specified value.

MAX-PARTNER-LEVEL=*UNCHANGED
The specification for MAX-PARTNER-LEVEL in this admission set remains unchanged.

MAX-PARTNER-LEVEL=*NOT-RESTRICTED
If FTP requests are processed with this admission profile, then the highest accessible security level is determined by the admission set.

MAX-PARTNER-LEVEL=<integer 0..100>
When you set a value for MAX-PARTNER-LEVEL that is less than the security level specified by the system administrator or the default value of 100, you (temporarily) prevent access to the admission profile for FTP requests.

FILE-NAME=
With FILE-NAME, you determine which files under your user ID may be accessed by FTP requests that use this admission profile.

FILE-NAME=*UNCHANGED
The specifications for FILE-NAME in this admission profile remain unchanged.

FILE-NAME=*NOT-RESTRICTED
The admission profile permits unrestricted access to all files of the user ID.

FILE-NAME =*EXPANSION(PREFIX = <filename 1..53> / <partial-filename 2..53> / <c-string 1..511 with-low>)
This entry can be used to restrict access to a number of files which all begin with the same prefix. If a file name is entered in an FTP request which uses this admission profile, depending on the currently set working directory, FTAC places the prefix defined with EXPANSION before this file name. The FTP request is then permitted to access the file *PrefixFilename*.
It is not possible to switch between the POSIX and DMS file systems. If the prefix contains a “/” or begins with “.”, only the POSIX file system can be accessed. In all other cases, only the DMS file system can be accessed.

Example

PREFIX=DAGOBERT.; an FTP request in which the file name BOURSE is specified accesses the file DAGOBERT.BOURSE.

Please note that the part of a DMS file name which is specified in the FTP command still has to be of the type <full-filename>.

FILE-PASSWORD=
With FILE-PASSWORD, you can enter a password for files into the admission profile. The FTAC functionality then only permits access to files which are protected with this password and to unprotected files. When a FILE-PASSWORD is specified in an admission profile, the password may no longer be specified in an FTP request which uses this admission profile. This allows you to grant users on remote systems access to certain files without having to divulge the file passwords.

FILE-PASSWORD=*UNCHANGED
The specifications for FILE-PASSWORD in this admission profile remain unchanged.

FILE-PASSWORD=*NOT-RESTRICTED
The admission profile permits access to all files. If a password is set for a file, then it must be specified in the FTP request.

FILE-PASSWORD=*NONE
The admission profile only permits access to files without file passwords.

FILE-PASSWORD=<c-string 1..4> / <x-string 1..8> /
<integer -2147483648..2147483647>
The admission profile only permits access to files which are protected with the specified password and to unprotected files. The password which has already been specified in the profile may not be repeated in the FTP request.

FILE-PASSWORD=*SECRET
The system prompts you to enter the password. The entry does not appear on the screen.

WRITE-MODE=
With WRITE-MODE, you determine the write mode that applies to this FTP request. WRITE-MODE is only effective if the receive file is on the same system on which the admission profile was defined. In FTP commands, the write mode is not specified explicitly, but it is an implicit part of the FTP command:

appe	*EXTEND-FILE
stor, *rnfr*, *site file*, *dele*, *rmd*, *xrmd*	*REPLACE-FILE
stou	*NEW-FILE

WRITE-MODE=*UNCHANGED
The specifications for WRITE-MODE in this admission profile remain unchanged.

WRITE-MODE=*NOT-RESTRICTED
In an FTP request which accesses this admission profile, all FTP write modes may be used without restrictions.

WRITE-MODE=*NEW-FILE
The *dele*, *rmd* and *xrmd* FTP commands are not permitted.

WRITE-MODE=*REPLACE-FILE
The *stou* FTP command is not permitted.

WRITE-MODE=*EXTEND-FILE
The *stor*, *stou*, *dele*, *rmd* and *xrmd* FTP commands are not permitted.

FT-FUNCTION=
This operand enables you to restrict the validity of the profile to certain FTP functions (=file transfer and file management functions).

FT-FUNCTION=*UNCHANGED
The existing scope of file management functions remains unchanged.

FT-FUNCTION=*NOT-RESTRICTED
The full scope of FTP functions is available.

FT-FUNCTION=(*TRANSFER-FILE, *MODIFY-FILE-ATTRIBUTES,
***READ-DIRECTORY, *FILE-PROCESSING)**

The following functions are available:

***TRANSFER-FILE**
The admission profile may be used for the “transfer files”, “view file attributes” and “delete files” functions.
The following server commands are not permitted:

list, *mlsd*, *mlst*, *nlist*, *pwd*, *xpwd*, *cwd*, *xcwd*, *cdup*, *xcup*, *rnfr*, *size*, *mdtm*

***MODIFY-FILE-ATTRIBUTES**
The admission profile may be used for the “view file attributes” and “modify file attributes” functions.
The following server commands are not permitted:

retr, *stor*, *appe*, *stou*, *dele*, *list*, *mlsd*, *mlst*, *nlist*, *pwd*, *xpwd*, *cwd*, *xcwd*, *cdup*, *xcup*, *size*, *mdtm*

***READ-DIRECTORY**
The admission profile may be used for the “view directories” and “view file attributes” functions.
The following server commands are not permitted:

retr, *stor*, *appe*, *stou*, *dele*, *rnfr.*

***FILE-PROCESSING**
The admission profile may be used for the “pre-processing” and “post-processing” file transfer functions. The “transfer files” function must also be permitted.
The *FILE-PROCESSING specification is of relevance only for FTAC profiles without a filename prefix. Otherwise the first character of the filename prefix determines whether only normal data transfer (no pipe symbol “|”) or only pre- and post-processing (pipe symbol “|”) are to be possible with this FTAC profile

Example

After Donald Duck has created an admission profile with the name *profile1*, which grants other users access to his user ID with the LOGON authorization, he decides to restrict this profile so that only FTP accesses to files which begin with the prefix *BRANCH* are possible.

The required command is:

```
/MODIFY-FT-PROFILE␣NAME=profile1,
                      FILE-NAME=*EXPANSION(PREFIX=branch.)
```

A possible short form of this command is:

```
/MOD-FT-PROF␣profile1,FILE-N=(PRE=branch.)
```

This places heavy restrictions on the admission profile. The other specifications remain unchanged.

Command return codes

(SC2)	SC1	Maincode	Meaning
0	0	FTC0051	A user ID with the same name already exists in the system.
0	64	FTC0053	No FT profile exists which meets the criteria specified.
0	64	FTC0055	The partner restrictions were lifted.
0	0	FTC0056	Transfer admission is blocked.
0	64	FTC0100	An FT profile with this name already exists.
0	64	FTC0101	An FT profile with this transfer admission already exists.
0	64	FTC0150	The access password is missing.
0	64	FTC0151	Modifications can only be made by the administrator or owner.
0	64	FTC0153	The owner ID entered is not the user´s own ID.
0	64	FTC0170	The partner entered is unknown within the partner system available for this user.
0	64	FTC0171	The profile entered does not exist.
0	64	FTC0172	The user admission entered does not exist in the system.
0	64	FTC0173	The processing admission entered does not exist in the system.
0	64	FTC0174	The parameters “NEW-NAME” and “TRANSFER-ADMISSION” may only used together in conjunction with unique selection criteria (“NAME” or “TRANSFER-ADMISSION”).
0	64	FTC0178	The partner name entered occurs several times.
0	64	FTC0179	The maximum number of partner restrictions has been exceeded.
0	64	FTC0182	The maximum length of partner names has been exceeded.
0	64	FTC0200	The total length of the two follow-up processing commands is too long.
0	64	FTC0255	A system error has occurred.

SC1/2 = Subcode 1/2 in decimal form
More information can be found in the manual “openFT - Command Interface”.

5.2.8 SHOW-FT-ADMISSION-SET - Display admission sets

The command SHOW-FT-ADMISSION-SET is used to view admission sets. You can optionally have the following information output to SYSOUT or SYSLST:

- whether the admission set is privileged (if it is, then you are the FTAC administrator).
- whether a password is required to use FTAC commands on this user ID. The password itself is not displayed.
- the limiting values that were set by the owner of this user ID for the accessible security levels.
- the limiting values that were preset by the FTAC administrator for the accessible security levels.

If you call *HELP* for the SDF command syntax shown below, you may also see some operands that are not indicated here. This is because only the operands relevant for FTP are described in this section.

SHOW-**FT-ADM**ISSION-**S**ET - showing operands relevant for FTP
USER-IDENTIFICATION = ***OWN** / ***ALL** / ***STD** / <alphanum-name 1...8> , **OUT**PUT = ***SYSOUT (LAYOUT = *STD** / ***CSV)** / ***SYSLST (LAYOUT = *STD** / ***CSV)**

Operands

USER-IDENTIFICATION=
The user ID for which you want to view the admission set. FTAC users can only obtain information on their own admission sets and the default admission set.

USER-IDENTIFICATION=*OWN
FTAC shows the admission set of your own user ID.

USER-IDENTIFICATION=*ALL
FTAC shows the default admission set and the admission set of your own user ID.

USER-IDENTIFICATION=*STD
FTAC only shows the default admission set.

USER-IDENTIFICATION=<alphanum-name 1..8>
FTAC shows the admission set for this user ID. The operand stands for the USER-ID of the specified user ID. FTAC users may only enter their own user IDs here.

OUTPUT=
Determines the output medium for the information requested.

OUTPUT=*SYSOUT(...)
The output is sent to SYSOUT.

OUTPUT=*SYSLST(...)
The output is sent to SYSLST.

LAYOUT=*STD
The output is formatted using a standard layout that can be easily read by the user.

LAYOUT=*CSV
The output is supplied in CSV (Comma Separated Value) format. This is a widely used tabular format, especially in the PC environment, in which individual fields are separated by a delimiter, which is usually a semicolon ";".

Example

Dagobert Duck, the FTAC administrator of the Duck Bank, wants to obtain information on the admission sets in his system. He enters the command

```
/SHOW-FT-ADMISSION-SET␣USER-IDENTIFICATION=*ALL
```

Short form:

```
/SHOW-FT-AD␣*ALL
```

and receives the following output:

```
%                MAX. USER LEVELS              MAX. ADM LEVELS        ATTR
% USER-ID  OBS  OBR  IBS  IBR  IBP  IBF  OBS  OBR  IBS  IBR  IBP  IBF
% *STD      10   10  100  100    0    0   10   10  100  100    0    0
% DAGOBERT 100  100    0   99    0*   0* 100  100    0   99    0*   0* PRIV
% DAISY     50   50   99  100   50   50   50   50  100  100   50   50  PW
% DANIEL     0   10   99   99    0    0   10   10  100  100    0    0  PW
% DONALD    50  100   99  100*   0    0   50  100  100  100    0    0
```

These can be explained as follows:

The user ID of each admission set is in the column USER-ID. In this example, there is a default admission set as well as admission sets for the user IDs DAGOBERT, DAISY, DANIEL and DONALD.

The ATTR column shows the privileged admission set. We can see that DAGOBERT is the FTAC administrator.

The ATTR column also indicates whether an FTAC password has been defined (with PW). DAGOBERT, DAISY and DANIEL have done this to prevent others from issuing any FTAC commands on their user ID that could result in changes.

The six columns under MAX-USER-LEVELS show the limiting values which the FTAC users have set for their admission sets. The six columns under MAX-ADM-LEVELS show the
limiting values set by the FTAC administrator. The smaller of the two values indicates up to which security level the owner of the admission set may use each basic function. The basic functions are abbreviated in the output as follows:

IBS = INBOUND-SEND

IBR = INBOUND-RECEIVE

IBP = INBOUND-PROCESSING

IBF = INBOUND-FILEMANAGEMENT

To begin with, it must be observed that FTP users (regardless of the security level assigned to their system) always receive a security level specified by the system administrator or the default security level of 100.

The default admission set has been set up to permit FTP users to send files to the FTP server and to retrieve files from the FTP server, but not to perform any file management actions.

DAGOBERT does not permit any file transfer accesses from outside under his user ID (IBS=0, IBR=99, IBP=0).

The files of the user ID DAISY may not be read using FTP (IBS=99), but files may be transferred to this user ID (IBR=100). The user ID DANIEL cannot be accessed with FTP (IBS, IBR and IBF are less than 100).

Command return codes

(SC2)	SC1	Maincode	Meaning
0	64	FTC0052	The information output was interrupted.
0	64	FTC0152	The user ID entered is not the user´s own ID.
0	64	FTC0181	The entered FT profile name occurs several times.
0	64	FTC0255	A system error occurred.

SC1/2 = Subcode 1/2 in decimal form
More information can be found in the manual “openFT - Command Interface”.

5.2.9 SHOW-FT-LOGGING-RECORDS - Display logging records

With FTAC functionality, you can use SHOW-FT-LOGGING-RECORDS to also display the FTAC log records of your own user ID. Even FT administrators can only display those FTAC log records which refer to their own user IDs. The FTAC administrator is the only user who can display all FTAC log records on the system.

If you call *HELP* for the SDF command syntax shown below, you may also see some operands that are not indicated here. This is because only the operands relevant for FTP are described in this section.

SHOW-FT-LOGGING-**REC**ORDS - showing operands relevant for FTP

```
SELECT = *OWN / *ALL / *PARAMETERS(...)

  *PARAMETERS(...)
    LOGGING-ID = *ALL / <alphanum-name 1..12> / *INTERVAL(...)
      *INTERVAL(...)
        FROM = 1 / <alphanum-name 1..12>
        ,TO = *HIGHEST-EXISTING / <alphanum-name 1..12>
    ,OWNER-IDENTIFICATION = *OWN / *ALL / <name 1..8>
    ,CREATION-TIME = *INTERVAL (...) / *DAYS(...)
      *INTERVAL(...)
        FROM = 1970-01-01 (...) / <date 8..10> (...)
          (...)
            TIME = 00:00 / <time 1..8>
        ,TO = *TOMORROW (...) / *TODAY (...) / <date 8..10> (...)
          (...)
            TIME = 00:00 / <time 1..8>
      *DAYS(...)
        NUMBER = <integer 1..1000>
    ,RECORD-TYPE = *ALL / *PARAMETERS(...)
      *PARAMETERS(...)
        ,FTAC = (*TRANSFER-FILE,*READ-FILE-ATTRIBUTES,*DELETE-FILE,
                 *CREATE-FILE,*MODIFY-FILE-ATTRIBUTES,
                 *READ-DIRECTORY,*MOVE-FILE,*CREATE-DIRECTORY,
                 *DELETE-DIRECTORY,*MODIFY-DIRECTORY. *LOGIN) / *NONE /
                 list-poss(11): *TRANSFER-FILE / *READ-FILE-ATTRIBUTES / *DELETE-FILE /
                 *CREATE-FILE / *MODIFY-FILE-ATTRIBUTES / *READ-DIRECTORY /
                 *MOVE-FILE / *CREATE-DIRECTORY / *DELETE-DIRECTORY /
                 *MODIFY-DIRECTORY / *LOGIN
```

```
        ,INITIATOR = (*LOCAL,*REMOTE) / *REMOTE
        ,PARTNER-NAME = *ALL / <text 1..200 with-low>
        ,FILE-NAME = *ALL / <filename 1..54> / <filename-prefix 2..53> /
                     <c-string 1..512 with-low> / *DIRECTORY(...) / *POSIX(NAME=<posix-pathname 1..510>)
          *DIRECTORY(...)
            NAME = *ALL / <partial-filename 1..53> / <c-string 1..512 with-low>
        ,REASON-CODE = *ALL / *FAILURE / <text 1..4>
        ,TRANSFER-ID = *ALL / <integer 1.. 2147483639>
,NUMBER = 1 / *ALL / <integer 1..99999999>
,INFORMATION = *STD / *ALL
,OUTPUT = *SYSOUT (LAYOUT = *STD / *CSV ) /
          *SYSLST (LAYOUT = *STD / *CSV)
```

Operands

SELECT=
Used to select a group of log records.

SELECT=*OWN /*ALL
Selects log records under the user's own user ID. If no other selection criteria are specified, the most recent log record under the user's own user ID is displayed. The entries *OWN and *ALL generate the same output.

SELECT=*PARAMETERS(...)

LOGGING-ID=
Specifies the number of the log record.

LOGGING-ID=*ALL
The number of the log record is not a selection criterion.

LOGGING-ID=<alphanum-name 1..12>
Number of the log record to be output. The value range for the logging ID is from 1 through 999999999999.

LOGGING-ID=*INTERVAL(...)
Range of log records to be output.

FROM=<alphanum-name 1..12>
First log record to be output. The value range for the logging ID is from 1 through 999999999999.

TO=*HIGHEST-EXISTING / <alphanum-name 1..12>
Last log record to be output. The value range for the logging ID is from 1 through 999999999999.

OWNER-IDENTIFICATION=
Specifies the user ID of the user whose log records are to be displayed.

OWNER-IDENTIFICATION=*OWN
Shows the log records of the user's own ID.

OWNER-IDENTIFICATION=*ALL
Shows the log records of all user IDs. This operand can be used by the FTAC administrator to display the FTAC log records for all user IDs.
Normal FTP users receive information only on the log records of their own user IDs even if they select this entry.

OWNER-IDENTIFICATION=<name 1..8>
Any user ID for which the log records are to be output.
Normal FT users may only specify their own user IDs.

CREATION-TIME=*INTERVAL(...)
Selects a range of log records to be output by specifying a creation date.

FROM=1970-01-01(...) / < date 8..10>(...)
Date in the format *yyyy-mm-dd* or *yy-mm-dd*, e.g. 2004-01-29 or 04-01-29 for the 29th of January, 2004. FT then displays all log records written after the specified date and time.

TIME=00:00 / <time 1..8>
Time for the day specified with CREATION-TIME. FT displays all log records written after the specified time. The time is entered in the format *hh:mm:ss*, e.g. 14:30:10.

TO=*TOMORROW / *TODAY(...) / <date 8..10>(...)
Date in the format *yyyy-mm-dd* or *yy-mm-dd*, e.g. 2004-01-29 or 04-01-29 for the 29th of January, 2004. FT then displays all log records written up to the specified date and time.

TIME=00:00 / <time 1..8>
Time for the day specified with CREATION-TIME. FT displays all log records written up to the specified time. The time is entered in the format *hh:mm:ss*, e.g. 14:30:10.

CREATION-TIME = *DAYS(NUMBER=<integer 1..1000>)
This field is specified in number of days. All logging sets that were created in the last n calendar days, including today, are output.

RECORD-TYPE=
Defines which type of log record is to be displayed.

RECORD-TYPE=*ALL
The record type is not a selection criterion.

RECORD-TYPE=*PARAMETERS(...)
The type of the log record.

FTAC=
(*TRANSFER-FILE, *READ-FILE-ATTRIBUTES, *DELETE-FILE, *CREATE-FILE, *MODIFY-FILE-ATTRIBUTES, *READ-DIRECTORY, *MOVE-FILE, *CREATE-DIRECTORY, *DELETE-DIRECTORY, *MODIFY-DIRECTORY / *LOGIN) / *NONE / list-poss(11): *TRANSFER-FILE / *READ-FILE-ATTRIBUTES / *DELETE-FILE / *CREATE-FILE / *MODIFY-FILE-ATTRIBUTES / *READ-DIRECTORY / *MOVE-FILE / *CREATE-DIRECTORY / *MODIFY-DIRECTORY / *DELETE-DIRECTORY / *LOGIN
Specifies whether or not FTAC log records are to be output. If yes, it can also be specified for which FTP functions the FTAC log records are to be output. In this case, the following definitions apply:

*TRANSFER-FILE
All log records for the function “transfer files” are shown.
This corresponds to the server commands *retr*, *stor*, *stou* and *appe*.

*READ-FILE-ATTRIBUTES
All logging records for the function “Read file attributes” are displayed.

*DELETE-FILE
All log records for the function “delete files” are shown.
This corresponds to the server command *dele*.

*CREATE-FILE
All log records for the function “create files” are shown.
This corresponds to the server command *site file*.

*MODIFY-FILE-ATTRIBUTES
All log records for the function “modify file attributes” are shown.
This corresponds to the server command *rnfr*.

*READ-DIRECTORY
All log records for the function “read directories” are shown.
This corresponds to the server commands *cwd*, *xcwd*, *list*, *mlsd*, *mlst*, *nlst*, *pwd*, *xpwd*, *cdup*, *xcup*, *size* and *mdtm*.

*MOVE-FILE
All logging records for the function “Copy and delete files” are displayed.

*CREATE-DIRECTORY
All log records for the function “create directory” are shown.
This corresponds to the server commands *mkd* and *xmkd*.

*DELETE-DIRECTORY

All log records for the function “delete directory” are shown.
This corresponds to the server commands *rmd* and *xrmd*.

*MODIFY-DIRECTORY
All logging records for the function “Modify directory” are displayed.

*LOGIN
All logging records for the function “Inbound FTP access” are displayed. Log records of the type *LOGIN are only written in the case of an incorrect transfer admission.

INITIATOR=
Selects log records by the initiator.

INITIATOR=(*LOCAL,*REMOTE)
The initiator is not a selection criterion.

INITIATOR=*REMOTE
This corresponds to the default value, since FTP requests are always remote requests.

PARTNER=
Client system.

PARTNER=*ALL
The client system is not a selection criterion.

PARTNER=<text 1..200 with-low>
The client system for which log records are to be shown. For more information on address specifications, see „openFT (BS2000) - Command Interface“.

FILE-NAME=
File name.

FILE-NAME=*ALL
The file name is not a selection criterion.

FILE-NAME=<full-filename 1..54> / <c-string 1..512 with-low>
***POSIX(NAME=posix_pathname_1..510)**
Fully qualified name of the files for which you want to view the log records.

FILE-NAME=<partial-filename 2..53>
Partially qualified name of the files for which you want to view the log records.

FILE-NAME = *DIRECTORY(...)
Name of the directory.

***DIRECTORY(...)**
The directory specification relates to the corresponding specification in the SHOW-REMOTE-FILE-ATTRIBUTES command
(see „openFT (BS2000) - Command Interface“).

NAME = *ALL
The directory is not a selection criterion

NAME = <partial-filename 1..53> / <c-string 1..512 with-low>
Name of the directory. Directories are represented by partially qualified file names in DVS.

REASON-CODE=
Selects log records by the REASON code.

REASON-CODE=*ALL
The REASON code is not a selection criterion; all records are shown.

REASON-CODE=*FAILURE
All log records with error codes are shown.

REASON-CODE=<text 1..4>
Defines which log records to be shown by specifying the error codes. Leading zeros may be omitted (e.g. 14 for 0014).

TRANSFER-ID =
Selection on the basis of the request ID.

TRANSFER-ID = *ALL
The request ID is not used as a selection criterion.

TRANSFER-ID = <integer 1..2147483639>
Only outputs log records for the specified request ID.

NUMBER=
The maximum number of log records to be shown.

NUMBER=1 / <integer 1..99999999>
The maximum permissible number of log records that can be displayed. By default, only one log record is shown.

NUMBER=*ALL
All log records are shown.

INFORMATION=
Scope of the required information.

INFORMATION=*STD
Only the standard information is shown for the log records.

INFORMATION=*ALL
The log records are shown in their entirety (long form).

OUTPUT=
Determines the output medium.

OUTPUT=*SYSOUT(...)
The output is sent to SYSOUT.

OUTPUT=*SYSLST(...)
The output is sent to SYSLST.

LAYOUT=*STD
The output is formatted using a standard layout that can be easily read by the user.

LAYOUT=*CSV
The output is supplied in CSV (Comma Separated Value) format. This is a widely used tabular format, especially in the PC environment, in which individual fields are separated by a delimiter, which is usually a semicolon ";".

Description of the output fields

Short output form of an FT logging record (example)

```
/SHOW-FT-LOGGING-RECORDS NUMBER=2
%TYP LOGG-ID TIME    RC    PARTNER  INITIATOR INIT USER-ADM FILENAME
%2010-06-22
%PM    3283 18:26:59 0000 <G133H301 *REMOTE        FT2V292  TEST2
%P     3212 11:33:53 0000 >G133H301 *REMOTE        FT2V292  TEST1
```

Long output form (example)

```
/SHOW-FT-LOGGING-RECORDS NUMBER=2,INFORMATION=*ALL
%LOGGING-ID  = 00003283  RC     = 0000       TIME    = 2010-06-22 18:26:59
%   INITIATOR= *REMOTE   PARTNER = G133H301   REC-TYPE = FTAC(FTP)
%   INITSN   =           TRANS   = FROM       FUNCTION = MODIFY-FILE-ATTR
%   USER-ADM = FT2V292   PROFILE = Z          PRIV     = NO
%   FILENAME = TEST2

%LOGGING-ID  = 00003212  RC     = 0000       TIME    = 2010-06-22 11:33:53
%   INITIATOR= *REMOTE   PARTNER = G133H301   REC-TYPE = FTAC(FTP)
%   INITSN   =           TRANS   = TO         FUNCTION = TRANSFER-FILE
%   USER-ADM = FT2V292   PROFILE = Z          PRIV     = NO
%   FILENAME = TEST1
```

Explanation

Identifier	**Explanation**	
TYP (column 1) or REC-TYPE	Indicates whether an FT or FTAC log record is involved. In the short output form, the first column T indicates the FT log record; C indicates the FTAC log records, and P indicates the FTP-specific FTAC log record; in the long output form (REC-TYPE), these details are written out in full.	
TYP (column 2-3) or FUNCTION	Definition of FT function:	
	␣	ansfer file
	V	transfer file and delete send file (only inbound possible)
	A	read file attributes
	D	delete file
	C	create file
	M	modify file attributes
	R	read directory
	CD	create director
	MD	modify directory
	DD	delete directory
	L	login (inbound FTP access)
LOGG-ID or LOGGING-ID	Number of the log record (max. 12 positions)	
TIME	Time when the log record was written	
RC	Reason Code. Indicates if a request was successfully executed, and if not, why it was rejected or terminated. If an FTP request is rejected for “FTAC reasons” (e.g. 0014), the precise reason for the termination can be found in the FTAC log record. Further information on the reason code can be obtained by using the BS2000 command HELP-MSG-INFORMATION.	
PARTNER	Returns information on the client system involved. The output shows the possibly abbreviated symbolic name with up to eight characters. In the short form, the client system name is preceded by an identifier from which you can determine the direction of the request.	
TRANS=TO or > with PARTNER	The transfer direction is to the client system.This direction is specified for a - send request - request to view remote file attributes - request to view remote directories	
TRANS=FROM or < with PARTNER	The transfer direction is to the local system. This direction is specified for a - receive request - request to modify remote file attributes - request to delete remote files	

Identifier		Explanation
TRANS	BOTH	The transfer direction is to the client system and to the local system.
INITIATOR		Initiator of the request; for FTP, always *REMOTE
INIT or INITSN		For FTP, this field is always empty.
USER-ADM		User ID referenced by the requests on the local system.
FILENAME		File name on the local system
PROFILE		Admission profile used
PRIV	*NO	Non-privileged admission profile
	*YES	Privileged admission profile

Command return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
0	0	CMD0001	No log records available for the selection criteria.
33	32	CMD0221	Request rejected. Internal error.
36	32	CMD0221	Request rejected. Request data inconsistent.
83	32	CMD0221	Internal error
88	32	CMD0221	Error during OPS generation.
36	64	FTR1036	User not authorized for other user IDs,
2	0	FTR2225	Information output cancelled.

SC1/2 = Subcode 1/2 in decimal form
More information can be found in the manual “openFT - Command Interface”.

Example

You want to view all log records created under your user ID before 01.01.11.

```
/SHOW-FT-LOGGING-RECORDS␣SELECT=*PARAMETERS(LOGGING-DATE=2011-01-01), -
/                         NUMBER=*ALL
```

All log records entered before 00:00 on 01.01.11 are displayed.

You now want to display the first record of this output in detail.

```
/SHOW-FT-LOG-REC␣(LOG-DATE=2011-01-01),INF=*ALL
```

5.2.10 SHOW-FT-PROFILE - Display admission profile

With the command SHOW-FT-PROFILE, FTAC users can obtain information on their admission profiles. Either the contents of the selected admission profile or only its name can be displayed. It is not possible to use SHOW-FT-PROFILE to access passwords or transfer admissions defined in the profile! If a transfer admission is forgotten, a new one must be specified with MODIFY-FT-PROFILE.

If you call *HELP* for the SDF command syntax shown below, you may also see some operands that are not indicated here. This is because only the operands relevant for FTP are described in this section.

```
SHOW-FT-PROFILE - showing operands relevant for FTP

 NAME = *ALL / <alphanum-name 1..8> / *STD
, SELECT-PARAMETER = *OWN / *PARAMETERS(...)
    *PARAMETERS(...)
        | TRANSFER-ADMISSION = *ALL / *NOT-SPECIFIED / <alphanum-name 8..32> /
        |                                <c-string 8..32 with-low> / <x-string 15..64> / *SECRET
        | , OWNER-IDENTIFICATION = *OWN / *ALL / <name 1..8>
, INFORMATION = *ONLY-NAMES / *ALL
, OUTPUT =*SYSOUT (LAYOUT = *STD / *CSV ) /
              *SYSLST (LAYOUT = *STD / *CSV)
```

Operands

NAME=
With NAME, you enter the name of the admission profile that you want to view.
NAME accesses a user-wide unique name of an admission profile.

NAME=*ALL
You want to view all admission profiles under your user ID.

NAME=<alphanum-name 1..8>
You want to view the admission profile with the specified name.

NAME = *STD
Displays the default admission profile for your own user ID.

SELECT-PARAMETER=
With SELECT-PARAMETER, you can specify selection criteria for the admission profiles you want to view.

SELECT-PARAMETER=*OWN
With *OWN, you can view all the admission profiles of which you are the owner. This means that you can view all the admission profiles that are assigned to your user ID.

SELECT-PARAMETER=*PARAMETERS(...)
This structure contains the selection criteria with which you can access your admission profiles.

TRANSFER-ADMISSION=
With TRANSFER-ADMISSION, you can enter the transfer admission defined in an admission profile as a selection criterion.

TRANSFER-ADMISSION=*ALL
TRANSFER-ADMISSION is not used as a selection criterion.

TRANSFER-ADMISSION=*NOT-SPECIFIED
Only admission profiles without a defined transfer admission are displayed.

TRANSFER-ADMISSION=<alphanum-name 8..32> /
<c-string 8..32 with-low> / <x-string 15..64>
You want to view your admission profile that can be addressed with this transfer admission.

TRANSFER-ADMISSION=*SECRET
The system prompts you to enter the transfer admission, but does not display your entry on the screen.

OWNER-IDENTIFICATION =*OWN / *ALL / <name 1..8>
OWNER-IDENTIFICATION authorizes the FTAC user to access his own admission profiles. All three entries have the same effect.

INFORMATION=
With INFORMATION, you determine the scope of information desired.

INFORMATION=*ONLY-NAMES
FTAC only shows the names of the admission profiles and indicates whether they are privileged.

INFORMATION= *ALL
FTAC shows the contents of the admission profile, excluding any passwords and the transfer admission.

OUTPUT=
Determines the output medium for the information requested.

OUTPUT=*SYSOUT(...)
The output is sent to SYSOUT.

OUTPUT=*SYSLST(...)
The output is sent to SYSLST.

LAYOUT=*STD
The output is formatted using a standard layout that can be easily read by the user.

LAYOUT=*CSV
The output is supplied in CSV (Comma Separated Value) format. This is a widely used tabular format, especially in the PC environment, in which individual fields are separated by a delimiter, which is usually a semicolon ";".

Example

The FTAC administrator wants to view the admission profile PROFPROD with the command SHOW-FT-PROFILE to determine if this profile could compromise data security. This is achieved with the following command:

```
/SHOW-FT-PROFILE␣NAME=PROFPROD,
          SELECT-PARAMETER=(OWNER-IDENTIFICATION=DONALD),INFORMATION=*ALL
```

Short form:

```
/SHOW-FT-PROF␣PROFPROD,(,DONALD),INF=*ALL
```

The output appears as follows:

```
%PROFPROD
% IGN-MAX-LEV = (IBR)
% FILE-NAME   = (PREFIX=SALES.)
% USER-ADM    = (DONALD,M4711DON,OWN)
% PROC-ADM    = SAME
```

The first line shows the name of the admission profile. The next two lines show the settings that were made by Donald in the command CREATE-FT-PROFILE with the operands IGNORE-MAX-LEVELS=(INBOUND-RECEIVE=*YES) and FILENAME=(PREFIX=SALES). The values for USER-ADMISSION and PROCESSING-ADMISSION were not set by Donald, so the default values are being used.

Command return codes

(SC2)	SC1	Maincode	Meaning
0	64	FTC0052	The information output was interrupted.
0	64	FTC0053	No FT profile that meets the specified criteria exists.
0	0	FTC0054	No information exists for the specified criteria.
0	64	FTC0153	The owner identification entered is not the user´s own ID.
0	64	FTC0171	The specified profile does not exist.
0	64	FTC0255	A system error occurred.

SC1/2 = Subcode 1/2 in decimal form
More information can be found in the manual “openFT - Command Interface”.

6 TELNET

This section describes both the TELNET client in BS2000 and the TELNET server (as of page 325).

Basic function

When a user sets up a connection to a partner system with the TELNET program, his or her terminal behaves like a line terminal directly connected to the partner system.

TELNET enables interactive communications only in line mode. Consequently, if the partner system is a BS2000 system, for example, FHS and EDOR cannot be used, and SDF and EDT may only be used in line mode. Similarly, if the partner system is a Unix system, for example, CED cannot be used.

Other functions

- Information for the user
  In a TELNET session, the user can obtain information on
  - the available commands and their significance,
  - the partner system with which the user is currently connected and the conditions (options) for data transfer.
- Monitoring functions
  Trace functions corresponding to the server trace functions that can be initiated by the administrator via */INFORM-PROGRAM* command of the system console have also been implemented in the client for maintenance and diagnostic purposes.
- Interface to the local and remote operating system
  An interface to the local operating system is provided to allow the user to issue a BS2000 command without having to interrupt the program run (and first clear the connection to the partner system).

6.1 TELNET client in BS2000

Every TELNET user in BS2000 opens a separate client task. On issuing the *open* command, the client establishes a connection with the desired TELNET server on a remote host.

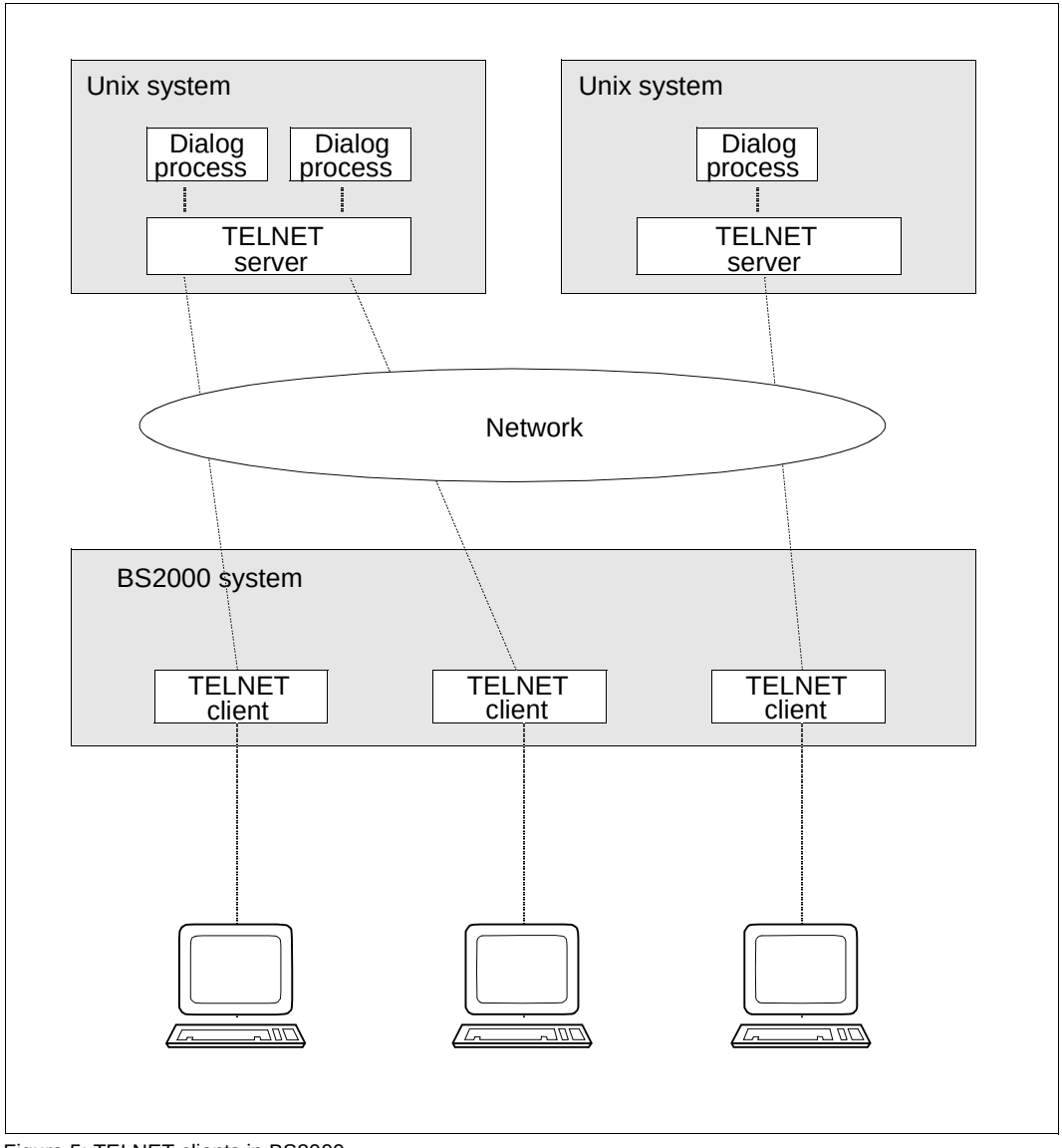


Figure 5: TELNET clients in BS2000

6.1.1 Command mode and input mode

TELNET can be used in interactive mode with hosts attached to the local network. In order to determine whether inputs are directed to the TELNET client or to the remote host with which a dialog is being maintained, TELNET recognizes two operating modes:

- command mode, in which commands are passed to the TELNET client.
- input mode, in which data and commands are passed to the remote interactive host.

Only line-oriented dialogs are possible with TELNET. Consequently, if the remote computer is a Unix system, for example, the editor CED cannot be used. Similarly, if the partner is a BS2000 system, FHS cannot be used at all, and SDF and EDT can be used only in line mode.

The TELNET command *!* can be used to switch from TELNET command mode to BS2000 or POSIX command mode.

The figure on page 280 shows the three modes for entering data and commands and the transitions between them. (The precise behavior on switching to BS2000 or POSIX command mode with the *!* command and to TELNET command mode with escape characters is given in the detailed description of the commands.)

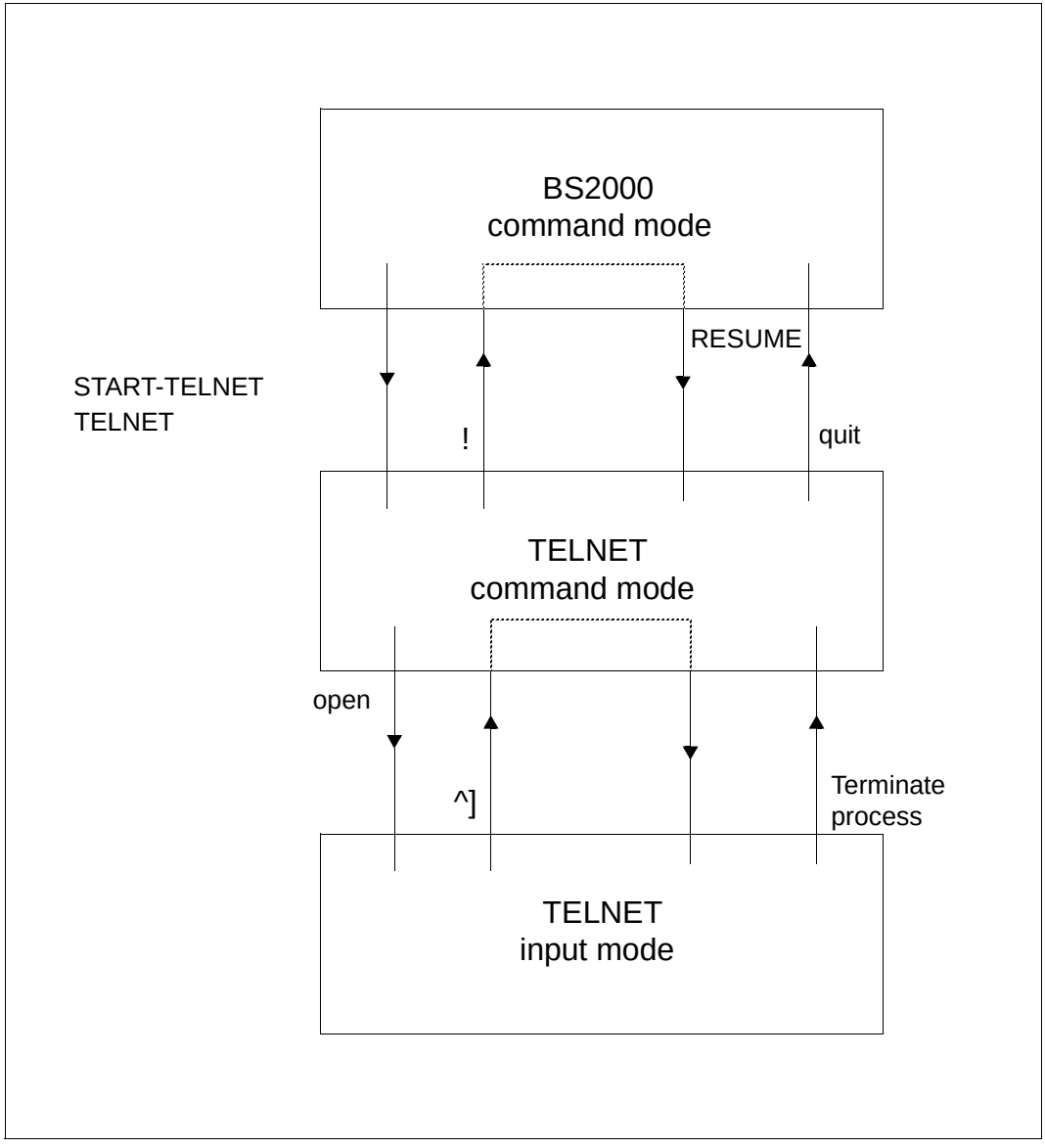


Figure 6: Input modes

Starting TELNET

You start the TELNET client as follows:

START-TELNET or TELNET

The client is terminated with the *quit* command.

In interactive mode, TELNET responds with a version number and the prompt *telnet>*. Task switch 1 must not be set. On starting TELNET, you will find yourself in TELNET command mode. In this mode, you can control TELNET and also retrieve data.

The *open <ipadr>* command connects you to a remote host. You will now be in the TELNET input mode and can direct any line-oriented commands to the operating system on the remote host. The connection to the remote system is cleared if the appropriate command to terminate the current process is issued on the remote operating system.

If you want to issue a TELNET command, you can interrupt the input mode by entering the escape character. In TELNET command mode, the connection to the remote system can be cleared by issuing the *close* command.

The predefined setting for the escape character is ^], but can be changed by using the *escape* command. The function of the CONTROL key on asynchronous terminals (Unix world) corresponds to the symbol ^. If the ^ character is to be transmitted, it must be entered twice.

6.1.2 TELNET client in POSIX

If the TELNET client is running under POSIX, it issues the following start message:

```
POSIX-TELNET <vers> <date> <time>
```

<date> and <time> here specify the date and time when the main module was compiled. At startup time, the TELNET client switches to the POSIX directory.

The TELNET call with switch -n in POSIX corresponds to batch mode in BS2000.

The prompt for the login password appears only after a short wait time when separate entry is used.

Secure TELNET client in POSIX

Secure operation (by means of authentication and encryption) of the TELNET client in POSIX is ensured in same way as secure operation of the TELNET client in BS2000 (see the section “Security in the TELNET client” on page 283).

The private option file has the name $HOME/.telnet.options. The file names relate to the POSIX file system. If, for example, you want to use a BS2000 file as CertificateFile, you must prefix the file name with “/BS2/”.

6.1.3 Security in the TELNET client

There are three methods of guaranteeing secure operation of TELNET by means of authentication and encryption:

- START-TLS option

  The START-TLS option was implemented exclusively for TLS/SSL and is supported in BS2000 by the client option *-Z tls-required* and the client command *tls*.

- “Telnet Authentication Option” (RFC 2941) for negotiating an authentication method

  In BS2000 only TLS/SSL is currently supported. The “Telnet Authentication Option” is selected using the *-A* option or the client command *auth*. The “Telnet Authentication Option” will possibly gain in importance in the future because it permits a very wide variety of authentication methods to be supported, including Kerberos. In the following, the “Telnet Authentication Option” will be referred to as AUTHENTICATION option.

- “Telnet Data Encryption Option” (RFC 2946) for negotiating a symmetric encryption method and the associated key

  In BS2000 only DES 64 (RFC 2952, RFC 2953) is currently supported. The “Telnet Data Encryption Option” is selected using the *-E* option or the client command *encrypt*. In the following, the “Telnet Data Encryption Option” will be referred to as ENCRYPTION option.

START-TLS option (see page 286), AUTHENTICATION option (see page 298) and ENCRYPTION option (see page 299) are described in detail in the following sections.

6.1.3.1 Selecting the options via the option file

You store the options settings in one or more option files. One of these option files is read in when the TELNET client is started. However, you can also adapt the option settings to suit current requirements later by reading in an option file with the TELNET client command *readopt*.

Determination of the relevant option file by the TELNET client

The TELNET client proceeds as follows to determine the relevant option file:

1. First the TELNET client searches for a centrally stored option file with the default file name $.SYSDAT.TCP-IP-AP.*nnn*.T-GOPT.
2. Regardless of the central file named under 1.), the TELNET client also searches for a user-owned option file under the name SYSDAT.TCP-IP-AP.*nnn*.TEL.OPT. If this file exists, the TELNET client reads the options from there.

For information on the special features in TLS support of the TELNET client in POSIX see page 282.

If an option is included in both files but with different values, the value defined in the user-owned file applies.

If security settings are specified using a TELNET client command, these have priority over those specified in the option files.

Notation of the options in the option file

The various options must be entered in the option file according to the following rules:

- Each line that is to be continued must be terminated with the continuation character “\”.
- A line beginning with the character “#” in column 1 is ignored when the file is read in.
- The option names are not case-sensitive.

6.1.3.2 Controlling the security settings using TELNET client commands

The following TELNET client commands can be used to control the security settings on the TELNET client:

- *tls* - Enable/disable TLS support (see page 319)
- *auth* - Modify settings of the authentication option, show status (see page 302)
- *encrypt* - Modify settings of the encryption option, show status (see page 306)

The security settings made via TELNET client commands have priority over those specified in the option files.

6.1.3.3 START-TLS option

This option enables you to control TLS support on the TELNET client. Negotiation of the arrangements for authentication is handled by TLS in this case to relieve the load on TELNET.

You enter the options for using TLS support in the option file(s) mentioned above as follows:

```
-Z <option>
```

Time when the options or changes to the options should become effective

After the TELNET client has started up you can read in the required option file using the client command *readopt* (see page 314). You can repeat this procedure as often as you wish.

The *-Z OpenSSLLibname* option (see page 297) is only evaluated only once during a TELNET session, namely when the OpenSSL library is loaded. All other options become effective after the connection to the server has been set up.

Description of the -Z options

The -Z options are described below. The following must be observed here:

– With the exception of the *-Z tls-required* option (see page 287) all the -Z options can also be used for the AUTHENTICATION option (see page 298), provided TLS/SSL support is connected with the AUTHENTICATION option.

– Parallel support of the START-TLS option (*-Z tls-required*) and AUTHENTICATION option (*-A*) is meaningless. When both of these options are specified the following error message is consequently issued:

```
Both START-TLS and AUTHENTICATION-Option not allowed
```

– The *-Z OpenSSLlibName* option (see page 297) is also significant for supporting the ENCRYPTION option (see page 299) as only encryption routines from this OpenSSL library are used.

-Z tls-required - Enable/disable TLS security on the TELNET client

The *-Z tls-required* option is used to enable and disable TLS security on the TELNET client via the START-TLS option.

-Z tls-required
[**=** { **yes** | **no** }]

yes
START-TLS support is enabled.

no
START-TLS support is disabled.

-Z tls-required specified without operand
-Z tls-required = yes applies (START-TLS support is enabled).

-Z tls-required not specified
START-TLS support is not enabled.

-Z CertificateFile - Specify file with X.509 client certificates

The *-Z CertificateFile* option is used to specify a file which contains the X.509 client certificate in PEM format. This file can also contain the client's private key. However, generally the certificate and key are stored in different files. In this case the key file is specified using the *-Z KeyFile* option (see page 289).

-Z CertificateFile
={<file-name 1..54> | ***NONE**}

<file-name 1..54>
Name of the file which contains the X.509 client certificate in PEM format.

***NONE**
No client certificate is used (and thus no client authentication, either).
*NONE is the default.

-Z KeyFile - Specify file with client key in PEM format

The *-Z KeyFile* option is used to specify a file which contains the private client key in PEM format.

If both an X.509 client certificate and a private client key are contained in the same file (see the *-Z CertificateFile* option on page 288), the *-Z KeyFile* option need not be specified.

If the client key is protected with a passphrase, this passphrase must be entered after the TELNET client has started up when establishing a TLS-secured TELNET connection.

-Z KeyFile
={<file-name 1..54> | ***NONE**}

<file-name 1..54>
Name of the file which contains the private client key.

***NONE**
No separate client key file is used.

The default is the file name specified in the *-Z CertificateFile* option (see page 288).

-Z CACertificateFile - Specify file with server authentication

The *-Z CACertificateFile* option is used to specify a file containing the CA certificates in PEM format which are required for TELNET server authentication. The individual PEM certificates are arranged sequentially in the file.

You can process the file with a text editor of your choice when you wish to add or delete certificates. The individual certificates are registered in the file as follows:

```
-----BEGIN CERTIFICATE-----
< CA certificate in Base64 encoding >
-----END CERTIFICATE-----
```

Text outside these sequences is ignored by the TELNET client and can therefore be used to identify the certificates which, owing to the Base64 encoding, are not available in readable form.

-Z CACertificateFile
={<file-name 1..54> | ***NONE**}

<file-name 1..54>
Name of the file containing the certificates in PEM format which are required for TELNET server authentication.

***NONE**
No file with CA certificates is specified.
*NONE is the default.

-Z CARevocationFile - Specify file with CRL

The *-Z CARevocationFile* option is used to specify a file which contains the CRLs (Certificate Revocation Lists) of the Certificate Authorities. (Certificates issued by a Certificate Authority can be declared invalid by publication of a Certificate Revocation List (CRL).)

-Z CARevocationFile
={<file-name 1..54> | ***NONE**}

<file-name 1..54>
Name of the file which contains the CRLs of the Certificate Authorities.

***NONE**
No file with CRLs is specified.
*NONE is the default.

-Z VerifyServer - Verify TELNET server certificate (yes/no)

The *-Z VerifyServer* option defines whether the TELNET server certificate should be verified.

-Z VerifyServer
={**YES** | **NO**}

YES
The TELNET server certificate should be verified.
YES is the default.

NO
The TELNET server certificate should not be verified.
This setting makes you vulnerable to “man in the middle” attacks.

-Z VerifyDepth -Define verification depth

The *-Z VerifyDepth* option is used to define the verification depth, in other words the maximum permissible number of certificates between the TELNET server certificate and the certificate which is known to the TELNET client.

Here you must note the following:

- If the value 1 (default) is specified as the maximum depth, the server certificate must have been signed directly by a CA (Certificate Authority) that the TELNET client knows for it to be accepted.
- If the maximum depth is exceeded, the connection is cleared, unless verification of the TELNET server certificate has been disabled with *-Z VerifyServer NO* (see page 292).
- Specifying the depth as 0 is meaningless. In this case only self-signed certificates would be permissible.

-Z VerifyDepth
=<depth>

<depth>
Maximum permissible number of certificates between the TELNET server certificate and the certificate known to the TELNET client (including the TELNET server certificate).
Default: 1

-Z CipherSuite - Specify a cipher suite preference list

The *-Z CipherSuite* option is used to specify a cipher suite preference list. If this option is not specified, a default preference list is used.

-Z CipherSuite
=<specification>

<specification>
Specification of a cipher suite preference list, see section “Specification of a cipher suite preference list” on page 60.
ALL: !EXP: !ADH:!RC4 is the default.

-Z RandomSeed - Initialize pseudo random number generator

The *-Z RandomSeed* option is used to specify how the pseudo random number generator used by TLS is initialized. Efficient initialization with values that are as random and unforeseeable as possible is of decisive importance for TLS security. If the BS2000 subsystem PRNGD (**P**seudo **R**andom **N**umber **G**enerator **D**emon) is active on the system on which the TELNET client is running, PRNGD is used for initialization, so the setting of the *-Z RandomSeed* option is of practically no significance. Subsystem PRNGD is described in the “interNet Services Administrator Guide”.

-Z RandomSeed
={**PROGRAM** | **USER**}

PROGRAM

Program-internal functions are used. These utilize above all fluctuations of the real-time clock in relation to the timer of the system CPU to generate random numbers for initialization.

USER

Part of the initialization is the same as with the specification PROGRAM. In addition, the user is repeatedly prompted to key in characters as randomly as possible and/or to press the ENTER key. The time stamp from the input is used for initialization. The characters entered are also used for initialization. Since, however, the randomness of the characters is not known and these characters can in many cases be intercepted, they are not taken into account when estimating whether there is already enough initialization material. Only the number of times the ENTER key is pressed is taken into account in this estimate.
*USER is the default.

If the TELNET client is operated in batch mode, it is generally recommendable to use the PROGRAM setting as no user is available to enter random numbers in batch mode.

-Z Protocol - TLS/SSL protocol selection

OpenSSL supports Versions 3 of the SSL protocol and also Versions 1, 1.1 and 1.2 of the TLS protocol. Some of these protocols can be activated selectively using the *-tlsProtocol* option.

-Z Protocol
={**+** | **-**} {**SSLv3** | **TLSv1** | **TLSv1.1** | **TLSv1.2** | **All** } ...

+
The protocol specified after this sign is permissible.

-
The protocol specified after this sign is not permissible.

SSLv3
SSL protocol Version 3

Version 3 of the SSL protocol displays some security-related deficiencies and should therefore not be used if possible.

TLSv1
TLS protocol Version 1

TLSv1.1
TLS protocol Version 1.1

TLSv1.2
TLS protocol Version 1.2

ALL
All protocols are to be enabled.

All -SSLv3 is the default.

Example

The specifications `-Z Protocol=TLSv1 TLSv1.1 TLSv1.2` and `-Z Protocol=All -SSLv3` have the same effect as long as no support of the future TELNET version 1.3 is added to the TELNET.

-Z OpenSSLlibName - Define an LMS file for an OpenSSL library

The *-Z OpenSSLlibName* option is used to define the LMS file from which the OpenSSL library should be dynamically loaded. The OpenSSL library is only dynamically loaded if at least one of the two options *-Z tls-required* or *-A on* is specified.

It may be necessary to specify a name other than the default name if, for example, the OpenSSL library is also used by other products.

Dynamic loading of the OpenSSL library can be expedited with the aid of DAB using caches. If the OpenSSL library is used jointly by a number of products, the size of the DAB buffer used is reduced.

-Z OpenSSLlibName
=<openssl-libname>

<openssl-libname>
Name of the LMS file from which the OpenSSL library is to be dynamically loaded.
Default: $.SYSLNK.TCP-IP-AP.*nnn*

6.1.3.4 Option -A - Enable/disable AUTHENTICATION option

The *-A* option allows support of the AUTHENTICATION option to be enabled and disabled. In BS2000 the AUTHENTICATION option is currently only implemented for TLS/SSL. The settings required for SSL operation can thus be made with the aid of the *-Z* options (see page 286 ff).

Parallel support of the START-TLS option (*-Z tls-required*) and AUTHENTICATION option is meaningless. If the *-A on* and *-Z tls-required* options are specified simultaneously, the following error message is therefore issued:

```
Both START-TLS and AUTHENTICATION-Option not allowed
```

Alternatively you can also select that the AUTHENTICATION option should be supported using the TELNET client command *auth* (see page 302).

-A
on | **off** | **debug** | **status**

on
The AUTHENTICATION option is supported.

off
The AUTHENTICATION option is not supported.
off is the default.

debug
The authentication trace is enabled.

status
Species the current status of the AUTHENTICATION option.

6.1.3.5 Option -H -Enable/disable the ENCRYPTION option

With the *-H* option you can enable and disable support of the ENCRYPTION option, which is used to negotiate the encryption method and the key used. Currently only variants DES_CFB64 and DES_OFB64 of DES64 are supported in TELNET.

As only encryption routines from the OpenSSL library are used, you can specify this library − if its name is different from the default (SYSLNK.TCP-IP-AP.*nnn*) − with the aid of the *-Z OpenSSLlibname* option (see page 297).

The *-H* option is only effective if the *-Z tls-required* or *-A on* option is not specified at the same time.

Alternatively you can also select that the ENCRYPTION option should be supported using the TELNET client command *encrypt* (see page 306).

-H
on | **off** | **debug** | **key** <x-string 1..16>

on
An encryption method and a key are negotiated.

off
No encryption method and no key are negotiated.
off is the default.

debug
The encryption trace is enabled.

key <x-string 1..16>
Encryption key for DES

Note that no distinction is made between the key for encryption and the key for decryption. The TELNET client and TELNET server use the same key.

6.1.3.6 Option -X - Switch code tables

This option enables you to switch the currently selected code tables with which the TELNET client converts EBCDIC to ISO characters (extended ASCII character set). TCP-IP-AP uses the XHCS services here, in other words only those code tables may be specified which are entered in XHCS as compatible (see the “XHCS” manual). In the event of a connection between two BS2000 systems it must be ensured that both systems use the same code tables.

Alternatively you can also select the *-X* option settings using the TELNET client command *setcode* (see page 317).

-X
<ebcdic-table>:<iso-table>

<ebcdic-table> <iso-table>

Between the EBCDIC and ISO tables a code conversion table is generated with XHCS which is used by the TELNET client for all code conversions.

CAUTION!

The code tables may not be entered in reverse order because this results in incorrect conversion tables being generated, which means no further data transfer is possible.

The BS2000 TELNET server only supports simple 7-bit terminals, consequently it is meaningless to use the command for connections to the BS2000 TELNET server.

Example

```
-X EDF045:ISO88595
```

6.1.4 Overview of commands

TELNET commands may be abbreviated up to the point required for unique identification. Operands are separated by spaces.

Command	Function	Page
open	Open connection to remote host	310
close	Close connection to remote host	303
!	Switch to BS2000/POSIX command mode	323
crmod	Enable/disable insertion of carriage returns	304
escape	Change escape character	307
exit	Enable/disable client exit	308
setcode	Change code tables	317
^]	Switch to TELNET command mode	321
quit	Exit the TELNET client	313
rexit	Enable/disable server exits	315
send	Send special commands *ao* / *ip* / *ayt* / *nop*	316

Controlling TELNET

Command	Function	Page
?	Show information on TELNET commands	322
help	Same as ?; show information on TELNET commands	309
status	Show TELNET status information	318
options	Enable/disable display of options	312
trace	Set socket trace level	320
debug	Enable the TELNET trace	305

Information on TELNET

Command	Function	Page
auth	Enable/disable AUTHENTICATION option	302
encrypt	Enable/disable ENCRYPTION option	306
readopt	Read option file	314
tls	Enable/disable TLS support on the TELNET client	319

Information on TELNET

auth - Enable/disable AUTHENTICATION option

With *auth* you can enable and disable the AUTHENTICATION option. Alternatively you can also specify the settings of the *auth* command using the *-A* option (see page 298).

In BS2000 authentication is only implemented for SSL. You can therefore specify the required settings using the *-Z* options (see page 286 ff).

The settings of the *auth* command become effective in the next TELNET session.

auth
disable | **enable** | **status**

disable
Disables the AUTHENTICATION option.

enable
Enables the AUTHENTICATION option.

status
Shows the current status of the authentication types supported.

close - Close connection to remote host

The *close* command clears the connection to the remote host.

The connection to a remote host is set up by using the *open* command. The *status* command shows the name of the host for which a connection exists.

close

On executing the *close* command, you will be placed in the TELNET command mode.

Example

1. The status query shows that a connection to *systemd* exists.

```
status
Connected to systemd.
Escape character is '^]'.
ISO-codetable is ISO88591, EBCDIC-codetable is EDF041.
```

2. Switch to TELNET command mode.

```
^]
```

3. Close the connection to *anlaged*.

```
close
Connection closed
```

crmod - Enable/disable insertion of carriage returns

If the remote host sends messages that contain no carriage returns, the *crmod* command (enable insert carriage return) can be used to instruct the TELNET client to execute a line feed. The insertion of carriage returns is disabled by default.

Since a Unix system transmits carriage returns, enabling the insertion of carriage returns will result in to a duplication of the line feeds.

crmod

Example

The remote host is a Unix system. Duplicate line feeds are enabled.

```
crmod
Will map carriage return on output
```

Remark

Duplicate line feeds are enabled.

debug - Enable/disable DEBUG output

DEBUG output is primarily used by network administrators and support staff to diagnose problems on the network. Users do not usually require DEBUG output.

debug
<debug-value>

<debug-value>
Values in the range 0 through 9 are permitted.

0 No DEBUG output

1 All messages from the FTP client to the FTP server and from the FTP server to the FTP client are output.

If no operand is specified, the DEBUG output switch is toggled, i.e. if the DEBUG output is enabled, it is now disabled, and if it is disabled, it is now enabled. Values greater than 1 are treated in the same way as 1.

encrypt - Enable/disable ENCRYPTION option

With *encrypt* you can enable and disable support of the ENCRYPTION option, which is used to negotiate the encryption method and the key used. Currently only variants DES_CFB64 and DES_OFB64 of DES64 are supported in TELNET. *encrypt* is only effective if the *-Z tls-required* or *-A on* option is not specified at the same time.

As only encryption routines from the OpenSSL library are used, you can specify this library − if its name is different from the default (SYSLNK.TCP-IP-AP.*nnn*) − with the aid of the *-Z OpenSSLlibname* option (see page 297).

Alternatively you can specify the *encrypt* command settings using the *-H* option (see page 299).

The settings of the *encrypt* command become effective in the next TELNET session.

encrypt
disable | **enable** | **key** <x-string 1..16> | **status**

disable
Disables encryption for both directions (input/output).

enable
Enables encryption for both directions (input/output).

key <x-string 1..16>
Encryption key for DES

status
Shows the current settings for the encryption option. The display mainly covers the encryption algorithms supported.

escape - Change ESCAPE character

Entering the escape character switches you from TELNET input mode to TELNET command mode.

The escape character consists of two elements. The first element is ^, and the second element can be freely selected by the user with the *escape* command.

Switching from input mode to TELNET command mode is described under the *^]* command on page 321.

escape

exit - Enable/disable client exit

You can set exit routines for the TELNET client using the *exit* command.

You will find a more precise description of the exit mechanism in TELNET and of the *exit* command in the “interNet Services Administrator Guide”.

exit
[receive : < selector 1 >] [send : < selector 2 >]

help - Show information on TELNET commands

This command returns information on a specific TELNET command or on all TELNET commands.

help
[<command>]

<command>
TELNET command for which information is desired. If the operand is omitted, all the permitted TELNET commands are listed.

Example

```
telnet>  help
Commands may be abbreviated.  Commands are:

open            connect to a site
close           close current connection
quit            exit telnet
!               BS2000 MCLP
escape          set escape character
status          print status information
options         toggle viewing of options processing
crmod           toggle mapping of received carriage returns
trace           set socket trace level
help            print help information
setcode         change the codetables
send            send special commands ao/ip/ayt/nop
exit            set local exit
rexit           set remote exit
debug           set telnet trace
?               print help information
tls             switch on/off START-TLS Option
readopt         read Option File
auth            switch on/off AUTHENTICATION Option
encrypt         switch on/off ENCRYPTION Option
telnet>
```

open - Open connection to a remote host

The *open* command sets up a connection to a remote host. Either the name or the internet address of this host must be known. The host must either be part of the local network or be accessible via a gateway. The names and addresses of the hosts that can be reached via the *open* command can be obtained from the network administrator.

You can now log on to a remote BS2000 host with LOGON. If you enter the LOGON command without a password, BS2000 subsequently prompts the password with JMS0151. The input box for entering the password is non-displaying in TELNET.

After the connection has been correctly set up, line-oriented commands of the remote operating system can be entered (the user is in input mode). The connection is cleared by issuing the appropriate command to terminate the current process in the remote operating system.

open
<ipadr> | <remote-host> | local-host loopback [<port>]

<ipadr>
: Internet address (IPv4 or IPv6 address) of the remote host to which the connection is to be made:

- An IPv4 address must be specified in the usual “decimal-dotted” notation.
- An IPv6 address must be specified in the usual hexadecimal notation with a colon (:).

<remote-host>
: Symbolic name of the remote host to which the connection is to be made.

local-host
: A separate Internet is generated for *local-host* that is not identical to the address of the user’s own host in the local network. This address is known only to the local host; other computers cannot use it.

loopback
: *loopback* stands for the Internet address by means of which the local host can be accessed in the LAN.

<port>
: Port number of the TELNET server. The TELNET server is assigned port number 23 by default.

Example

The remote system is a Unix host.

1. Open a connection to the Unix host *systemd*.

```
open systemd
Trying...
Connected to systemd.
Escape character is '^]'.
ISO-codetable is ISO88591, EBCDIC-codetable is EDF041.
local exits defined:  receive: -, send: -
remote exits defined: receive: -, send: -
Host name: systemd
```

2. Login under the user ID *guest*.

```
guest
guest
Continue by entering RETURN or MENUE
```

options - Enable/disable display of options

A number of different options are negotiated between the TELNET client and the TELNET server (when setting up the connection, for example). If this switch is “enabled”, these TELNET options are displayed on the screen. Options transmitted by the local client are identified as SENT, while those received by the local client are identified as RCVD.

Options are not displayed by default.

options

Example

1. *option* command with response.

```
options
Will show option processing.
```

2. *open* command with the options displayed.

   The ECHO option is negotiated here between the sender and receiver.

```
open systemd
Trying...
Connected to systemd.
Escape character is '^]'.
ISO-codetable is ISO88591, EBCDIC-codetable is EDF041.
RCVD will ECHO (reply) SENT dont ECHO (reply)
RCVD wont ECHO (don't reply)
Host name: systemd
```

quit - Exit TELNET

The *quit* command terminates the TELNET program. If a connection to a remote host is still open, it is cleared (implicit *close*).

quit

Example

```
quit
Connection closed.
CPU time used 4.7232 seconds
```

readopt - Read in option file

The option file is read in with the *readopt* command. This command supplements/replaces the options which were read from the global and local option files when the TELNET client was started (if these files existed at startup time). You can execute the *readopt* command as often as you wish, for example to form option records which are distributed over different files, or to switch between various option records.

The *-Z OpenSSLlibname* option (see page 297) only becomes effective after the TELNET client has restarted. All other options become effective after the connection to the TELNET server has been set up.

readopt
<option-file 1..54>

<option-file 1..54>
Name of the option file.

rexit - Enable/disable server exits

The client allows you to set TELNET server exits using the *rexit* (remote exit) command. However, this only relates to exits for *send* and *receive* and only affects the connection of this special client to the server.

You will find a more precise description of the exit mechanism in TELNET and of the *exit* command in the “interNet Services Administrator Guide”.

rexit
[receive : < selector 1>] [send : < selector 2>]

send - Send special commands

send
ao | ip | ayt | nop | ?

ao
: (abort output) Purges the entire output of the remote host and dumps it to the terminal of the user process currently running on the remote host.

ip
: (interrupt process) Interrupts the process currently running on the remote host.

ayt
: (are you there) Checks whether the server is still “alive”. If this is the case, the server responds with *YES*.

nop
: Sends *No operation* to the server.

?
: Returns help on the *send* command.

setcode - Change code tables

The *setcode* command is used to change the current code tables with which the TELNET client converts from EBCDIC to ISO characters (extended ASCII character set).
The services of XHCS are utilized for this purpose, so only the code tables that have been entered as compatible in XHCS may be specified here (see the “XHCS” manual). If a connection is set up between two BS2000 systems, it is important to ensure that both partners use the same code tables.

Alternatively you can define the settings of the *setcode* command using the *-X* option (see page 300).

setcode
<ebcdic-table> <iso-table>

<ebcdic-table> <iso-table>
A code conversion table is generated with XHCS for conversion between EBCDIC and ISO codes. This conversion table is used by the TELNET client for all code conversions.

CAUTION!

It is essential to ensure that the code tables are specified in the correct sequence; otherwise, invalid conversion tables will be created, and the file transfer will not be possible.

Example

```
setcode EDF045 ISO88595
```

Remarks

Since the BS2000 TELNET server supports only simple 7-bit terminals, this command is meaningless for connections to the BS2000 TELNET server.

status - Display TELNET status information

The status message indicates,

- whether a connection exists and to which host,
- the elements of the current escape character.

The escape character is defined with the *escape* command.

status

Example

The currently set parameters of the TELNET client are shown.

```
status
No connection.
Escape character is'‘^]‘.
ISO-codetable is ISO88591 , EBCDIC-codetable is EDF041
local exits defined:  receive: -, send: -
remote exits defined: receive: -, send: -
TLS off
Authentication off
Encryption off
Decryption off
```

```
status
Connected to loopback.
Escape character is'‘^]‘.
ISO-codetable is ISO88591 , EBCDIC-codetable is EDF041
local exits defined:  receive: -, send: -
remote exits defined: receive: -, send: -
TLS off
Authentication off
Encryption off
Decryption off
```

tls - Enable/disable TLS support on the TELNET client

With *tls* you can enable and disable TLS security on the TELNET client. If TLS support is enabled, negotiation of the arrangements for authentication is handled by SSL to relieve the load on TELNET.

Alternatively you can also specify the settings of the *tls* command using the START-TLS option (*-Z tls-required*, see page 286 and page 287). You can specify the settings required for TLS/SSL operation using the *-Z* options. The description of the *-Z* options starts on page 288.

The settings of the *tls* command become effective in the next TELNET session.

tls
[**on** | **off** | **status**]

on
Enables support of the START-TLS option on the TELNET client.

off
Disables support of the START-TLS option on the TELNET client.

status
Shows the current status of TLS support.

tls command without operands specified
Enables support of the START-TLS option in the TELNET client.

If TLS security is not enabled, all *-Z* options are ignored, except in some cases the *-Z OpenSSLlibName* option (see page 297).

trace - Enable/disable SOCKET trace output

SOCKET trace output is primarily used by the network administrator and the support staff to diagnose problems on the local network. Normal users do not usually need any SOCKET trace output.

If SOCKET trace output is enabled, TELNET displays various types of diagnostic information on the screen.

trace
[<trace-value>]

<trace-value>
The permitted values are in the range 0 through 10. Higher parameter values display more information.

0 No TRACE output (disable TRACE output).

1 - 10 Displays information on SOCKET traces.

If no operand is specified, the TRACE output setting is toggled, i.e. if TRACE output was enabled, it is disabled, and if it was disabled, it is enabled (*trace-value*=1). Parameter values greater than 1 imply the output of all TRACE information at the lower levels.

^] - Switch to TELNET command mode

Entering the escape character switches you from TELNET input mode to the TELNET command mode.

^]

^] is the default setting for the escape character. After executing a valid TELNET command, you are normally returned to the input mode. If the command is not valid, TELNET remains in command mode.

The *escape* command allows you to select some other character instead of] as the escape character. On executing the *?* command, the TELNET command mode remains in effect. You can use the *!* command to switch to BS2000 command mode or to POSIX command mode. Note that the TELNET command mode also remains in effect on executing the *close* command.

? - Show information on TELNET commands

The *?* command returns information on a particular TELNET command or all TELNET commands.

?
[<command>]

See the description of *help* on page 309.

! - Switch to BS2000 or POSIX command mode

Switching to BS2000 command mode (TELNET client in BS2000)

The *!* command enables you to enter BS2000 commands. If no operand is specified, the system switches to BS2000 command mode. You can then enter any BS2000 commands and subsequently return to TELNET by issuing the BS2000 command *RESUME*. Alternatively, if you enter *!* with a BS2000 command as the operand, you will automatically be returned to TELNET after execution of the command.

!
[<bs2000-command>]

<bs2000-command>
Any line-oriented BS2000 command.
The BS2000 commands *LOAD, EXEC, LOGOFF* and *ABEND* may not be entered either directly or indirectly, since they terminate TELNET or the task in which TELNET is running.

Example

1. Switch to BS2000 command mode.

```
!
% PROGRAM BREAK AT 0013E8, AMODE = 24
```

2. Enter the BS2000 command *STATUS PROG*.

```
STA P
NAME    TSN TYPE         SIZE CURR-CMD  PROGRAM-NAME
PETER   2726 3 DIALOG    79 STA         :5:$TSOS.TELNET
BRONCO  2716 3 DIALOG    87 EXECUTE     :5:$TSOS.FTP
```

3. Return to the TELNET client.

```
R
```

Switching to POSIX command mode (TELNET client in POSIX)

The command *!* enables you to enter POSIX commands. If no operand is specified, the system switches to POSIX command mode and subsequently return to TELNET by issuing the POSIX command *exit*. If you enter *!* with a POSIX command as the operand, you will automatically be returned to TELNET after execution of the command.

!
[<posix-command>]

<posix-command>
Any line-oriented POSIX command.

6.2 TELNET server

The TELNET server in BS2000 accepts connection requests from TELNET clients on the local network and forwards them to *$DIALOG*.

The TELNET server is described in the System Administrator Guide for InterNet Services.

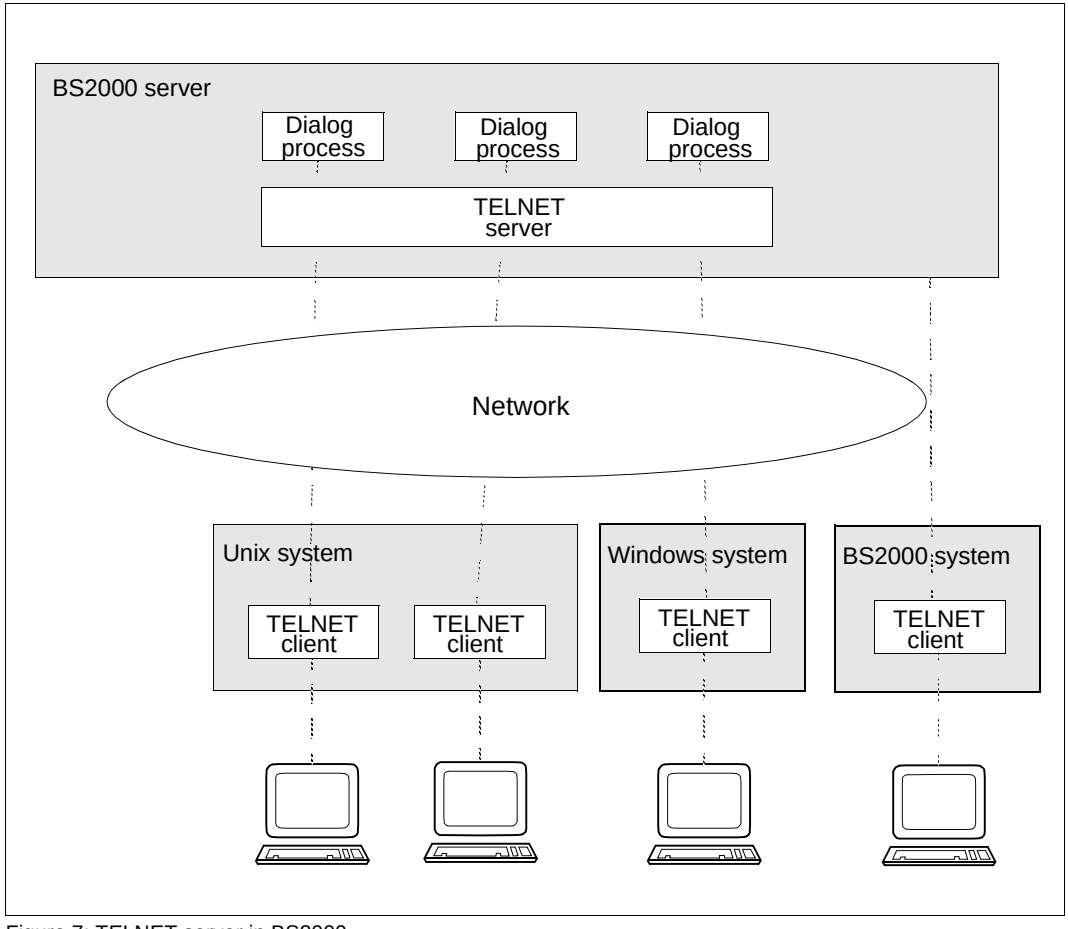


Figure 7: TELNET server in BS2000

Mode of operation

The server accepts connection requests and sets up a DCAM connection to *$DIALOG*. A simple printer terminal is emulated with respect to *$DIALOG*, i.e. only a line-oriented dialog is possible. No overflow control is implemented.

7 OpenSSH

This chapter is based on the manual pages (man pages) of OpenSSH and describes the OpenSSH client. The OpenSSH server is described in the “interNet Services Administrator Guide”.

The description provided here has been shortened to contain only the parts relevant to BS2000. At certain places in the chapter, such as in the descriptions of options, the OpenSSH man pages are referred to as only these contain the description of the most up-to-date versions. You can find the OpenSSH man pages on the Internet at *http://www.openssh.org/manual.html* or, after you have installed the component OpenSSH on your server,

- under *<installationpath>/readme/TCP-IP-SV.openssh/html/* as an HTML file,
- under *<installationpath>/readme/TCP-IP-SV.openssh/pdf/* as a PDF file,
- under *<installationpath>/readme/TCP-IP-SV.openssh/text/* as a text file.

The default installation path is: */opt/TCP-IP-SV/openssh*

When the “OpenSSH man pages” are referred to in the course of this chapter, these sources are meant. The man pages supplied with the product should preferably be used as these man pages contain the BS2000-specific adaptations (changed path names, extended functionality, etc.).

SSH (**S**ecure **Sh**ell) is a cryptographic protocol for performing the following tasks:

- Login on a remote computer
- Interactive / non-interactive command execution on a remote computer
- File transfer between different computers in a network

SSH designates not just the protocol itself but also concrete implementations.

You will find detailed information on the concept behind OpenSSH in the “interNet Services Administrator Guide”.

7.1 Component parts of the OpenSSH protocol suite

The OpenSSH protocol suite (only openSSH version 2 is supported) comprises the following programs and command :

- On the server side: server program *sshd* (see the “interNet Services Administrator Guide”)
- On the client side:
  - Client program *ssh* or *slogin* (see page 329): replaces *rlogin* and *telnet*.
  - *scp* (see page 339): replaces *rcp*.
  - *sftp* (see page 340): replaces *ftp*.
- Administration utilities:
  - *ssh-agent* (see page 345)
  - *ssh-add* (see page 347)
  - *ssh-keygen* (see page 349)
  - *ssh-keyscan* (see page 351)

To permit secure communication of Windows systems with BS2000 via SSH, the open source PuTTY (see *http://www.chiark.greenend.org.uk/~sgtatham/putty/*), for example, provides the functionality of the client side of OpenSSH.

7.2 OpenSSH client application ssh (slogin)

The OpenSSH client application *ssh* allows you to:

- Log in on a remote computer
- Execute commands on a remote computer

ssh is the secure alternative to *rlogin*, *rsh* and *telnet* and should be used instead of these programs. In contrast to *rlogin* and *rsh*, the OpenSSH client ensures secure, encrypted communication over an unsecured network.

7.2.1 Configuring the OpenSSH client ssh

The OpenSSH client *ssh* reads its configuration options sequentially from the following sources:

1. Command line arguments which you specify when calling *ssh* (see page 331), *scp* (see page 339) and *sftp* (see page 340).
2. User-specific configuration file (*$HOME/.ssh/config*)

   Although this file does not generally contain information which is relevant to security, read/write permission should only be granted for its owner. Access should be refused for all other users.
3. System-wide configuration file (*/etc/ssh/ssh_config*)

   This file contains default values for configuration parameters

   – if no user-specific configuration file exists or

   – if the relevant parameters are not specified in the user-specific configuration file.

The first value found applies for each option.

Syntax and semantics of the *ssh* configuration files

The *ssh* configuration files must have the following format:

- The configuration file is subdivided into one or more logical sections. Each section starts with a `Host` option which is followed by configuration options which differ from the `Host` option. The next `Host` option marks the start of the next section, and so on. A section's configuration options are only relevant for computers whose names are specified in the associated `Host` option.
- In each line the configuration file contains: optional blank, followed by a keyword and associated argument/associated argument list.

  The keyword and argument (list) can be separated by:

  – a blank

  – an optional blank and precisely one "="

  For keywords no distinction is made between upper and lower case. Arguments are case-sensitive; even "yes" and "no" must be entered in lower case.

  – Empty lines and lines beginning with "#" are interpreted as comments.

A detailed description of the configuration options is provided in the OpenSSH man pages.

7.2.2 Starting the OpenSSH client application

You start *ssh* with the following command:

```
ssh [-1246AaCfGgKkMNnqsTtVvXxYy] [-b bind_address] [-c cipher_spec]
    [-D [bind_address:]port] [-E log_file] [-e escape_char]
    [-F configfile] [-I pkcs11] [-i identity_file]
    [-J [user@]host[:port]] [-L address] [-l login_name] [-m mac_spec]
    [-O ctl_cmd] [-o option] [-p port] [-Q query_option] [-R address]
    [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
    [user@]hostname [command]
```

A detailed description of the operands is provided in the OpenSSH man pages.

ssh sets up the connection to the computer `hostname` and logs on there under the (optional) user name `user`. The user must prove his/her identity to the remote computer. Details on this are provided in the next section “Authentication between OpenSSH client ssh and server sshd” on page 332. If `command` is specified, this command is executed in the remote shell instead of in the user’s login shell.

Return value

If successful: Exit status of the command executed remotely

If not successful: 255

7.2.3 Authentication between OpenSSH client ssh and server sshd

Client authentication

When using the default value for the *PreferredAuthentication* option in the client configuration file *ssh_config* (see section “Configuring the OpenSSH client ssh” on page 329), the OpenSSH client executes the following authentication methods one after the other:

1. host based authentication
2. Public key authentication
3. Password authentication

The methods are applied one after the other until a method has successfully provided authentication or until all methods have failed.

Public key authentication permits the use of RSA, DSA ECDSA and Ed25519 algorithms. The OpenSSH client signs the session ID together with other data with its private key (*$HOME/.ssh/id_rsa*, *$HOME/.ssh/id_dsa*, *$HOME/.ssh/id_ecdsa* or *$HOME/.ssh/id_ed25519*) and sends the result to the OpenSSH server. The server checks whether a corresponding public key is contained in the file *<user home>/.ssh/authorized_keys* file. If yes the server accepts the connection. *<user home>* is the home directory of the user with whose user ID the *ssh* caller wishes to log in.

SSH protocol is using (amongst others) the following mechanisms which ensure confidentiality and integrity of the connection:

- Confidentiality is guaranteed by encrypting the data traffic using AES, ChaCha20 or 3DES.
- Integrity is guaranteed by encrypting the data traffic using mac-sha2, hmac-sha1 or umac.

Server authentication

The OpenSSH client authenticates the server by checking whether a public key is stored for the respective system in the user's file *$HOME/.ssh/known_hosts* or in the file provided centrally by the system administrator */etc/ssh/ssh_known_hosts* and, if this is the case, whether it matches the host key type sent from *sshd* (RSA/DSA/ECDSA/Ed25519).

The *StrictHostKeyChecking* option in the configuration file *ssh_config* controls the behavior of the client in the event that no suitable entry is found in the *known_hosts* files:

– If *no* is returned, the previously unknown host key is entered in *$HOME/.ssh/known_hosts* without requesting confirmation. If *ask* is returned, the user is asked whether the host key is to be entered.

– If *yes* is returned, the host key is never entered by the client but must be entered in the respective *known_hosts* file by the user or system administrator instead.

A detailed description of the *StrictHostKeyChecking* options is provided in the OpenSSH man pages.

7.2.4 Command execution and data forwarding

As soon as the OpenSSH client (*ssh*) has successfully authenticated itself a dialog begins for preparing the OpenSSH session. At this point the client can, for example, request the following activities:

- Setup of a pseudo-tty
- Forwarding of TCP/IP connections
- Forwarding of the connection to an authentication agent (*ssh-agent*, see page 345) via a secure channel

Finally *ssh* requests either a shell or execution of a command. Client and server are in session mode. In this mode each side can send data at any time. This data is then forwarded as follows:

- from the user terminal on the client side to the shell / command on the server side, or
- from the shell / command on the server side to the user terminal on the client side.

7.2.5 Login session and command execution on a remote computer

As soon as the OpenSSH server has accepted the user's identity, the server either executes the current command on the remote computer or it makes the remote computer's normal shell available to the user. Here all communication with the remotely executed command or the remote shell is automatically encrypted.

A distinction must be made between the following cases:

- If no pseudo-terminal (pseudo-tty) was assigned, the session is transparent and can be used for the secure transfer of binary data.
- If a pseudo-terminal (pseudo-tty) was assigned (normal login), the user can use escape characters (see the next section "Escape characters").

On most systems disabling the escape characters leads to a transparent session even if a pseudo-tty is used. (Use of escape characters is controlled using the *EscapeChar* option in the client configuration file *ssh_config* (see page 330).)

The session is terminated when command execution or the shell on the remote computer is terminated and all TCP/IP connections have been terminated. The exit status of the program executed remotely is returned as the exit status of *ssh*.

7.2.6 Escape characters

If a pseudo-tty is requested, *ssh* supports a raft of functions by using escape characters.

The following escape characters are supported (default: “~”):

~.	Close down connection.
~^Z	Execute *ssh* in the background.
~#	List forwarded connections.
~&	Execute *ssh* in the background in the event of a logout as long as forwarded connections are still active.
~?	Output list of escape characters.
~B	Send BREAK to the remote system.
~C	Open command line. (Only meaningful for additional port forwardings with the -L and -R options). OpenSSH itself “prompts” with a mini input line in which you can enter, for example, `-L ... <RETURN>`.
~R	Request reencryption of the session.

A single “~” (tilde) character can be sent as “~~” or as “~” followed by a character other than one of those explained above. To permit an escape character to be interpreted as a special character, the escape character must always be at the beginning of a new line. You can change the escape character using the *EscapeChar* option in the configuration file *ssh_config* (see page 330) of the OpenSSH client *ssh* or using the command line parameter *-e* in the *ssh* start call.

7.2.7 Port forwarding (TCP forwarding)

You can specify the forwarding of any TCP/IP connections via a secure channel either as a command line parameter in the *ssh* start call or in the configuration file *ssh_config* (see page 330). Possible applications of port forwarding include, for example, a secure connection to an account when electronic banking is used and tunnelling by a firewall.

7.2.8 ssh environment variables

The OpenSSH client *ssh* sets a number of environment variables. Details on these variables are provided in the OpenSSH man pages.

7.2.9 ssh files

In addition to the configuration files (see page 329), the OpenSSH client *ssh* also uses other files, and these are described below.

$HOME/.ssh/known_hosts

This file contains the associated host keys for all hosts on which the user has logged on, provided these keys are not contained in the */etc/ssh/ssh_known_hosts* file.

The user-specific files *$HOME/.ssh/known_hosts* can be maintained automatically in the file *ssh-config* depending on the *StrictHostKeyChecking* option settings (unknown hosts are entered automatically upon initial contact) or must be supplied manually.

$HOME/.ssh/id_rsa
$HOME/.ssh/id_dsa
$HOME/.ssh/id_ecdsa
$HOME/.ssh/id_ed25519

These files contain the user's authentication data, i.e. the private DSA or RSA key. The files contain data which is relevant to security and may only be readable for the owner. No other users may have read, write or execution permission for the files. Note that *ssh* ignores private keys if these are accessible to other users. When generating the key you can specify a passphrase through which the sensitive part of the files is encrypted with AES.

$HOME/.ssh/id_rsa.pub
$HOME/.ssh/id_dsa.pub
$HOME/.ssh/id_ecdsa.pub
$HOME/.ssh/id_ed25519.pub

These files contain the public key for authentication in plaintext, in other words the public part of the files *$HOME/.ssh/id_rsa*, *$HOME/.ssh/id_dsa*, *$HOME/.ssh/id_ecdsa* and *$HOME/.ssh/id_ed25519*:

- The content of the files*$HOME/.ssh/id_rsa.pub*, *$HOME/.ssh/id_dsa.pub*, *$HOME/.ssh/id_ecdsa.pub* and *$HOME/.ssh/id_ed25519.pub* should be added to the *$HOME/.ssh/authorized_keys* file on all computers on which users wish to authenticate themselves via RSA/DSA/ECDSA/Ed25519 and log on.

The *$HOME/.ssh/id_rsa.pub*, *$HOME/.ssh/id_dsa.pub*, *$HOME/.ssh/id_ecdsa.pub* and *$HOME/.ssh/id_ed25519.pub* files contain no sensitive data and can (but need not) be readable for everyone. As these files are never used automatically they are not mandatory, but are offered to the user merely to simplify handling.

$HOME/.ssh/authorized_keys

This file contains all public keys (RSA/DSA/ECDSA/Ed25519) which the user can use for the login. The file format is described under sshd (8) in the OpenSSH man pages. In its simplest form, the format of *$HOME/.ssh/authorized_keys* corresponds to the format of the files *$HOME/.ssh/id_rsa.pub*, *$HOME/.ssh/id_dsa.pub*, *$HOME/.ssh/id_ecdsa.pub* and *$HOME/.ssh/id_ed25519.pub*.

Only the user may have write access to this file; read access by third parties is non-critical. Nevertheless, it is recommended to make the file as a whole accessible to the user only (read and write). The file should be inaccessible for all other users.

/etc/ssh/ssh_known_hosts

This file contains a list of all host keys that are known throughout the system. The system administrator should prepare the file to incorporate the public host keys of all available computers in the organization. The file should be readable for everyone.

Each file line contains the following parts which are separated by blanks:

– Name(s) of the host; multiple names for the same host are separated by commas when specified
– Public key
– Comment (optional).

The file format is described under sshd (8) in the OpenSSH man pages.

The client uses this file in addition to the user-specific file *$HOME/.ssh/known_hosts* to ensure that it is connected to the intended remote host.

/etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_ecdsa_key
/etc/ssh/ssh_host_ed25519_key

These files contain the private sections of the host keys and are used for rhosts authentication (Hostbased Authentication):

- In the case of rhosts authentication (see page 332), *ssh* uses the *ssh-keysign* (8) utility to access the host key. (*ssh-keysign* (8) is described in the OpenSSH man pages.) The effective user ID (setuid) of *ssh* then need not necessarily be the root authorization. By default *ssh* does not have the root authorization as effective authorization.

$HOME/.rhosts

This file contains a list of the host/user pairs which are permissible for a login and which are required for rhosts authentication. Note that this file is also used by *rlogin* and *rsh* and is therefore not secure.
Each line in the *$HOME/.rhosts* file contains a host name in canonical form (as supplied by the name server) and a user name on this host. The host name and user name are separated by a blank.
If the user's home directory is located in an NFS partition, it can be that the file must be readable for all on some computers because the server daemon *sshd* reads the file as root. Furthermore, the user must be the owner of this file, and no other user may have write permission for the file. For most computers the recommended permissions are read/write for the user and no access right for all other users.

By default *sshd* is set so that is permits rhosts authentication only after RSA host authentication has been successfully completed. If the client's host key does not exist in the */etc/ssh/ssh_known_hosts* file on the server system, it can be stored in the *$HOME/.ssh/known_hosts* file. This is done most simply by using *ssh* to set up a connection from the server system to the client. The host key is then automatically stored in *$HOME/.ssh/known_hosts*.

$HOME/.shosts

This file is used in exactly the same way as *$HOME/.rhosts*. The *$HOME/.shosts* file enables rhosts authentication to be used without permitting a login via *rlogin* or *rsh*.

/etc/hosts.equiv

This file is used during rhosts authentication. It contains the host names in canonical form. A host name is contained in each line. A complete description can be found under *sshd* (8) in the OpenSSH man pages. If the client's host is included in the file and the user name matches on client and server, the login is permitted automatically provided a successful RSA-host authentication is not also required. The RSA-host authentication is usually required.
Only users with root authorization should be able to write to the file.

/etc/ssh/shosts.equiv

This file is used in exactly the same ways as the */etc/hosts.equiv* file. */etc/ssh/shosts.equiv* can be helpful to permit a login with *ssh*, but not with *rsh* or *rlogin*.

/etc/ssh/sshrc

This file contains commands which *ssh* performs when logging in the user before the user shell (or the user command) is started (see also the section “Login session and command execution on a remote computer” on page 333). A complete description of the */etc/ssh/sshrc* file can be found under sshd (8) in the OpenSSH man pages.

$HOME/.ssh/rc

This file contains commands which *ssh* performs when login in the user before the user shell (or the user command) is started (see also the section “Login session and command execution on a remote computer” on page 333). A complete description of the */etc/ssh/sshrc* file can be found under sshd (8) in the OpenSSH man pages.

$HOME/.ssh/environment

This file contains additional definitions for environment variables.

7.3 OpenSSH client applications scp and sftp

The OpenSSH client applications *scp* and *sftp* are programs which you can use for the secure copying of files and for secure file transfer between computers in a network.

7.3.1 scp - Secure copying of files between computers in the network

Syntax

```
scp [-12346BCpqrv] [-c <cipher>] [-F <ssh_config>] [-i <identity_file>]
    [-l <limit>] [-o <ssh_option>] [-P <port>] [-S <program>] [-X binary]
    [[<user>@]<host1>:]<file1> [...] [[<user>@]<host2>:]<file2>
```

A detailed description of the operands is provided in the OpenSSH man pages.

Description

scp enables you to copy files between computers in a network. In contrast to the r utility *rcp*, *scp* inquires passphrases and passwords if these are required for authentication.

Each file name can contain details on the computer and user to show that this file is to be copied from / to the computer concerned.

The `-X binary` option can be used for connections to ASCII servers in order to transfer files in binary format (bit-identical). For connections to EBCDIC servers, this option must be omitted in order to get a bit-identical transfer.

Return value

If successful: 0

If not successful: < 0

7.3.2 sftp - Secure file transfer between computers in a network

Syntax

```
sftp [-1246aCfpqrv] [-B <buffer_size>] [-b <batchfile>] [-c <cipher>]
   [-D <sftp_server_path>] [-F <ssh_config>] [-i <identity_file>] [-l <limit>]
   [-o <ssh_option>] [-P <port>] [-R <num_requests>] [-S <program>]
   [-s <subsystem> | <sftp_server>] [-X binary] <host>

sftp [-X binary] [[<user>@]<host>[:<file> [<file>]]]

sftp [-X binary] [[<user>@]<host>[:<dir>[/]]]

sftp [-X binary] -b <batchfile> [<user>@]<host>
```

A detailed description of the operands is provided in the OpenSSH man pages.

Description

sftp is an interactive program for file transfer (like FTP) which handles all operations at TCP level via an encrypted *ssh* channel. In addition, *sftp* can utilize many features of *ssh* such as public key authentication and data compression. *sftp* establishes the connection to the specified computer and logs on there. Afterward *sftp* is in interactive mode for command input.

The `-X binary` option can be used for connections to ASCII servers in order to transfer files in binary format (bit-identical). In interactive mode the `type [text|binary]` command (see page 343) can also be used for this purpose. For connections to EBCDIC servers, this option must be omitted in order to get a bit-identical transfer.

Transferring files from a remote directory

The command format

```
sftp [-X binary] [[<user>@]<host>[:<file> [<file>]]]
```

enables you to use *sftp* to transfer files automatically from a directory on a remote computer to the local directory. A prerequisite for this is that non-interactive authentication is used, otherwise file transfer is only possible after successful interactive authentication.

Starting sftp in a remote directory

The following command format enables you to start *sftp* in a directory on a remote computer:

```
sftp [-X binary] [[<user>@]<host>[:<dir>[/]]]
```

Automated sftp sessions

The command format

```
sftp [-X binary] -b <batchfile> [<user>@]<host>
```

permits automatic *sftp* sessions using the `-b` option. In this case public key authentication is generally required to bypass the need to enter a password for connection setup (see the section “Authentication between OpenSSH client ssh and server sshd” on page 332).

Interactive commands

In interactive mode *sftp* provides you with a number of interactive commands which are similar to those of FTP.

The following must be borne in mind for the command notation:

- It is not case-sensitive.
- Path names containing blanks must be enclosed in single or double quotes.

`bye`
: Terminates *sftp*.

`cd <path>`
: Changes the directory on the remote computer in <path>.

`chgrp <group> <path>`
: Changes the group ID of the <path> file in <group>.
<group> must be a numerical group ID.

`chmod <mode > <path>`
: Changes access rights of the <path> file on <mode>.

`chown <own> <path>`
: Changes owner of the <path> file to <own>. <own> must be a numerical UID.

`exit`
: Exits *sftp*.

`get [<flags>] <remote-path> [<local-path>]`
: Transfers the <remote-path> file from the remote computer to the local computer and stores it under <local-path>. If <local-path> is not specified, the file is stored under the same name as on the remote computer.
If the `-P` flag is specified, all access rights and the time the file was accessed are also copied.

`help`
Displays help text (brief reference material for the commands).

`lcd <path>`
Changes local directory in <path>.

`lls [<ls-options> [<path>]]`
Displays local directory. Either displays <path> or, if <path> is not specified, displays the current directory.

`lmkdir <path>`
Creates the local directory specified by <path>.

`ln <oldpath> <newpath>`
Creates symbolic link from <oldpath> to <newpath>.

`lpwd`
Prints local working directory.

`ls [<flags>] [<path>]`
Displays the contents of the remote directory. Displays either contents of <path> or, if <path> is not specified, displays the contents of the current directory. If the `-l` flag is not specified, displays additional information, including permissions and ownership.

`lumask <mask>`
Changes local umask (access rights) to <mask>.

`mkdir <path>`
Creates remote directory specified by <path>.

`progress`
Enables/disables progress display.

`put [<flags>] <local-path> [<remote-path>]`
Uploads <local-path> file and transfers it to the remote computer. If <remote-path> is not specified, the file on the remote computer is given the same name as on the local computer. If the *-P* flag is specified, all the file's access rights and the time the file was accessed are also copied.

`pwd`
Displays remote working directory.

`quit`
Quits *sftp*.

`rename <oldpath> <newpath>`
Renames remote file from <oldpath> to <newpath>.

`rm <path>`
Removes remote file <path>.

`rmdir <path>`
Removes remote directory <path>.

`symlink <oldpath> <newpath>`
Creates symbolic link from <oldpath> to <newpath>.

`type [binary|text]`
Sets transfer mode (for connections to ASCII computers) to “binary” or resets it to “text”. If no argument is specified, the current mode is displayed (see also call option `-X binary`).

`version`
Displays protocol version of *sftp*.

`! <command>`
Executes command <command> in the local shell.

`!`
Generates local subshell.

`?`
Displays help text (brief reference material for the commands) (same as `help`, see above).

7.4 OpenSSH Basic utilities

OpenSSH provides the basic utilities described below to support user-friendly, automated client/server authentication:

- *ssh-agent* - Authentication agent

  In the case of client authentication (see page 332) the private keys are always stored in files in encrypted form for security reasons, and consequently you must enter the associated password each time you use the private key. In this case and in others, too, the authentication agent facilitates the handling of the private keys.

- *ssh-add* - Loading private RSA/DSA keys in the authentication agent

  After it has been started the authentication agent initially receives no private RSA/DSA keys. You load these in the authentication agent using the *ssh-add* utility.

- *ssh-keygen* - Generating and administering a user-specific key pair (RSA/DSA)

  In the case of RSA/DSA-based client authentication, each user requires a key pair comprising a private and a public RSA or DSA key. You can generate and administer such key pairs with the *ssh-keygen* utility.

- Creating and checking *ssh-keyscan* - *ssh_known_host* files

  The *ssh-keyscan* utility enables you to query the public SSH host keys of different hosts from any OpenSSH client and to transfer these to the *ssh_known_hosts* files. Furthermore, *ssh-keyscan* supports you in checking existing *ssh_known_hosts* files.

7.4.1 ssh-agent - Authentication agent

Syntax

```
ssh-agent [-a <bind_address>] [-c | -s] [-t <life>]
          [-d] [<command> <[args> ...]]

ssh-agent [-c | -s] -k
```

A detailed description of the operands is provided in the OpenSSH man pages.

Description

The authentication agent *ssh-agent* is a utility for administering the private keys for public key authentication (RSA, DSA, ECDSA, Ed25519). Initially *ssh-agent* has no key. You can use the *ssh-add* utility (see page 347) to assign keys to *ssh-agent*. *ssh-add* adds the files *$HOME/.ssh/id_rsa*, *$HOME/.ssh/id_dsa*, *$HOME/.ssh/id_ecdsa* and *$HOME/.ssh/id_ed25519* to the authentication agent.

It makes sense to start *ssh-agent* on your local PC, notebook or terminal at the start of a login session. All other windows and programs are then executed as clients of *ssh-agent*. Environment variables enable the clients to locate the *ssh-agent* and use it for automatic authentication if they log onto remote computers with the OpenSSH client *ssh*.

Executing *ssh-agent* on your local PC, notebook or terminal has the advantage that *ssh-agent* never sends authentication data over the channel via which it receives the requests. Instead, *ssh-agent* itself performs operations which require a private key and returns the result to the computer from which the request came. This behavior is referred to as agent forwarding.

If you call authentication agents with the format

```
eval ‘ssh-agent [-c | -s]‘
```

ssh-agent is run as a background process (daemon) and generates shell commands as its output. These commands enable the shell and its child processes to utilize the authentication agent’s services.

You terminate the *ssh-agent* daemon with the following command:

eval ‘ssh-agent -k [-c | -s] ‘

If you call authentication agents with the format

ssh-agent [<command> [<args> ...]]

the command <command> and all child processes of <command> can utilize the authentication agent. After <command> has been executed, the authentication agent is terminated automatically.

If you call the authentication agent without parameters, it generates a Unix domain socket and listens there in the background for user requests.

Files

ssh-agent uses the following files:

$HOME/.ssh/id_dsa

Contains the user's private DSA key for authentication.

$HOME/.ssh/id_rsa

Contains the user's private RSA key for authentication.

$HOME/.ssh/id_ecdsa

Contains the user's private ECDSA key for authentication.

$HOME/.ssh/id_ed25519

Contains the user's private Ed25519 key for authentication.

/tmp/ssh-XXXXXXXX/agent.<ppid>

Unix domain sockets which contain the connections to the *ssh-agent*. Only the owner should have read and write permissions for these sockets; no other user should have read or write permission. When *ssh-agent* terminates the sockets are automatically removed.

7.4.2 ssh-add - Loading private keys in the authentication agent

Syntax

```
ssh-add [-lLdDxXc] [-t <life>] [<file> ...]
ssh-add -s <reader>
ssh-add -e <reader>
```

A detailed description of the operands is provided in the OpenSSH man pages.

Description

ssh-add provides the authentication agent *ssh-agent* with private RSA, DSA, ECDSA or Ed25519 keys. If called without arguments, *ssh-add* adds the files *$HOME/.ssh/id_rsa*, *$HOME/.ssh/id_dsa*, *$HOME/.ssh/id_ecdsa* und *$HOME/.ssh/id_ed25519* to the authentication agent. You can specify alternate file names in the command line.

If a file is protected by a passphrase, *ssh-add* requests the user to enter a passphrase, which *ssh-add* then reads in from the user's terminal. If there are multiple secret RSA or DSA keys, *ssh-add* attempts to reuse the last passphrase read in.

`ssh-add -l` enables you to have the keys currently administered by *ssh-agent* to be displayed.

A prerequisite for executing *ssh-add* is that the authentication agent *ssh-agent* has been started and that the name of its socket is contained in the environment variable *SSH_AUTH_SOCK*. The environment variable *SSH_AUTH_SOCK* is set automatically when *ssh-agent* is started.

Return value

If successful: 0

If the command specified could not be executed: 1

If *ssh-add* could not set up a connection to *ssh-agent*: 2

Environment variable ssh-add

SSH_AUTH_SOCK
: Identifies the path name of the socket of a domain in the Unix system which is used for communicating with the authentication agent *ssh-agent*.

ssh-add files

ssh-add uses the following files:

$HOME/.ssh/id_dsa

Contains the user's private DSA key for authentication.

$HOME/.ssh/id_rsa

Contains the user's private RSA key for authentication.

$HOME/.ssh/id_ecdsa

Contains the user's private ECDSA key for authentication.

$HOME/.ssh/id_ed25519

Contains the user's private Ed25519 key for authentication.

ssh-add ignores the aforementioned files if they can be accessed by other user IDs. `chmod go-rwx ...` enables you to block the files against access by other users.

7.4.3 ssh-keygen - Generating and administering an RSA/DSA key pair

Syntax

```
ssh-keygen [-q] [-b <bits>] -t <type> [-N <new_passphrase>] [-C <comment>]
           [-f <output_keyfile>]
ssh-keygen -p [-P <old_passphrase>] [-N <new_passphrase>] [-f <keyfile>]
ssh-keygen -i [-f <input_keyfile>]
ssh-keygen -e [-f <input_keyfile>]
ssh-keygen -y [-f <input_keyfile>]
ssh-keygen -c [-P <passphrase>] [-C <comment>] [-f <keyfile>]
ssh-keygen -l [-f <input_keyfile>]
ssh-keygen -B [-f <input_keyfile>]
ssh-keygen -D <reader>
ssh-keygen -U <reader> [-f <input_keyfile>]
ssh-keygen -r <hostname> [-f <input_keyfile>] [-g]
ssh-keygen -G <output_file> [-v] [-b <bits>] [-M <memory>] [-S <start_point>]
ssh-keygen -T <output_file> -f <input_file> [-v] [-a <num_trials>]
           [-W <generator>]
```

A detailed description of the operands is provided in the OpenSSH man pages.

Description

Each authentication algorithm requires its own key pair comprising a private and a public key. The *ssh-keygen* utility enables you to create such a key pair (RSA, DSA, ECDSA or Ed25519). You specify the authentication algorithm with the `-t` parameter.

ssh-keygen generates, administers and converts authentication keys for the OpenSSH client *ssh*. *ssh-keygen* can generate RSA or DSA keys.

Generally a user who wants to use OpenSSH with RSA or DSA authentication will start *ssh-keygen* in order to generate the authentication key in *$HOME/.ssh/id_rsa*, *$HOME/.ssh/id_dsa*, *$HOME/.ssh/id_ecdsa* oder *$HOME/.ssh/id_ed25519*. In addition, the system administrator can generate hot keys using *ssh_keygen*.

Private keys are saved under *$HOME/.ssh/<identity>*, public keys under *$HOME/.ssh/<identity>.pub* in the user's directory. Here *<identity>* stands for

id_dsa (`-t dsa`) for DSA and
id_rsa (`-t rsa`) for RSA and
id_ecdsa (`-t ecdsa`) for ECDSA and
id_ed25519 (`-t ed25519`) for Ed25519

You can also give the file a different name:

- When the keys are generated, *ssh-keygen* inquires the name of the file in which the private key is to be stored.
- The file name for the public key is automatically adapted and given the extension ".pub".

Further information on *ssh-keygen* can be found in the OpenSSH man pages.

Files

A description of the *ssh-keygen* files is provided in the OpenSSH man pages.

7.4.4 ssh-keyscan

Syntax

```
ssh-keyscan [-v46H] [-p <port>] [-T <timeout>] [-t <type>] [-f <file>]
            [<host> | <addrlist> <namelist>] [...]
```

A detailed description of the operands is provided in the OpenSSH man pages.

Description

The *ssh-keyscan* utility enables you to inquire the public SSH host keys of different hosts from any OpenSSH client and to transfer it directly into the *ssh_known_hosts* files. In addition, *ssh-keyscan* supports you in verifying existing *ssh_known_hosts* files. *ssh-keyscan* provides a minimum interface which is equally suitable for shell and Perl scripts.

ssh-keyscan uses non-blocking socket input/output in order to connect simultaneously to as many computers as possible.

To scan the computers for host keys *ssh-keyscan* does not need a login authorization on these computers. Nor does the scanning process include any form of encryption.

Security aspects

If you create an *ssh_known_hosts* file with the aid of *ssh-keyscan*, you are vulnerable to “man-in-the-middle” attacks. If, on the other hand, the basic security model allows such risks, *ssh-keyscan* can support you in detecting counterfeited host key files. *ssh-keyscan* also assists you in detecting “man-in-the-middle” attacks which were started after the *ssh_known_hosts* file was created.

7.5 BS2000-specific special aspects

When working with OpenSSH in a BS2000 environment, the special aspects described below must be borne in mind.

Prompt when the password is empty

Normal Unix systems do not request a password if they use *login* or *slogin* to log into an ID without a password. However, the POSIX *rlogin* requests a password to be entered even for an ID without a password. This behavior does not result in increased security, though, because it is at the same time possible to issue an *rsh* command without a password for the same ID.

Here OpenSSH behaves like the other Unix systems and does not ask for a empty password. However, as a login to IDs without a password is, by default, blocked in OpenSSH, in this case you must set the *PermitEmptyPasswords* directive in the configuration file */etc/ssh/sshd_config* to “yes”.

Upper/lower case in the user name

Unlike in Unix operating system, no distinction is made between upper and lower case in BS2000 and OSD/POSIX. Thus in BS2000 and OSD/POSIX the user “Username” can log on as “username”, “USERNAME” or “uSeRnAmE”. The name of the user who has logged on is recorded in the */var/adm/utmp* file. You can use the `who` command to have the user name displayed.

Whereas rlogin enters the user name in upper case, OpenSSH specifies the user name in lower case (as is usual in Unix operating systems).

Accessing DVS files via sftp

sftp can also access DVS files (SAM and PAM). To do so, add /BS2/ as a prefix to the respective file name when using the get or put command. In the case of files to be written, it is not generally implicitly clear which file properties they are to be given. An option based on the *file* command for the FTP is therefore available (see page 133), which enables you to specify these file properties by appending a list of operands to the file names separated by a comma.

Example

```
put srcfile /BS2/DESTFILE.PAM,FCBTYPE=PAM
put srcfile /BS2/DESTFILE.SAM,FCBTYPE=SAM,RECF=V
get /BS2/SRC.LMS /BS2/DEST.LMS,FCBTYPE=PAM
```

8 Mail reader in BS2000

You can use the mail reader to pick up and process e-mails in BS2000 via the access services (POP3 and IMAP).

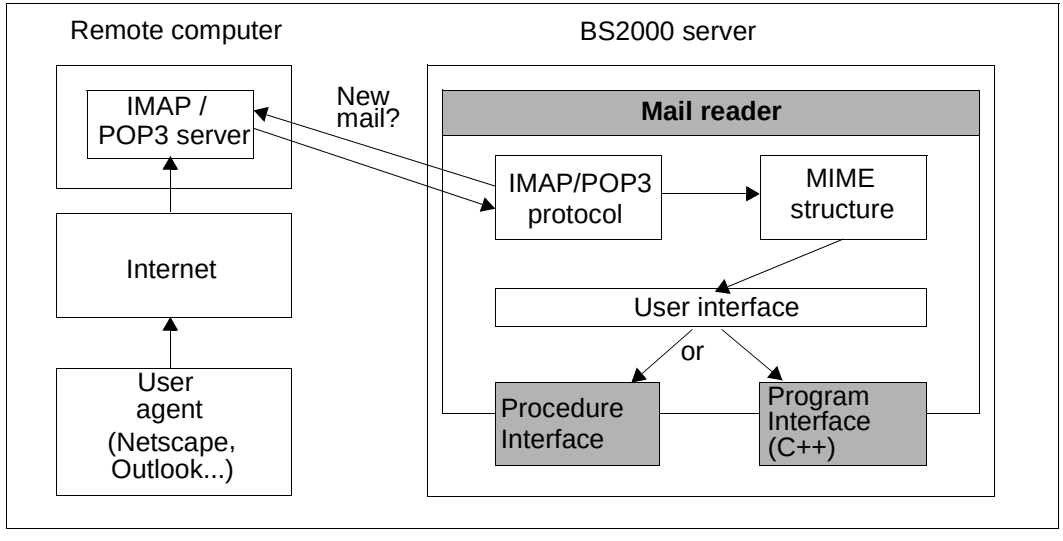


Figure 8: Mail reader in BS2000

The mail reader offers you two options for further processing in BS2000:

– The procedure interface
– The programming interface

The mail reader provides you with two options for processing in BS2000:

Procedure interface

The mail reader provides you with the option of using procedures to access e-mail message headers, message body texts and attachments.

Programming interface

Via the C++ interface, an instance of a C++ class `DwMessage` containing the complete e-mail is passed to a function. To do this, you must write a C++ subprogram and link it to the modules of the mail reader.

The open-source library “mimelib”, which provides classes for easy access to the e-mail, is used to represent the e-mail at the software level. This library is part of KDE (the *kdepim* package) under the GP license and is based on “mime++” by Doug Sander.

8.1 Starting/stopping the mail reader

Starting the mail reader

Use one of the following two commands to start the mail reader:

```
/START-MAILREADER
```

or

```
/START-PROGRAM *MODULE($.SYSPRG.MAIL.nnn,READER,*ANY,*ADVANCED)
```

Stopping the mail reader

Use one of the following two commands to stop the mail reader:

```
/INFORM-PROGRAM 'SHUTDOWN',*TSN(<tsn>)
```

or

```
/INFORM-PROGRAM 'CANCEL',*TSN(<tsn>)
```

These commands stop the mail reader in a controlled manner, i.e. after any ongoing accessing of a mailbox is completed.

The following conditions apply for the described /INFORM-PROGRAM commands:

– The commands must be entered via the console interface of the system user
– The mail reader must be executed as a batch task
– The TSN of the batch task must be specified

The user terminates the mail reader executed as a batch task as follows:

```
/CANCEL-JOB   JOB-IDENTIFICATION=*TSN(TSN=<tsn of the batch task>)
```

8.2 Configuration file

You use the *SYSDAT.MAIL.nnn.READER* configuration file to control the behavior of the mail reader. The configuration file is read when the mail reader starts up.

Structure of the configuration file

You use the configuration to define:

– The behavior of the e-mail reader in relation to POP3 or IMAP servers
– Event logging
– Processing of e-mail in BS2000

Accordingly, the configuration file is subdivided into the following parameter sections:

– SERVER
– TRACE
– MAILHANDLING

The sections that follow (starting on page 356) explain the syntax of these parameter sections with examples.

Lines in the configuration file that begin with # are treated as comments and ignored.

Changing the configuration of the mail reader using the configuration file

Changes to the *SYSDAT.MAIL.nnn.READER* configuration file do not affect the configuration of the mail reader until it is restarted.

If you want to change the configuration of the mail reader during operation, you can read the configuration file in again as follows:

- If the mail reader is executed as a batch task:
  - ► Enter the following command:

    ```
    /INFORM-PROGRAM 'RECONFIG',*TSN <TSN of the batch task>
    ```
- If the mail reader is executed as a dialog task (for test purposes, for example):
  - ► Interrupt the execution of the mail reader by pressing the K2 key.
  - ► Enter the following command:

    ```
    /INFORM-PROGRAM MSG='RECONFIG',JOB-IDENTIFICATION=*OWN
    ```
  - ► Enter the following command to resume execution of the mail reader:

    ```
    /RESUME-PROG
    ```

8.2.1 Mail processing: MAILHANDLING parameter section

In the MAILHANDLING parameter section you define how the received mail is to be handled.

Two files are created for each mail:

- Procedure file
  This contains the statements that are to be executed when a mail is received.
- Message file
  This contains the received message.

You accordingly define the procedure file structure, with the PROCEDURE parameter record. With the MESSAGEBODY parameter record you define the structure of the file that contains the actual message.

Syntax

```
MAILHANDLING = PARAMETERS ( ... )

    PARAMETERS ( ... )
        |  PREFIX=<string>
        |  ,ENTER=JOB / PROCEDURE
        |  ,DELETE=YES / NO
        |  ,SMIME=PARAMETERS( ... )
        |      PARAMETERS( ... )
        |          |  KEY=<private key file>
        |          |  ,CERTIFICATE=<certificate file>
        |          |  ,CRL=<CRL file>
        |          |  ,CA_CERTIFICATES=<CA certificates file>
        |          |  ,VERIFY_SIGNATURE=YES / NO
        |          |  ,VERIFY_DEPTH=<depth>
        |  ,BODY=PARAMETERS( ... )
        |      PARAMETERS( ... )
        |          |  PROCEDURE=PARAMETERS( ... )
        |          |      PARAMETERS( ... )
        |          |          |  SUFFIX=<string>
        |          |          |  ,TEXT=(<textlist>)
        |          |          |  ,ATTACHMENT=(<textlist>)
        |          |  ,MESSAGEBODY=PARAMETERS( ... )
        |          |      PARAMETERS( ... )
        |          |          |  TEXT=(<textlist>)
        |  ,ATTACHMENT=PARAMETERS( ... )
        |      PARAMETERS( ... )
        |          |  PROCEDURE=(<textlist>)
        |          |      ,MESSAGEBODY=(<textlist>)
```

Description of parameters

PREFIX=<string>
Prefix for the files created. The files are named *PREFIX.YYYY-MM-DD.HHMMSS*. If several files are created within one second, they are incremented with *#nnn*.
Default setting: MAIL

ENTER=JOB / PROCEDURE
Specifies whether the procedure file is to be started with */ENTER-JOB-* or */ENTER-PROCEDURE*.

ENTER=JOB
Starts the ENTER procedure with */ENTER-JOB*.

ENTER=PROCEDURE
Starts the ENTER procedure with */ENTER-PROCEDURE*.

DELETE=YES / NO
Specifies whether the procedure file is to be deleted after the execution of the ENTER procedure.

DELETE=YES
The procedure file is deleted after the execution of the ENTER procedure.

DELETE=NO
The procedure file is **not** deleted after the execution of the ENTER procedure.

SMIME=PARAMETERS(...)
Specifies how e-mails signed in accordance with the S/MIME standard and/or encrypted are to be handled. Only S/MIME Version 2 is supported.

KEY=<private key file>
Specifies the file containing the private keys required for the decryption of encrypted e-mails. It is essential that this private key remains secret. The key must be stored in PEM format.

CERTIFICATE=<certificate file>
Specifies the file containing the X.509 certificate belonging to the private key. The private certificate must be stored in PEM format.

CRL=<CRL file>
Specifies the file containing a Certificate Revocation List (CRL). The CRL is used with signed e-mails to check whether the certificates used with the signature are still valid. For this purpose, the CRLs are to be obtained from the relevant at regular intervals (of a few weeks, typically) and stored in a shared file whose name is specified by means of this operand. On a purely technical level, verification also works without the use of a CRL. However, if you cannot ensure in any other way that the certificates used are still valid, you should create and specify a CRL file as described here.

CA_CERTIFICATES=<CA certificates file>
Specifies a file containing the CA certificates required for the verification of S/MIME signatures. The X.509 certificates must be stored in PEM format and arranged sequentially in the file.

To add or delete certificates, you can edit the file with any text editor. Each certificate is entered in the file as follows:

```
-----BEGIN CERTIFICATE-----
< CA certificate in Base64 code >
-----END CERTIFICATE-----
```

Text outside these sequences is ignored by the mail reader and can therefore be used to identify the certificates that are not available in a readable form because of their ASN.1/Base64 coding.

VERIFY_SIGNATURE=YES / NO
Specifies whether an existing signature is to be checked with the help of the CA certificates and the CRL.

VERIFY_SIGNATURE=YES
The existing signature is checked.

VERIFY_SIGNATURE=NO
The existing signature is **not** checked.

VERIFY_DEPTH=<depth>
Specifies the verification depth, which is the maximum number of permissible certificates between the certificate of the S/MIME signer and the certificate known to the mail reader. If the maximum verification depth is exceeded, the verification is considered to have failed.

Note the following:

1 The signer's certificate is not accepted unless it has been signed directly by one of the certificate authorities (CA) known to the mail reader.

0 This makes no sense because in these cases only self-signed certificates are permissible.

Default: 9

BODY=PARAMETERS(...)
The following parameters specify how the e-mail body is to be handled. All parts of the e-mail for which the *name* parameter is not assigned in the content type header are mail bodies.

PROCEDURE=PARAMETERS(...)
Specifies the contents of the procedure file to be created.

SUFFIX=<string>
Specifies the suffix, which distinguishes the procedure file from the message file.
Default: PROC

TEXT=(<textlist>)
The operand value is a comma-delimited list of texts, each of which is enclosed in single quotes. For each part of a non-multipart e-mail and each part of a multipart e-mail for which the *name* operand of the content type header is not assigned, this text list is inserted at the beginning of the procedure file (each item in the list appears in its own line). The parameters enclosed in percentage signs (%) specified in the section “Substituting mail-specific parameters” on page 373 are substituted. In particular, %FILE-NAME% is replaced with the name of the message file written. By default, this text list is empty.

ATTACHMENT=(<textlist>)
The operand value is a comma-delimited list of texts, each of which is enclosed in single quotes. For each part of a multipart e-mail for which the *name* operand of the content type header is not assigned, this text list is inserted in the procedure file (each item in the list appears in its own line). The parameters enclosed in percentage signs (%) specified in the section “Substituting mail-specific parameters” on page 373 are substituted. In particular, %ATTACHMENT-FILE-NAME% is replaced with the name of the attachment message file, and %ATTACHMENT-PROCEDURE-NAME% is replaced with the name of the procedure file created for the attachment. By default, this text list is empty.

MESSAGEBODY=PARAMETERS(...)
Specifies the structure of the message file.

TEXT=(<textlist>)
The operand value is a comma-delimited list of texts, each of which is enclosed in single quotes. For part of a non-multipart e-mail at the uppermost level and for each part of a multipart e-mail for which the *name* operand of the content type header is not assigned, this text list is inserted in the message file (each item in the list appears in its own line). The parameters enclosed in percentage signs (%) specified in the section “Substituting mail-specific parameters” on page 373 are substituted. In particular, %TEXT% is substituted with the message text. By default, this text list is empty.

ATTACHMENT=PARAMETERS(...)
Specifies the structure of the procedure file and message file to be generated for each attachment. The name of the files is generated in such a way that the value of the *name* parameter from the content type header field is appended to the name of the e-mail body. In addition, the suffix is appended to the attachment procedure file (see the SUFFIX operand).

PROCEDURE=(<textlist>)
Specifies the structure of the procedure file to be generated for each attachment. The operand value is a comma-delimited list of texts, each of which is enclosed in single quotes. This list of texts is inserted in the procedure file, with each item in the list on a separate line. The parameters enclosed in percentage signs (%) specified in the section “Substituting mail-specific parameters” on page 373 are substituted. By default, this text list is empty.

MESSAGEBODY=(<textlist>)
Specifies the structure of the message file to be generated for each attachment. The operand value is a comma-delimited list of texts, each of which is enclosed in single quotes. This list of texts is inserted in the message file, with each item in the list on a separate line. The parameters enclosed in percentage signs (%) specified in the section “Substituting mail-specific parameters” on page 373 are substituted. In particular, %TEXT% is substituted with the message text and %HEADER% with the header lines. By default, this text list is empty.

Example

The following example illustrates the MAILHANDLING parameter section of the configuration file.

```
MAILHANDLING=PARAMETERS(
    ENTER=PROCEDURE,
    BODY=PARAMETERS(
        PROCEDURE=PARAMETERS(
            TEXT=(
                '/WRITE-TEXT ''To: %To%''',
                '/WRITE-TEXT ''Subject: %SUBJECT%''',
                '/INCLUDE-PROCEDURE MAILPROC(FILENAME=%FILE-NAME%)'
            ),
            ATTACHMENT=(
                '/REMARK ****************************',
                '/REMARK * Now we call the procedure ',
                '/REMARK * handling the attachment   ',
                '/REMARK ****************************',
                '/INCLUDE-PROCEDURE %ATTACHMENT-PROCEDURE-NAME%,(-',
                '/                    CONTENTTYPE=''%CONTENT-TYPE%'')'
            )
        )
    ),
    ATTACHMENT=PARAMETERS(
        PROCEDURE=(
            '/BEG-PAR-DECL',
            '/DECLARE-PARAMETER CONTENTTYPE',
            '/END-PAR-DECL',
            '/CALL-PROCEDURE ATTPROC,(CT=''&(CONTENTTYPE)'')'
        )
    )
)
```

If you use the example shown above to receive an e-mail with an attachment, the mail reader creates the following procedure file:

– MAIL.2010-06-27.134758.PROC, as follows, for processing the message:

```
/WRITE-TEXT 'To: mail01@system'
/WRITE-TEXT 'Subject: This is a funny test'
/INCLUDE-PROCEDURE MAILPROC,(FILENAME=MAIL.2010-06-27.134758)
/REMARK ****************************
/REMARK * Now we call the procedure
/REMARK * handling the attachment
/REMARK ****************************
/INCLUDE-PROCEDURE MAIL.2010-06-27.134758.testtxt.PROC,(-
/                    CONTENTTYPE='text/plain; name="test.txt"')
```

– MAIL.2010-06-27.134758.TESTTXT.PROC, as follows, for processing the attachment:

```
/BEG-PAR-DECL
/DECLARE-PARAMETER CONTENTTYPE
/END-PAR-DECL
/CALL-PROCEDURE ATTPROC,(CT='&(CONTENTTYPE)')
```

To process the message, the MAIL.2010-06-27.134758.PROC procedure started by the ENTER-PROCEDURE command calls the MAILPROC procedure defined elsewhere with the name of the message file. To process the attachment, the MAIL.2010-06-27.134758.TESTTXT.PROC procedure is then called. This in turn calls the ATTPROC procedure defined elsewhere with the content type header of the attachment as a parameter. You will find more examples in the section “Structure of an e-mail” on page 377.

8.2.2 Event logging: TRACE parameter section

The TRACE parameter set allows you to specify which events are logged and where the trace entries are written.

Syntax

```
TRACE = PARAMETERS( ... )
        | SOCKET_TRACE = <level>
        | ,TLS = <level>
        | ,PROTOCOL = <level>
        | ,IO = <level>
        | ,SMIME = <level>
        | ,HEADER = <level>
        | ,LEVEL = <level>
        | ,PARSE = <level>
        | ,STRING = <level>
        | ,FILENAME = <trace file>
```

Operand description

SOCKET_TRACE = <level>
You can use this to activate the logging of the SOCKETS subsystem. <level> specifies the trace level. The higher the value specified for <level>, the more detailed the trace data output to SYSOUT. Level 0 deactivates the trace. The SOCKET_TRACE operand is independent of the LEVEL operand. By default, the socket trace is not activated.

TLS = <level>
Controls the logging of specific actions/events during the processing of the TLS/SSL protocol.

0	Logging deactivated (default).
1	Logging activated. A trace is only output when the LEVEL has the value 0.

PROTOCOL =<level>
Controls the logging of specific events of the POP3/IMAP protocol such as the fetching or deletion of an e-mail. <level> specifies the trace level: the higher the value specified for <level>, the more trace entries are written. A level of 0 suppresses all trace entries (default). The maximum trace level is 2. A trace is only output when the LEVEL operand has a value of 0.

IO=<level>
Allows you to activate logging of the data traffic with the POP3/IMAP server.

0 Logging deactivated (default).

1 Logging activated. A trace is only output when the LEVEL operand has a value of 0.

SMIME =<level>
Controls the logging of specific actions during S/MIME decryption and/or verification. <level> specifies the trace level. The higher the value for <level>, the more trace entries are written. A level of 0 suppresses all trace entries. The default value is 0. The maximum trace level is 2. A trace is only output when the LEVEL operand has a value of 0.

HEADER =<level>
Specifies the level for the logging of events that occur when procedure and message files are created. <level> specifies the trace level. The higher the value for <level>, the more trace entries are written. A level of 0 suppresses all trace entries. The default value is 0. The maximum trace level is 3. A trace is only output when the LEVEL operand has a value of 0.

PARSE =<level>
Specifies the level of the trace for events when the configuration file is read in. <level> specifies the trace level. The higher the value for <level>, the more trace entries are written. A level of 0 suppresses all trace entries. The default value is 0. The maximum trace level is 2.

LEVEL =<level>
Controls the output of the PROTOCOL, IO and HEADER traces and other events from different message classes in accordance with the following table:

0 TLS, PROTOCOL, IO, SMIME and HEADER traces and INFO, WARNING, ERROR and FATAL events are logged.

1 INFO, WARNING, ERROR and FATAL events are logged.
TLS, PROTOCOL, IO, SMIME and HEADER traces are **not** logged.

2 WARNING, ERROR and FATAL events are logged.
TLS, PROTOCOL, IO, SMIME and HEADER traces are **not** logged.

3 ERROR, FATAL events are logged.
TLS, PROTOCOL, IO, SMIME and HEADER traces are **not** logged.

FILENAME =<trace file>
Specifies the output file for the trace entries. If this parameter is not specified, the trace entries are output to SYSOUT.

Example

In this example, all the events of the POP3/IMAP protocol with a level of 1 are written to SYSOUT.

```
TRACE = PARAMETERS (
    PROTOCOL = 1,
    LEVEL = 0
)
```

8.2.3 POP3/IMAP servers: SERVER parameter section

The SERVER parameter set allows you to specify the POP3 or IMAP server from which the mail reader retrieves the e-mail.

The following entries are required:

– The host name of the computer on which the POP3 or IMAP server is running

– The user ID and password for this computer

– In the case of an IMAP server: the MAILBOX name as well

You also have the option of specifying how often the mail reader checks whether there are any new messages.

You can also request TLS/SSL encryption of the connection to the server so that the password to be transferred cannot be read by anyone.

Syntax

```
SERVER = POP3 ( ... ) / IMAP (...)
    POP3 ( ... )
        HOSTNAME=<hostname>
        ,PORT=<port number>
        ,USER=<user-id>
        ,PASSWORD=<password>
        ,INTERVAL=<int>
        ,TLS=NO / YES
            YES(. . . )
                REQUIRED=NO / YES
                ,MODE=IMPLICIT / EXPLICIT
                ,PROTOCOL=<protocol spec>
                ,CIPHER_SUITE=<cipher suite spec>
                ,CERTIFICATE=<certificate file>
                ,KEY=<private key file>
                ,CA_CERTIFICATES=<CA certificates file>
                ,CRL=<CRL file>
                ,VERIFY_SERVER=YES / NO
                ,VERIFY_DEPTH=<depth>
    ,IMAP ( ... )
        HOSTNAME=<hostname>
        ,PORT=<port number>
        ,USER=<user-id>
        ,PASSWORD=<password>
        ,KEEP=NO / YES
        ,MAILBOX=<mailbox>
        ,INTERVAL=<int>
        ,TLS=NO / YES
            YES(. . . )
                REQUIRED=NO / YES
                ,MODE=IMPLICIT / EXPLICIT
                ,PROTOCOL=<protocol spec>
                ,CIPHER_SUITE=<cipher suite spec>
                ,CERTIFICATE=<certificate file>
                ,KEY=<private key file>
                ,CA_CERTIFICATES=<CA certificates file>
                ,CRL=<CRL file>
                ,VERIFY_SERVER=YES / NO
                ,VERIFY_DEPTH=<depth>
```

Operand description

SERVER = POP3(. . .) / IMAP(. . .)
Specifies which protocol is to be used to access e-mails.

SERVER=POP3
The POP3 protocol is to be used.

SERVER=IMAP
The IMAP protocol is to be used.

HOSTNAME=<hostname>
The server runs on the host <hostname>. <hostname> can be the server's BCAM processor name or DNS name.

PORT=<port number>
Specifies the port number to be used to set up the TCP connection to the server. If this parameter is used, when TLS/SSL is used, automatic port number selection is disabled depending on the MODE parameter and the specified port number is used.

USER=<user-id>
User ID of the user whose mailbox is to be accessed. The ID is enclosed in single quotes. Depending on the server, a distinction may be drawn between uppercase and lowercase.

PASSWORD=<password>
Password for accessing the mailbox. The password is enclosed in single quotes. Depending on the server, a distinction may be drawn between uppercase and lowercase.

KEEP=NO / YES
(For IMAP servers only)
Specifies whether the e-mail is to be deleted on the server after it is downloaded by the mail reader.

KEEP=NO
(For IMAP servers only)
The e-mail is deleted on the server after it is downloaded.

KEEP=YES
(For IMAP servers only)
After being downloaded, the e-mail is flagged as having been read but remains on the server. It can then be flagged as unread again using a standard e-mail client, for example, and thus made available or processing again. This parameter value is particularly useful for test purposes. In live operation, suitable measures must be taken to ensure that the e-mails are deleted in good time, since otherwise there will be a resource bottleneck on the IMAP server.

MAILBOX=<mailbox>
(For IMAP servers only)
IMAP mailbox to be accessed
Default: INBOX

INTERVAL=<int>
Interval in seconds after which the mail reader checks whether there are any new messages. Default: 900 seconds. If zero is specified, the check is carried out only once.

TLS=NO / YES
Specifies whether the connection to the server is to be secured by means of TLS/SSL.

TLS=NO
The connection to the server is not to be secured.

TLS=YES
The connection to the server is to be secured.

REQUIRED=NO / YES
Specifies whether the connection to the server is to be terminated if the server does not support TLS/SSL.

REQUIRED=NO
Even if the server does not support TLS/SSL, the e-mail is transferred. If it is discovered that the host name does not correspond to the server name in the X.509 certificate, an e-mail transfer is still carried out.

REQUIRED=YES
If the server does not support TLS/SSL, the connection is terminated. If it is discovered that the host name does not correspond to the server name in the X.509 certificate, the mail reader is terminated with an error message.

MODE=IMPLICIT / EXPLICIT
Specifies whether the TLS/SSL connection to the server is to be set up implicitly after the TCP connection is set up.

MODE=IMPLICIT
A TLS/SSL connection is set up implicitly immediately after TCP connection setup. If the PORT parameter was not used, the port number 995 (POP3 server) or 993 (IMAP server) is used to set up the TCP connection.

MODE=EXPLICIT
If the PORT parameter was not used, a TCP connection is set up using the default port number 110 (POP3 server) or 143 (IMAP server). A TLS/SSL connection is then set up by means of the POP3 command STLS or the IMAP command STARTTLS. Note that some servers (e.g. the UW POP3/IMAP server) only permit a TLSv1 connection to be set up, not an SSLv3 connection.

PROTOCOL=<protocol spec>
You can limit the protocols used. SSL Version 3 and TLS Version 1, 1.1 and 1.2 are supported.
You can specify SSLv3, TLSv1, TLSv1.1, TLSv1.2 and All.
Depending on whether the protocol is to be activated or deactivated, you can precede it with a plus or minus sign.
The entries “TLSv1 TLSv1.1 TLSv1.21” and “All -SSLv3” have the same effect as long as no support of the future TLS version 1.3 is added to the Mail reader.
Default: All
Some servers support only TLS when MODE=EXPLICIT is used. In this case, PROTOCOL=All -SSLv3 must be used, since otherwise the setup of the TLS/SSL connection fails.

CIPHER_SUITE=<cipher suite spec>
The cipher suite specification is converted into a preference list for encryption algorithms. The specification consists of one or more cipher mnemonics separated by a colon (:).

The permissible cipher mnemonics and cipher suites are described in section “Specification of a cipher suite preference list” on page 60.

CERTIFICATE=<certificate file>
Specifies a file containing the X.509 client certificate for client authentication in PEM format. This file can also contain the client’s private key. Generally, however, the certificate and key are stored in separate files.

KEY=<private key file>
Specifies a file containing the private client key in PEM format. If both the X.509 certificate and private client are contained in the same file (see the CERTIFICATE), this operand does not need to be specified.

The client key should not be protected with a passphrase. If it were, the passphrase would have to be entered each time the mail reader was started. However, file attributes must be set to ensure that unauthorized access to the private key is not possible.

CA_CERTIFICATES=<CA certificates file>
Specifies a file containing in PEM format the CA certificates required for the authentication of the server. The individual PEM certificates are arranged in the file sequentially.

To add or delete certificates, you can edit the file in any text editor. The certificates are entered in the file as follows:

```
-----BEGIN CERTIFICATE-----
< CA certificate in Base64 code >
-----END CERTIFICATE-----
```

Text outside these sequences is ignored by the mail reader and can therefore be used to identify the certificates, which exist in readable form thanks to their ASN.1/Base64 coding.

CRL=<CRL file>
Specifies a file containing the CRLs (certificate revocation lists) of the certificate authorities (CAs). (Certificates issued by a certificate authority can be declared invalid by publishing them in a CRL.)

VERIFY_SERVER=YES / NO
Specifies whether the server's certificate is to be verified.

VERIFY_SERVER = YES
The server's certificate is to be verified.

VERIFY_SERVER = NO
The server's certificate is not to be verified. This setting is vulnerable to "man in the middle" attacks.

VERIFY_DEPTH=<depth>
Specifies the verification depth, which is the maximum permissible number of certificates between the certificate of the server and the certificate known to the mail reader. If the maximum verification depth is exceeded, the connection is terminated unless verification of the server certificate has been disabled by means of VERIFY-SERVER=NO.

1 The server's certificate is not accepted unless it has been signed directly by one of the certificate authorities (CA) known to the mail reader.

0 This makes no sense because in these cases only self-signed certificates are permissible.

Default: 9

Example

In this example, the mail reader polls the server *localhost* every 600 seconds under the user ID *inetvalu* with the password *secret*.

```
SERVER=POP3(
      HOSTNAME=localhost,
      USER='inetvalu',
      PASS='secret',
      INTERVAL=600)
)
```

8.3 Substituting mail-specific parameters

Mail-specific parameters can be substituted in the procedure files and message files using the substitution rules.

In particular, the rules specify the following substitutions:

- The names of the files generated
- E-mail constituent parts (headers, in particular). For example, *%SUBJECT%* is replaced with the e-mail's subject line.

No distinction is drawn between uppercase and lowercase for the keywords. Another rule is that *%%* is replaced with *%* and *\'* and *' '* is replaced with *'*. The keywords that can be used are explained in following sections.

8.3.1 Keywords for the substitution of file names

File names are substituted in accordance with the following keywords:

FILE-NAME
: Name of the message file created

PROCEDURE-NAME
: Name of the procedure file for the processing of the message file

ATTACHMENT-FILE-NAME
: Name of the attachment file created

ATTACHMENT-PROCEDURE-NAME
: Name of the procedure file created for the processing of the attachment file

The file names are formed from several parts grouped together with dots to form a file name:

- The prefix defined in the configuration file or the default
- Date and time in the form YYYY-MMDD.HHMMSS
- A string formed from the *name* operand of the content type header, which is made up as follows: at most, the number of characters needed until the whole file name reaches a length of 38 characters are taken from the beginning. Only alphanumeric characters (letters and digits) are included (i.e. `file1.txt` becomes `file1txt`).
- The suffix defined in the configuration file or by default (for the procedure files)

8.3.2 Keywords for substituting e-mail constituent parts

E-mail constituent parts are substituted in accordance with the following keywords.

The standard e-mail headers listed below are substituted when they are enclosed in percentage signs (%...%).

You will find the associated description in the following RFCs:
– RFC 822 or RFC 2822 (standard mail)
– RFC 1036 (USENET messages)
– RFC 2045 (MIME messages)

Bcc	Sender
Cc	Subject
Comments	To
Date	Approved
Encrypted	Control
From	Distribution
In-Reply-To	Expires
Keywords	Followup-To
Message-Id	Lines
Received	Newsgroups
References	Organization
Reply-To	Path
Resent-Bcc	Summary
Resent-Cc	Xref
Resent-Date	Content-Description
Resent-From	Content-Id
Resent-Message-Id	Content-Transfer-Encoding
Resent-Reply-To	Cte (synonym: Content-Transfer-Encoding)
Resent-Sender	Content-Type
Resent-To	Mime-Version
Return-Path	Content-Disposition

The e-mail headers are only substituted when the header fields are contained in the e-mail.

The keyword “Text” is substituted with the message body text of the e-mail. The keyword “Header” is substituted with the header lines.

Note also that:

- All line breaks are removed from the header.
- No distinction is drawn between uppercase and lowercase.
- Two percentage signs (%%) are substituted with a single percentage sign (%).
- A backslash followed by a single quote (\') and a double quote (") are substituted with a single quote (').

Parameters of selected header fields

To facilitate access to selected e-mail headers, the following additional substitutions are made:

Content-Type-Type
: The *Type* field of the Content-Type header

  Example:
  In the case of *Content-Type: text/plain;name="filename.txt"*, it is *text*.

Content-Type-Subtype
: The *Subtype* field of the Content-Type header

  Example:
  In the case of *Content-Type: text/plain;name="filename.txt"*, it is *plain*.

Content-Type-Name
: The *name* field of the Content-Type header

  Example:
  In the case of *Content-Type: text/plain;name="filename.txt"*, it is *"filename.txt"*.

Content-Disposition-Name
: The *Filename* field of the Content-Disposition header

  Example:
  In the case of *Content-Disposition: inline; filename="filename.txt"*, it is *"filename.txt"*.

Disposition-Type-Name
: The *Filename* field of the Content-Disposition header

  Example:
  In the case of *Content-Disposition: inline; filename="filename.txt"*, it is *"filename.txt"*.

8.3.3 S/MIME

With S/MIME-signed and/or encrypted e-mails, to enable the success status of the decryption or verification of the signature to be returned, the following keywords are supported:

SMime-Decryption-Status
: “NOT-ENCRYPTED” if the e-mail was not encrypted,
“OK” if it was successfully decrypted,
otherwise a short description of the error

SMime-Verification-Status
: “NOT-SIGNED if the e-mail was not signed,
“OK” if the e-mail signature was successfully verified,
otherwise a short description of the error

SMime-Signer-DN
: DN (Distinguished Name) of the e-mail signatory.

SMime-Signer-CN
: CN (Common Name) of the e-mail signatory

SMime-Signer-E-Mail
: E-mail address(es) of the e-mail signatory, comma-delimited; there may be several e-mail addresses in the case of X.509v3 certificates when an e-mail address is specified for both DN and SubjectAltName.

SMime-Issuer-DN
: DN (Distinguished Name) of the certificate authority (CA) that issued the e-mail signatory certificate

In the case of multipart/mixed messages, the information about decryption and/or signature verification is only assigned to the message body text part, not the attachment parts. In the case of the latter, *%SMime-Decryption-Status%* is expanded to *NOT-ENCRYPTED* and *%SMime-Verication-Status%* to *NOT-SIGNED* unless the attachment is an S/MIME-signed or encrypted e-mail.

8.4 Processing an e-mail procedurally

Regardless of the structure of the e-mail, all of its parts (header, attachments) are made available in BS2000 procedures.

8.4.1 Structure of an e-mail

E-mails can vary in structural complexity:

- At their simplest, they consist of headers and a message.
- However, the e-mail can also contain attachments. The e-mail then consists of the general headers, such as “Subject” and “From”, the message (with its own headers) and the attachments (with their own headers).
- In the most complex case, the e-mail has an attachment containing a further e-mail, which can in turn contain attachments and/or other e-mails as attachments, and so on.

The mail reader analyzes the e-mail, subdivides it for procedural processing into smaller packages and makes these individually processable packages available to the procedures. Within the procedures this permits access to the headers of the e-mail and the individual attachments, regardless of the complexity of the e-mail’s structure.

The mail reader converts all the data received from the IMAP or POP3 server from ISO-8859-1 to EDF041. If a MIME part of the mail is used as the 'base64' Content-Transfer-Encoding and the content type is 'text', the data is in turn converted from ISO-8859-1 to EDF041 in accordance with the base64 decoder. In all other cases, the processing procedures must execute any necessary character set conversion.

8.4.2 Processing e-mails with BS2000 procedures

8.4.2.1 E-mail without attachment

In the case of an e-mail without an attachment, the mail reader creates the following two files:

- A file containing the e-mail’s message
- A file containing the procedure for the processing of the e-mail. The structure of the procedures created is largely freely configurable.

The mail reader generates a new name for each e-mail received.

The procedure is then called by means of an `/ENTER-JOB` or `/ENTER-PROCEDURE` command.

Example 1

This example shows how to configure the mail reader in such a way that the procedure created only has one line. All e-mails received are printed.

```
/PRINT-FILE MAIL.2010-06-27.073001,DELETE-FILE=*YES
```

MAIL.2010-06-27.073001 is the unique name under which the message text is stored.

Accordingly, the name of the procedure is *MAIL.2010-06-27.073001.PROC*.

The mail reader then calls the *MAIL.2010-06-27.073001.PROC* procedure, which in turn starts printout with */PRINT-FILE*.

The corresponding section of the configuration file must look like this:

```
MAILHANDLING = PARAMETERS(
    ENTER = PROCEDURE,
    BODY= PARAMETERS (
        PROCEDURE = PARAMETERS (
           TEXT = ('/PRINT-FILE %FILE-NAME%,DELETE-FILE=*YES'
           )
        )
    )
)
```

The mail reader substitutes the keyword expression *%FILE-NAME%* with the current name of the message file.

Example 2

Most headers can also be substituted with such variables. In the following section from the configuration file, %SUBJECT% is substituted with the subject of the e-mail:

```
TEXT = ('/PRINT-FILE %FILE-NAME%,DELETE-FILE=*YES,',
        '/  COVER-PAGES=PAR(HEADER-PAGE-TEXT=''Subject: %SUBJECT%'')'
)
```

Two single quotes (*' '*) are substituted with one (*'*).

This results in the following procedure, for example:

```
/PRINT-FILE MAIL.2010-06-27.073001,DELETE-FILE=*YES,-
/  COVER-PAGES=PAR(HEADER-PAGE-TEXT='Subject:Mail-Reader 2010')
```

Example 3

The *HANDLEMAIL* procedure is called for each e-mail received. The subject and file name of the e-mail are passed as parameters.

The corresponding section of the configuration file looks like this, for example:

```
MAILHANDLING = PARAMETERS(
    ENTER = PROCEDURE,
    BODY = PARAMETERS (
        PROCEDURE = PARAMETERS (
           TEXT = ('/CALL-PROCEDURE HANDLEMAIL,(FILENAME=%FILE-NAME%,-',
                    '/   Subject=''%SUBJECT%'')'
           )
        )
    )
)
```

8.4.2.2 E-mail with attachments

An e-mail contains several attachments. The e-mail thus consists of the following parts:

– Header
– Message with headers
– Attachment 1 with headers
– Attachment 2 with headers
– etc.

Accordingly, the mail reader creates the following files:

- Two files for the message:
  – A file for the procedure for processing the message
  – A file for processing the message body
- Two files for each attachment:
  – One file contains the attachment.
  – The other file may contain a procedure for processing the attachment.

  The names of the files created for the attachments are made up of the prefix, which is also used for the inline message, and a file name derived from the Content-Type or Content-Disposition headers.

Example 1: Processing the entire e-mail

The sample e-mail has two attachments: *file1.txt* and *file2.txt*. If you want to print the e-mail in its entirety, including the attachments *file1.txt* and *file2.txt*, the corresponding section of the configuration file looks like this:

```
MAILHANDLING = PARAMETERS(
    ENTER = PROCEDURE,
    BODY = PARAMETERS(
      PROCEDURE = PARAMETERS (
        TEXT = ('/PRINT-FILE %FILE-NAME%'),
        ATTACHMENT = ('/PRINT-FILE %ATTACHMENT-FILE-NAME%,-',
                      '/ COVER-PAGES=PAR(HEADER-PAGE-TEXT=''%CONTENT-TYPE%'')'
          )
       )
    )
)
```

This results in the following *MAIL.2010-06-27.073001.PROC* procedure:

```
/PRINT-FILE MAIL.2010-06-27.073001
/PRINT-FILE MAIL.2010-06-27.073002.file1txt,-
/ COVER-PAGES=PAR(HEADER-PAGE-TEXT='text/plain; name:“file1.txt“')
/PRINT-FILE MAIL.2010-06-27.073002.file2txt,-
/ COVER-PAGES=PAR(HEADER-PAGE-TEXT='text/plain; name:“file2.txt“')
```

The expression *%CONTENT-TYPE%* in the configuration file is substituted with the e-mail header received.

Example 2: Processing attachments

The sample e-mail has two attachments: *file1.txt* and *file2.txt*. For the attachments *file1.txt* and *file2.txt* you can create your own procedures (e.g. with the following configuration file):

```
MAILHANDLING = PARAMETERS(
    ENTER = PROCEDURE,
    BODY = PARAMETERS(
        PROCEDURE = PARAMETERS (
            TEXT = ('/PRINT-FILE %FILE-NAME%'
            ),
            ATTACHMENT = ('/CALL-PROC %ATTACHMENT-PROCEDURE-NAME%,-',
                          '/  (FILENAME=%ATTACHMENT-FILE-NAME%)'
            )
        )
    ),
    ATTACHMENT = PARAMETERS (
        PROCEDURE = ('/BEG-PAR-DECL',
                     '/DECL-PAR FILENAME',
                     '/END-PAR-DECL',
                     '/CALL-PROCEDURE ATTPROC,(FILENAME=&(FILENAME))'
        )
    )
)
```

This results in the following *MAIL.2010-06-27.073001.PROC* procedure:

```
/PRINT-FILE MAIL.2010-06-27.073001
/CALL-PROC MAIL.2010-06-27.073002.file1txt.PROC,-
/  (FILENAME=MAIL.2010-06-27.073002.file1txt)
/CALL-PROC MAIL.2010-06-27.073002.file2txt.PROC,-
/  (FILENAME=MAIL.2010-06-27.073002.file2txt)
```

In addition, the procedures *MAIL.2010-06-27.073002.file1txt.PROC* and *MAIL.2010-06-27.073002.file2txt.PROC* are created.

```
/BEG-PAR-DECL
/DECL-PAR FILENAME
/END-PAR-DECL
/CALL-PROCEDURE ATTPROC,(FILENAME=&(FILENAME))
```

8.5 Programming interface (C++)

You will find a PDF file containing a full description of the programming interface in the LMS library *SYSLIB.MAIL.nnn* (X element with the name *MIMELIB.PDF*).

To use the C++ interface, the user must make available a function of the following prototype:

```
int handle_mail(DwMessage *);
```

This function is called by the mail reader on receipt of an e-mail, and the e-mail is passed as `DwMessage`. In the function the user can process the e-mail. If a value of 0 is returned, the mail reader deletes the e-mail from the mailbox; otherwise, the e-mail remains in the mailbox and another attempt is made to process this e-mail again later.

9 Mail sender in BS2000

The mail sender is a user agent (Mail User Agent, MUA) with which you can send data in BS2000 as e-mails.

The BS2000 mail sender allows you to do the following:

- Send automated lists as e-mails to the local mail server in POSIX or to remote mail servers from within BS2000 procedures.
- Send notifications in error situations.
- Send text or binary files from within BS2000.
- Specify transfer encoding (7- and 8-bit encoding, quoted-printable encoding and Base64 encoding).
- Specify To, Cc and Bcc recipients.
- Specify additional conversion tables for national character sets.
- Send e-mails to the SMTP server over an SMTP connection secured with TLS.
- Encrypt and sign e-mails correctly in accordance with the S/MIME standard.
- Send e-mails asynchronously.

The last function in the above list is based on the BS2000 subsystem ASTI (**A**ssistant for **S**ervice **T**ask **I**ntegration). The frontend of the mail sender, which is implemented as the DSSM subsystem MAILCLNT, receives the e-mails at the SDF command, SVC or TPR interface and creates e-mail orders that it transfers to ASTI. The mail sender backend is a TU service task that receives the e-mails from ASTI and forwards them to a mail server over an SMTP connection.

The figure below shows the position of the mail sender and how it interacts with the other mail services in BS2000.

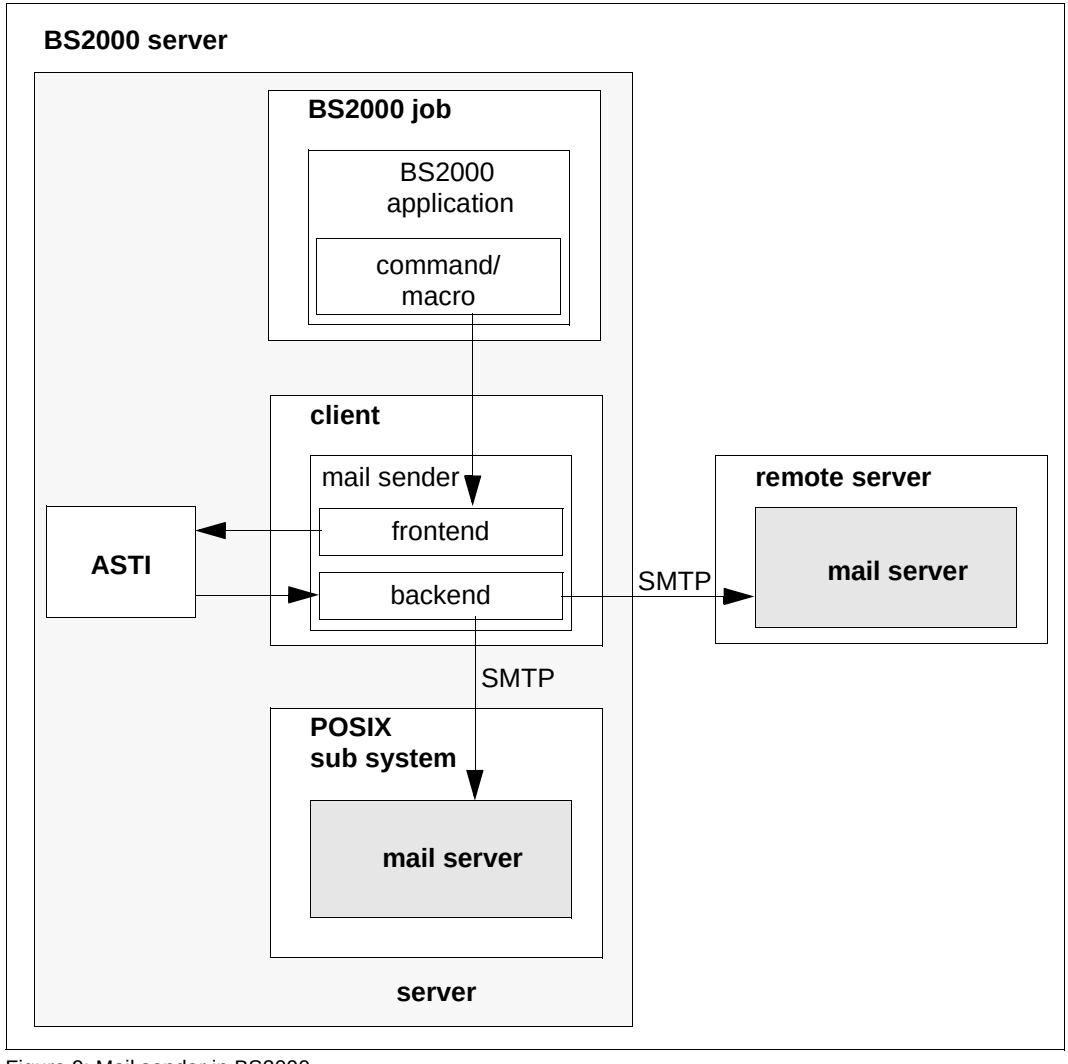


Figure 9: Mail sender in BS2000

This chapter provides information on the following:

- The configuration file for the mail sender frontend (user option file)
- The SDF interface of the mail sender frontend
- The macro interface of the mail sender frontend

9.1 Configuration file for the mail sender frontend (user option file)

To facilitate the definition of parameters that seldom change, the frontend interface of the mail sender supports a user option file (by default *SYSDAT.MAIL.nnn.USER.OPT*).

The options must comply with the following rules:

- Each option begins in a new line.
- If the arguments of an option are longer than one line, lines to be continued must end with a backslash (\).
- Lines that begin with the number sign (#) are ignored.
- The option names are not dependent on uppercase/lowercase.
- Option values are not dependent on uppercase/lowercase if that is not explicitly mentioned.

The options are described below.

fromAddress

Specifies the e-mail address to be used as the sender address, provided this has not already been defined by some other means. If the sender address is not defined either in the option file or in the corresponding operands of the SDF or macro interface, the mail sender does not accept the e-mail.

If a mail server cannot forward an e-mail, it sends a bounce e-mail to the sender address. It is therefore important to define a valid sender address.

fromAddress
`<sender-mail-address>`

<sender-mail-address>
E-mail to be used as the sender address

fromDisplayName

With this option a character string is specified, which is prepended to the proper sender address in the mail header, if not otherwise a sender address is specified for the mail header (e.g. via fake_from specification at the subprogram interface).

The specified character string is used without XHCS conversion and without QP/Base64 encoding, i.e. after a standard EBCDIC-ISO8859 conversion it must contain only 7 bit ASCII characters.

When nevertheless non-ASCII characters shall be displayed, the user has to do himself the QP or Base64 encoding according to RFC 2047.

fromDisplayName
<display name 1..63>

<display name 1..63>
character string which is prepended to the sender address

fromDisplayNameWithHostName

This option specifies whether the character string given with th option fromDisplayName shall be complemented with the host name put between square brackets if need be.

When due to the fromDisplayName option the sender address is complemented with a display name, then with the parameter value YES the hostname put between square brackets is appended to this display name.

fromDisplayNameWithHostName
NO | **YES**

NO The hostname put between square brackets is not appended.
NO is the default.

YES The hostname put between square brackets is appended.

sign

Specifies whether the e-mail is signed with S/MIME.

sign
NO | **YES**

NO The e-mail is not signed with S/MIME.
NO is the default.

YES The e-mail is signed with S/MIME.

encrypt

Specifies whether the e-mail is encrypted with S/MIME.

encrypt
NO | **YES**

NO The e-mail is no encrypted with S/MIME.
NO is the default.

YES The e-mail is encrypted with S/MIME.

privateKeyFile

The file specified in this option contains a private key in PEM format that can be used to sign e-mails with S/MIME. If the certificate and the private key are contained in the same file, you do not have to define this option. The key cannot be protected with a passphrase because it is not possible to enter a passphrase.

privateKeyFile
<filename 1..54> | ***NONE**

<filename 1..54>
File containing the private key in PEM format

***NONE**
A separate file is not used for the private key.
*NONE is the default.

signerCertificateFile

The file specified in this option contains an X.509 certificate in PEM format that can be used to sign e-mails with S/MIME. If required, the file can also contain the private key. Usually, however, the certificate and key are stored in different files. In this case, you specify the key in the *privateKeyFile* option. The certificate must correspond to the private key actually used. It is particularly important to note this when the certificate is defined by means of an SDF command and the key by means of a user option or vice versa.

signerCertificateFile
<filename 1..54> | ***NONE**

<filename 1..54>
File containing the X.509 certificate in PEM format

***NONE**
A separate file with X.509 certificates is not used.
*NONE is the default.

addSignerCertificatesFile

The file specified in this option contains additional X.509 certificates in PEM format. These certificates may be required in order to provide an uninterrupted chain of certificates from the signatory certificate to the root CA certificate.

addSignerCertificatesFile
<filename 1..54> | ***NONE**

<filename 1..54>
File containing the additional X.509 certificates in PEM format

***NONE**
A separate file with X.509 certificates is not used.
*NONE is the default.

recipientCertificatesFile

The file specified in this option contains X.509 certificates in PEM format that can be used for the S/MIME encryption of e-mails. The certificates must contain an e-mail address as part of the Subject DN (Distinguished Name) or part of the X.509v3 Subject Alternative Name. During encryption of the e-mail, this certificate is checked to establish whether it contains an e-mail address that corresponds to an e-mail address in the list of e-mail recipients. If it does, the certificate is used for encryption.

If the *recipientCertificatesFile* option is used more than once, a search is carried out for the files specified there in the order in which the corresponding options occur.

recipientCertificatesFile
<filename 1..54> | ***NONE**

<filename 1..54>
File containing the X.509 certificates in PEM format

***NONE**
A separate file containing X.509 certificates is not used.
*NONE is the default.

certificateRevocationListFile

The file defined by this option contains certificate revocation lists (CRLs) for the X.509 certificates specified with *recipientCertificatesFiles*.

certificateRevocationListFile
<filename 1..54> | ***NONE**

<filename 1..54>
File containing the certificate revocation lists for X.509 certificates

***NONE**
A file for certificate revocation lists is not used.
*NONE is the default.

CACertificatesFile

The *CACertificatesFile* option specifies a file containing the CA certificates in PEM format that are required to check the e-mail recipient certificates (with the help of certificate revocation lists). The individual PEM certificates are arranged in the file sequentially.

To add or delete certificates, you can edit the file in any text editor. The individual certificates are entered in the file as follows:

```
-----BEGIN CERTIFICATE-----
< CA certificate in Base64 encoding>
-----END CERTIFICATE-----
```

Text outside these sequences is ignored by the mail client and can therefore be used to describe the certificates, which are in non-readable form due to their ASN.1/Base64 encoding.

CACertificatesFile
<filename 1..54> | ***NONE**

<filename 1..54>
: Name of the file containing the certificates in PEM format that are required to check the e-mail recipient certificates.

***NONE**
: A file containing CA certificates is not specified.
*NONE is the default.

cipher

Specifies which symmetric encryption algorithm shall be used, when the mail is encrypted.

cipher
3DES | **DES** | **RC2-40** | **RC2-64** | **RC2-128** | **AES-128** | **AES-192** | **AES-256**

3DES The default is 3DES.

logFile

Names the file for the user specific logging.

When the file is temporarily not writable, because e.g. a SHOW-FILE command is issued onto it from another task, then the logging entries generated during this period are lost. The default for the file name is SYSDAT.MAIL.*nnn*.USER.MAILLOG.

logFile
`<file name 1..54>`

<file name 1..54>
File for the user specific logging

logItems

Controls what is written to the user specific logging file.

This option can be specified multiple times for defining a list of data to be logged.

With "Order" after the (attempted) handover of a mail send order to ASTI an entry is written to the logging file. The entry contains a time stamp, the ASTI order id (in case of a successful handover), the recipient address(es) and the subject header of the mail. In case of a failed handover an appropriate error message is provided.

With "Result", after the mail order status is requested successfully, a timestamp, the ASTI order id and the result information is written. The latter indicates whether the mail has been transferred successfully to a mail server by the backend task. If yes, then the message from the mail server typically contains a part of the message id, which possibly is useful for further mail tracking. If the transfer to a mail server fails, then this circumstance is indicated in the result with an explaining text. When the request of the mail order status failed, this is documented with an appropriate message in the logging file.

logItems
None | **Order** | **Result**

None Deletes the list of data to be logged.
The default is None, i.e. one starts with an empty list.

Order The corresponding logging entries are incorporated to the list.

Result The corresponding logging entries are incorporated to the list.

9.2 SDF command interface of the mail sender frontend

You can use the following SDF commands to send mail send orders to the ASTI subsystem and manage these orders:

- SEND-MAIL
- REQUEST-MAIL-ORDER-RESULT
- DELETE-MAIL-ORDER
- SHOW-MAIL-ORDER-STATUS

SEND-MAIL - Send mail

Domain:
UTILITIES

Required authorization:
STD-PROCESSING

The SEND-MAIL command allows you to create an order to send e-mails and send the order to the ASTI subsystem. ASTI forwards the mail send order to the server task of the mail sender backend, which carries out the order and returns a status code. ASTI restricts the size of the orders sent to 32 KB minus the bytes reserved for administration purposes.

If ASTI rejects a SEND-MAIL call because the maximum length has been exceeded, proceed as follows:

- Try to subdivide the e-mail into several smaller e-mails with smaller recipient lists and/or fewer attachments.
- Save the e-mail texts and attachments in files instead of specifying the contents directly in the command. Specify only the associated file names in the command.

SEND-MAIL

FROM = *USER-OPTION / <c-string 1..255 with-lower-case>

,TO = *NONE / <c-string 1..1800 with-lower-case>

,CC = *NONE / <c-string 1..1800 with-lower-case>

,BCC = *NONE / <c-string 1..1800 with-lower-case>

,REPLY-TO = *NONE / <c-string 1..255 with-lower-case>

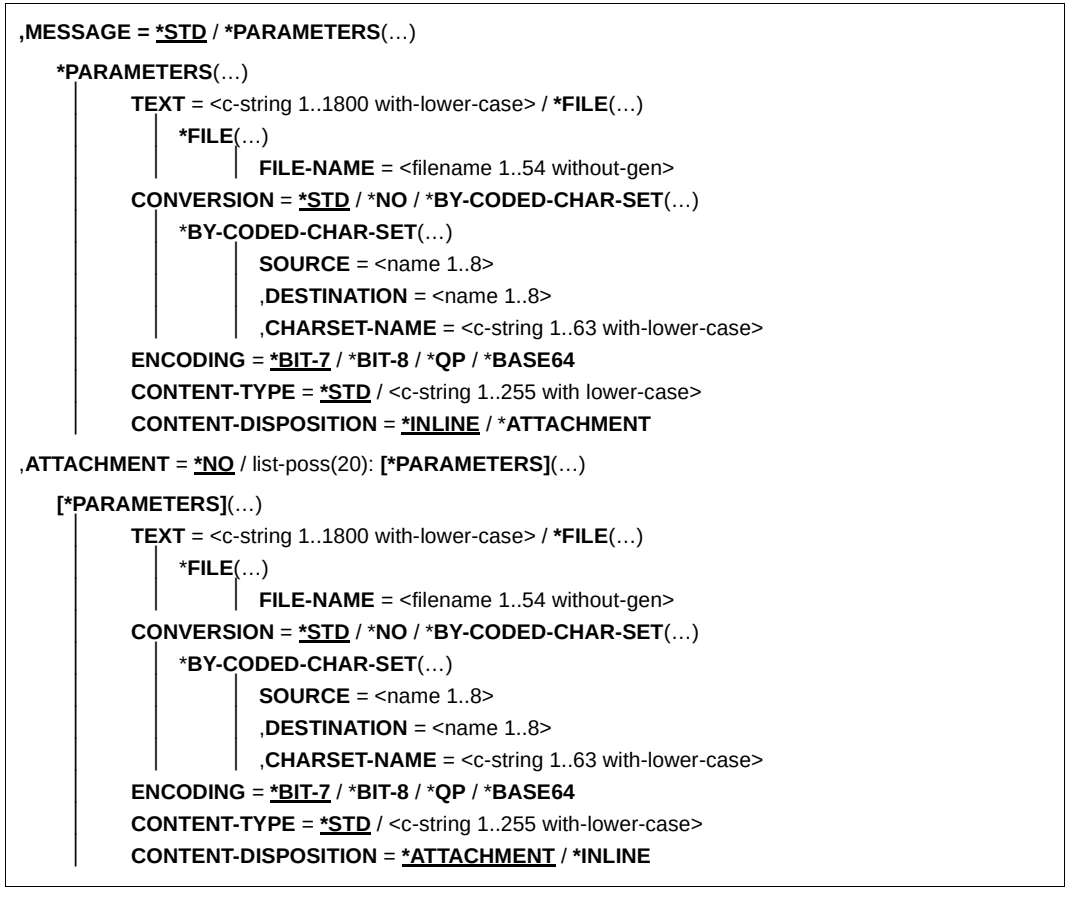
,SUBJECT = *NONE / <c-string 1..1800 with-lower-case>

,ADDITIONAL-HEADER = *NONE / list-poss(10): **[*HEADER]**(…)

[*HEADER](…)
NAME = <c-string 1..63 with-lower-case>
,**BODY** = <c-string 1..255 with-lower-case>

,HEADER-CONVERSION = *STD / ***BY-CODED-CHAR-SET**(…)

***BY-CODED-CHAR-SET**(…)
SOURCE = <name 1..8>
,**DESTINATION** = <name 1..8>
,**CHARSET-NAME** = <c-string 1..63 with-lower-case>


,MESSAGE = *STD / *PARAMETERS(...)
*PARAMETERS(...)
TEXT = <c-string 1..1800 with-lower-case> / *FILE(...)
*FILE(...)
FILE-NAME = <filename 1..54 without-gen>
CONVERSION = *STD / *NO / *BY-CODED-CHAR-SET(...)
*BY-CODED-CHAR-SET(...)
SOURCE = <name 1..8>
,DESTINATION = <name 1..8>
,CHARSET-NAME = <c-string 1..63 with-lower-case>
ENCODING = *BIT-7 / *BIT-8 / *QP / *BASE64
CONTENT-TYPE = *STD / <c-string 1..255 with lower-case>
CONTENT-DISPOSITION = *INLINE / *ATTACHMENT
,ATTACHMENT = *NO / list-poss(20): [*PARAMETERS](...)
[*PARAMETERS](...)
TEXT = <c-string 1..1800 with-lower-case> / *FILE(...)
*FILE(...)
FILE-NAME = <filename 1..54 without-gen>
CONVERSION = *STD / *NO / *BY-CODED-CHAR-SET(...)
*BY-CODED-CHAR-SET(...)
SOURCE = <name 1..8>
,DESTINATION = <name 1..8>
,CHARSET-NAME = <c-string 1..63 with-lower-case>
ENCODING = *BIT-7 / *BIT-8 / *QP / *BASE64
CONTENT-TYPE = *STD / <c-string 1..255 with-lower-case>
CONTENT-DISPOSITION = *ATTACHMENT / *INLINE

```
,SECURITY = *NO / *SMIME(...)
    *SMIME(...)
        SIGNING = *USER-OPTION / *NO / *YES(...)
            *YES(...)
                CERTIFICATE-FILE = *USER-OPTION / <filename 1..54 without-gen>
                ,KEY-FILE = *USER-OPTION / <filename 1..54 without-gen>
                ,ADDITIONAL-CERT-FILE = *USER-OPTION / *NONE / <filename 1..54 without-gen>
        ENCRYPTING = *USER-OPTION / *NO / *YES(...)
            *YES(...)
                CERTIFICATE-FILE = *USER-OPTION / <filename 1..54 without-gen>
                ,CIPHER = *USER-OPTION / *DES3 / *DES / *RC2-40 / *RC2-64 / *RC2-128 /
                          *AES-128 / *AES-192 / *AES-256
                ,CRL-FILE = *USER-OPTION / *NONE / <filename 1..54 without-gen>
                ,CA-CERTIFICATES-FILE = *USER-OPTION / *NONE / <filename 1..54 without-gen>
,USER-OPTION-FILE = *STD / <filename 1..54 without-gen>
,WAIT-FOR-RESULT = *NO(...) / *YES
        *NO(...)
            RESULT = *DISCARD / *BY-REQUEST-CMD
```

Operands

FROM = *USER-OPTION / <c-string 1..255 with-lower-case>
E-mail address of the e-mail sender. Information about problems sending the e-mail (“bounce e-mails”) are sent to this address. The value of the operand is also entered in the e-mail’s *FROM* header field. Specify the address in the following format:
“<local part>@<domain>“.

FROM = *USER-OPTION
The address of the e-mail sender is taken from the user option file. If the address is not defined there and the SYSSSI option senderSuffix is not used, the command is rejected with an error code.

FROM = <c-string 1..255 with-lower-case>
You specify the address of the sender in the format “<local part>@<domain>”.

TO = *NONE / <c-string 1..1800 with-lower-case>
Specifies the e-mail recipient address(es).

TO = *NONE
You have to define the e-mail recipients by means of other operands. The *TO* header field is generated.

TO = <c-string 1..1800 with-lower-case>
E-mail address of the recipient or a list of the e-mail addresses of the recipients. The addresses must be separated by commas in the list. The value of the operand is also entered in the e-mail's *TO* header field. You specify the addresses in the following format: "<local part>@<domain>".

CC = *NONE / <c-string 1..1800 with-lower-case>
Specifies the CC (carbon copy) e-mail recipient address(es).

CC = *NONE
You have to define the e-mail recipients by means of other operands. The *CC* header field is not generated.

CC = <c-string 1..1800 with-lower-case>
CC e-mail address of the recipient or list of CC e-mail addresses of the recipients. The CC addresses must be separated by commas in the list. The value of the operand is also entered in the e-mail's *CC* header field. You specify the addresses in the following format: "<local part>@<domain>".

BCC = *NONE / <c-string 1..1800 with-lower-case>
Specifies the BCC (blind carbon copy) mail recipient address(es).

BCC = *NONE
You have to define the e-mail recipients by means of other operands.

BCC = <c-string 1..1800 with-lower-case>
BCC e-mail address of the recipient or list of the BCC e-mail addresses of the recipients. The BCC addresses must be separated by commas in the list. The value of the operand is not entered in any of the mail's header fields. Consequently, the recipients cannot see the other addresses to which the e-mail has been sent. You specify the addresses in the following format: "<local part>@<domain>".

REPLY-TO = *NONE / <c-string 1..255 with-lower-case>
Specifies the e-mail address of the e-mail sender to which the recipients are to send replies.

REPLY-TO = *NONE
A *REPLY-TO* header field is not added. In order to reply, the recipient has to take the address from a different header field (e.g. the *FROM* header field).

REPLY-TO = <c-string 1..255 with-lower-case>
E-mail address of the e-mail sender to which the recipients are to send replies. The value of the operand is entered in the *REPLY-TO* header field. You specify the address in the following format: "<local part>@<domain>".

SUBJECT = *NONE / <c-string 1..255 with-lower-case>
Subject of the e-mail

SUBJECT = *NONE
A *SUBJECT* header field is not added to the e-mail. It is advisable to specify a subject to enable recipients to deal with their e-mails more easily.

SUBJECT = <c-string 1..255 with-lower-case>
Subject of the e-mail. The value of the operand is entered in the e-mail's *SUBJECT* header field.

ADDITIONAL-HEADER = *NONE / list-poss(10): *HEADER(…)
Definition of additional header fields for the e-mail

ADDITIONAL-HEADER = *NONE
No additional header fields

ADDITIONAL-HEADER = list-poss(10): *HEADER(…)
You can define up to 10 additional header fields in a list. The name and body are specified in the corresponding operands.

NAME = <c-string 1..63 with-lower-case>
Name of the header field

BODY = <c-string 1..255 with-lower-case>
Body of the header field

HEADER-CONVERSION = *STD / *BY-CODED-CHAR-SET(…)
Specifies the type of character set conversion to be used for the header fields.

HEADER-CONVERSION = *STD
Character set conversion from XHCS CCSN EDF03IRV to CCSN ISO646 is used. It is assumed that the header fields contain ASCII characters exclusively, so that RFC 2047 encoding is not necessary.

The user can also use character sets that are not supported by XHCS. The prerequisite is that the user must provide headers that comply with RFC 2047 encoding following the character set conversion described above.

HEADER-CONVERSION = *BY-CODED-CHAR-SET(…)
The header fields are handled as data to which character set conversion from an EBCDIC variant to an ISO-8859 variant has to be applied. The conversion is carried out with the help of XHCS. You will find more information on XHCS in the manual "XHCS (BS2000/OSD)".

SOURCE = <name 1..8>
Source XHCS CCSN of the user data

DESTINATION = <name 1..8>
Destination XHCS CCSN of the user data

CHARSET-NAME = <c-string 1..63 with-lower-case>
Name of the character set used in RFC 2047 encoding

MESSAGE = *STD / *PARAMETERS(…)
Defines the message to be sent.

MESSAGE = *STD
A blank message is sent. This can contain attachments defined in the ATTACHMENT operand.

MESSAGE = *PARAMETERS(…)
A message containing the parameters explained below is created:

TEXT = <c-string 1..1800 with-lower-case> / *FILE(…)
Specifies the text of the message to be sent to the recipient(s).

TEXT = <c-string 1..1800 with-lower-case>
Text of the message. You force an explicit line break by inserting the string “\n” at the appropriate point. If this string is to be part of the text, it must be entered as “\\n”. No other conversions are carried out.

TEXT = *FILE(…)
The text of the message must be read from a file.

FILE-NAME = <filename 1..54 without-gen>
Name of the file containing the message. The file must remain unchanged until mail sending is concluded by the backend.

CONVERSION = *STD / *NO / *BY-CODED-CHAR-SET(…)
Specifies whether the user data consists of text or binary data. If the user data consists of text, it is also specified which character set is used and which type of EBCDIC-*x* to ISO-*y* conversion is to be carried out.

CONVERSION = *STD
Character set conversion from XHCS CCSN EDF03IRV to CCSN ISO646 is carried out. “text/plain; charset=us-ascii” is specified for the *Content-Type* header field. The value “7bit” is set for the *Content-Transfer-Encoding* header field.

CONVERSION = *NO
The user data is treated as binary data. In other words, there is no character set conversion, for example. If the ENCODING operand has the (default) value *BIT-7, this value is replaced with *BASE64.

CONVERSION = *BY-CODED-CHAR-SET(…)
The user data is treated as text, for which character set conversion from an EBCDIC variant to an ISO-8859 variant usually has to be carried out. The conversion is carried out with the help of XHCS. You will find more detailed information on XHCS in the manual “XHCS (BS2000/OSD)”. When XHCS in version 2.0 or higher is available, then also the Unicode related CCSNs UTF8, UTF16, UNICODE and UTFE can be used.

SOURCE = <name 1..8>
Source XHCS CCSN of the user data

DESTINATION = <name 1..8>
Destination XHCS CCSN of the user data

CHARSET-NAME = <c-string 1..63 with-lower-case>
If the contents of the message consist of text (see the CONTENT-TYPE operand), the value of the CHARSET-NAME operand is inserted in the *Content-Type* header field of the message as a character set parameter. Otherwise, the operand is ignored.

ENCODING = *BIT-7 / *BIT-8 / *QP / *BASE64
Specifies how the user data is encoded for the transfer.

ENCODING = *BIT-7
After character set conversion, it should be possible to represent all characters as 7-bit ASCII characters. If this is not the case, some characters may be altered during mail transport. The value “7bit” is entered in the *Content-Transfer-Encoding* header field. If your mail contains lines with more than 998 characters (without the trailing CR LF), use ENCODING=*QP or ENCODING=*BASE64, because otherwise your mail will be most probably truncated or otherwise altered during mail transport.

ENCODING = *BIT-8
The value “8bit” is entered in the *Content-Transfer-Encoding* header field.

When you are not sure whether all mail servers in the transport chain are "8-bit-clean", you should use ENCODING=*QP or ENCODING=*BASE64. The same recommendation applies, when your mail contains lines with more than 998 characters (without the trailing CR LF), because otherwise your mail will be most probably truncated or otherwise altered during mail transport.

ENCODING = *QP
After character set conversion, all characters which are not directly represantable as 7 bit characters are encoded in accordance with the quoted-printable algorithm (see RFC 2045). This results in lines of at most 80 7 bit characters. This encoding is helpful, above all, when only a few characters have to be converted. In this case, the great majority of the data remains readable to humans. The value “quoted-printable” is entered in the *Content-Transfer-Encoding* header field.

ENCODING = *BASE64
After character set conversion, the data is encoded in accordance with the Base64 algorithm (see RFC 2045). This results in lines of at most 80 7 bit characters. This encoding is effective, above all, for binary data when only a few characters can directly be represented as 7-bit ASCII characters. In this case, quoted-printable encoding would result in much more data than Base64 encoding and would not be readable to humans. Base64 encoding increases the volume of data by a third. The value “base64” is entered in the *Content-Transfer-Encoding* header field.

CONTENT-TYPE = *STD / <c-string 1..255 with-lower-case>
The value of this operand is entered in the message's *Content-Type* header field. This field displays to the receiving client the medium type of the message transferred (e.g. pure text, text editor file, image file, movie file).

Example

For human-readable text, specify "text/plain". For images in JPEG format, specify "image/jpeg".

CONTENT-TYPE = *STD
The *Content-Type* header field is set to the value "text/plain" and the value of the CHARSET-NAME operand.

CONTENT-TYPE = <c-string 1..255 with-lower-case>
The value of this operand is entered in the *Content-Type* header field. Use a value that corresponds to your data (see RFC 2046, for example).

CONTENT-DISPOSITION = *INLINE / *ATTACHMENT
The value of this operand is entered in the message's *Content-Disposition* header field (see RFC 2183). This indicates to the receiving mail client whether this part of the e-mail is to be displayed automatically or only when requested by the user who reads the e-mail.

CONTENT-DISPOSITION = *INLINE
The message's *Content-Disposition* header field is set to "inline". This indicates to the receiving mail client that this mail body is to be displayed automatically when the message is displayed.

CONTENT-DISPOSITION = *ATTACHMENT
The message's *Content-Disposition* header field is set to "attachment". This indicates to the receiving mail client that this part of the message body is only to be displayed on the request of the user.

ATTACHMENT = *NO / *PARAMETERS(...)
Creates one or more attachments to be added to a message.

ATTACHMENT = *NO
No attachments are created.

ATTACHMENT = *PARAMETERS(...)
An attachment is created with the parameters described below.

TEXT = <c-string 1..1800 with-lower-case> / *FILE(...)
Specifies the user data sent to the recipient(s).

TEXT = <c-string 1..1800 with-lower-case>
Text of the message. You force an explicit line break by inserting the string “\n” at the appropriate point. If this string is to be part of the text, it must be entered as “\\n”. No other conversions are carried out.

TEXT = *FILE(...)
The user data must be read from a file.

FILE-NAME = <filename 1..54 without-gen>
The name of the file that contains the user data. The file must remain unchanged until mail sending is concluded by the backend.

CONVERSION = *STD / *NO / *BY-CODED-CHAR-SET(...)
Specifies whether the user data consists of text or binary data. If the user data consists of text, it is also specified which character set is used and which EBCDIC-*x* to ISO-*y* conversion is to be carried out.

CONVERSION = *STD
A character set conversion is carried out from XHCS CCSN EDF03IRV to CCSN ISO646. The value “text/plain; charset=us-ascii” is entered in the *Content-Type* header field. The value “7bit” is entered in the *Content-Transfer-Encoding* header field.

CONVERSION = *NO
The user data is treated as binary data. In other words, no character set conversion takes place, for example. If the ENCODING operand has the (default) value *BIT-7, this value is replaced with *BASE64.

CONVERSION = *BY-CODED-CHAR-SET(...)
The user data is treated as text, for which character set conversion from an EBCDIC variant to an ISO-8859 variant usually has to be carried out. The conversion is carried out with the help of XHCS. You will find more detailed information on XHCS in the “XHCS (BS2000/OSD)” manual. When XHCS in version 2.0 or higher is available, then also the Unicode related CCSNs UTF8, UTF16, UNICODE and UTFE can be used.

SOURCE = <name 1..8>
Source XHCS CCSN of the user data

DESTINATION = <name 1..8>
Destination XHCS CCSN of the user data

CHARSET-NAME = <c-string 1..63 with-lower-case>
If the contents of the message consist of text (see the CONTENT-TYPE operand), the value of the operand is inserted as a character set parameter in the *Content-Type* header field. If the contents do not consist of text, the operand is ignored.

ENCODING = *BIT-7 / *BIT-8 / *QP / *BASE64
Specifies how the user data is encoded for the transfer.

ENCODING = *BIT-7
After character set conversion, it should be possible to represent all characters as 7-bit ASCII characters. If this is not the case, some characters may be altered during mail transport. The value “7bit” is entered in the *Content-Transfer-Encoding* header field. If your mail contains lines with more than 998 characters (without the trailing CR LF), use ENCODING=*QP or ENCODING=*BASE64, because otherwise your mail will be most probably truncated or otherwise altered during mail transport.

ENCODING = *BIT-8
The value “8bit” is entered in the *Content-Transfer-Encoding* header field.

When you are not sure whether all mail servers in the transport chain are "8-bit-clean", you should use ENCODING=*QP or ENCODING=*BASE64. The same recommendation applies, when your mail contains lines with more than 998 characters (without the trailing CR LF), because otherwise your mail will be most probably truncated or otherwise altered during mail transport.

ENCODING = *QP
After character set conversion, all characters which are not directly represantable as 7 bit characters are encoded in accordance with the quoted-printable algorithm (see RFC 2045). This results in lines of at most 80 7 bit characters. This encoding is helpful, above all, when only a few characters have to be converted. In this case, the great majority of the data remains readable to humans. The value “quoted-printable” is entered in the *Content-Transfer-Encoding* header field.

ENCODING = *BASE64
After character set conversion, the data is encoded in accordance with the Base64 algorithm (see RFC 2045). This results in lines of at most 80 7 bit characters. This encoding is effective, above all, for binary data when only a few characters can directly be represented as 7-bit ASCII characters. In this case, quoted-printable encoding would result in much more data than Base64 encoding and would not be readable to humans. Base64 encoding increases the volume of data by a third. The value “base64” is entered in the *Content-Transfer-Encoding* header field.

CONTENT-TYPE = *STD / <c-string 1..255 with-lower-case>
The value of this operand is entered in the message’s *Content-Type* header field. This field displays to the receiving client the medium type of the message transferred (e.g. pure text, text editor file, image file, movie file).

Example

For human-readable text, specify “text/plain”. For images in JPEG format, specify “image/jpeg”.

CONTENT-TYPE = *STD
The *Content-Type* header field is set to the value “text/plain” and the value of the CHARSET-NAME operand.

CONTENT-TYPE = <c-string 1..255 with-lower-case>
The value of this operand is entered in the *Content-Type* header field. Use a value that corresponds to your data.

CONTENT-DISPOSITION = *ATTACHMENT / *INLINE /
The value of this operand is entered in the message's *Content-Disposition* header field (see RFC 2183). This indicates to the receiving mail client whether this part of the e-mail is to be displayed automatically or only when requested by the user who reads the e-mail.

CONTENT-DISPOSITION = *ATTACHMENT
The message's *Content-Disposition* header field is set to "attachment". This indicates to the receiving mail client that this part of the message body is only to be displayed on the request of the user.

CONTENT-DISPOSITION = *INLINE
The message's *Content-Disposition* header field is set to "inline". This indicates to the receiving mail client that this mail body is to be displayed automatically when the message is displayed.

SECURITY = *NO / *SMIME(…)
Specifies whether the e-mail is to be encrypted and/or signed.

SECURITY = *NO
The e-mail is neither encrypted nor signed.

SECURITY = *SMIME(…)
The e-mail is encrypted and/or signed with S/MIME.

SIGNING = *USER-OPTION / *NO / *YES(…)
Specifies whether the e-mail is to be signed.

SIGNING = *USER-OPTION
The e-mail is signed depending on the setting in the user option file.

SIGNING = *NO
The e-mail is not signed.

SIGNING = *YES(…)
The e-mail is signed.

CERTIFICATE-FILE = *USER-OPTION / <filename 1..54 without-gen>
Specifies the file in which the X.509 certificate used for the signing is saved.

CERTIFICATE-FILE = *USER-OPTION
The file with the X.509 certificate to be used must be specified in the user option file.

CERTIFICATE-FILE = <filename 1..54 without-gen>
Name of the file containing the X.509 certificate to be used. The certificate must be saved in PEM format.

KEY-FILE = *USER-OPTION / <filename 1..54 without-gen>
Specifies the file containing the private key belonging to the X.509 certificate specified in the CERTIFICATE-FILE parameter.

KEY-FILE = *USER-OPTION
The file with the private key to be used must be specified in the user option file.

KEY-FILE = <filename 1..54 without-gen>
Name of the file containing the private key to be used. The private key must be saved in PEM format.

ADDITIONAL-CERT-FILE = *USER-OPTION / *NONE / <filename 1..54 without-gen>
Specifies a file with additional X.509 certificates that can be used for signing. These certificates help the recipient to verify the signature if the certificate specified in the CERTIFICATE-FILE operand was published by an intermediate CA (certificate authority) rather than a root CA.

ADDITIONAL-CERT-FILE = *USER-OPTION
If a file with additional X.509 certificates is required, it must be specified in the user option file.

ADDITIONAL-CERT-FILE = *NONE
No additional certificates are used.

ADDITIONAL-CERT-FILE = <filename 1..54 without-gen>
Name of the file containing the additional certificates to be used. The certificates must be saved in PEM format.

ENCRYPTING = *USER-OPTION / *NO / *YES(…)
Specifies whether the e-mail is sent encrypted.

ENCRYPTING = *USER-OPTION
The encryption of the e-mail depends on the setting in the user option file.

ENCRYPTING = *NO
The e-mail is not encrypted.

ENCRYPTING = *YES(…)
The e-mail is encrypted.

CERTIFICATE-FILE = *USER-OPTION / <filename 1..54 without-gen>
Specifies the file in which the X.509 certificates used for encryption are saved.

CERTIFICATE-FILE = *USER-OPTION
The file containing the X.509 certificates to be used must be specified in the user option file.

CERTIFICATE-FILE = <filename 1..54 without-gen>
Name of the file containing the X.509 certificates to be used. The certificates must be specified in PEM format.

CIPHER = *USER-OPTION / *DES3 / *DES / *RC2-40 / *RC2-64 / *RC2-128 / *AES-128 / *AES-192 / *AES-256
Specifies the encryption algorithm to be used. The selected encryption algorithm must be supported by all e-mail recipients. If it is not, some of the recipients will not be able to decrypt the e-mail.

CIPHER=*USER-OPTION
The wanted cipher must be named in the user option file.

CIPHER = *DES3
Encryption with Triple DES. The effective length is 112 bits.

CIPHER = *DES
Encryption with DES.
You should only select this method when there are no better alternatives available, since the key length (56 bits) in this method is now considered to be too short.

CIPHER = *RC2-40
Encryption with RC2-40.
You should only select this method when there are no better alternatives available, since the key length (40 bits) in this method is now considered to be much too short.

CIPHER = *RC2-64
Encryption with RC2-64.
You should only select this method when there are no better alternatives available, since the key length (64 bits) in this method is now considered to be too short.

CIPHER = *RC2-128
Encryption with RC2-128. The key length is 128 bits.

CIPHER = *AES-128
Encryption with AES-128.
The key length is 128 bits. AES is a successor of DES/3DES and still relatively new. AES may therefore not yet be supported by all e-mail programs.

CIPHER = *AES-192
Encryption with AES-192.
The key length is 192 bits. AES is a successor of DES/3DES and still relatively new. AES may therefore not yet be supported by all e-mail programs.

CIPHER = *AES-256
Encryption with AES-256.
The key length is 256 bits. AES is a successor of DES/3DES and still relatively new. AES may therefore not yet be supported by all e-mail programs.

CRL-FILE = *USER-OPTION / *NONE / <filename 1..54 without-gen>
Specifies the file containing the CRL (certificate revocation list). The CRL is used to check the validity of recipient certificates.

CRL-FILE = *USER-OPTION
The file containing the CRL to be used is specified in the user option file. If *NONE is specified in the user option file (default), the recipient certificates are not checked for validity.

CRL-FILE = *NONE
A CRL is not used. In other words, the recipient certificates are not checked for validity.

CRL-FILE = <filename 1..54 without-gen>
File containing the CRL to be used.

CA-CERTIFICATES-FILE = *USER-OPTION / *NONE / <filename 1..54 without-gen>
Specifies the file containing the certificates in PEM format required to check the validity of the recipient certificates (see the CRL-FILE operand).

CA-CERTIFICATES-FILE = *USER-OPTION
The file containing the CA certificates to be used is specified in the user option file.

CA-CERTIFICATES-FILE = *NONE
A file containing CA certificates is not used.

CA-CERTIFICATES-FILE = <filename 1..54 without-gen>
File containing the certificates.

USER-OPTION-FILE = *STD / <filename 1..54 without-gen>
Specifies a user option file containing default values for different operands. You will find a detailed description of the user option file in the section “Configuration file for the mail sender frontend (user option file)” on page 385.

USER-OPTION-FILE = *STD
The default user options are obtained from the file specified in the *SYSSSI* option file with the *defaultOptionFileName* option (see “interNet Services BS2000, Administrator Guide”). The default setting for this option is *SYSDAT.MAIL.nnn.USER.OPT.*

USER-OPTION-FILE = <filename 1..54 without-gen>
The file specified here is used as the user option file.

WAIT-FOR-RESULT = *NO(...) / *YES
The operand specifies whether the command is terminated immediately after the mail send order is sent or whether the command is to wait for the order to be carried out.

WAIT-FOR-RESULT = *NO(...)
The command does not wait until the transfer order to the mail server is completed.

RESULT = *DISCARD
The execution status is not secured. It therefore cannot be queried subsequently by means of the REQUEST-MAIL-ORDER-RESULT command (see page 411).

RESULT = *BY-REQUEST-CMD
You can and should query the execution status subsequently by using the REQUEST-MAIL-ORDER-RESULT command (see page 411). If the status is not queried, the status information takes up storage space in the ASTI subsystem for an unlimited period. ASTI saves this status information in SYS.* files under the user ID of the mail sender.

WAIT-FOR-RESULT = *YES
The command waits for the mail send order to be completed and indicates whether the transfer to the (first) mail server was successful.

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error.
	64	CMD0216	The user does not have the required authorization for the command.
	32	CMD0220	Internal error.
	32	CMD2009	Error during the creation of the S variable.
	64	YML0120	ASTI subsystem is not available.
	64	YML0142	User option file does not exist or is not readable.
	64	YML0144	File specified as an attachment does not exist or is not readable.
	64	YML0146	Mail send order is too large.
	128	YML0148	Maximum number of orders reached.
	64	YML0171	SMTP protocol error.
	64	YML0172	Mail service not activated.
	64	YML0174	No sender address specified.
	32	YML0176	Unexpected ASTI error.
	64	YML0203	S/MIME file has an error or was not found.
	128	YML0214	Resources are exhausted.

output data

If a mail send order is successfully transferred to ASTI, a YML0160 message is output containing the order ID as an insert. If requested, the order ID is also stored in an S variable with the ORDER-ID component.

When the SEND-MAIL command is used synchronously (WAIT-FOR-RESULT=*YES), than additionally a YML0170 message (or in case of an error another adequate error message) is displayed. If requested the components RETURN-CODE and RETURN-MSG of an OPS variable are supplied with corresponding values (see also the description of the “Output data” on page 413 at the REQUEST-MAIL-ORDER-RESULT command).

Examples:

```
/EXECUTE-CMD CMD=(SEND-MAIL -
/                 TO='Heinrich.Schuetz@dresden.example', -
/                 SUBJECT='Opus ultimum', -
/                 WAIT=*NO(RESULT=*BY-REQUEST-CMD)), -
/            STRUCTURE-OUTPUT=OUT
%  YML0160 MAIL SEND ORDER SUBMITTED WITH ORDER ID
'077AF4BF00000046'

/SHOW-VARIABLE OUT
OUT(*LIST).ORDER-ID = 077AF4BF00000046
OUT(*LIST).RETURN-CODE = 0k
OUT(*LIST).RETURN-MSG =

/EXECUTE-CMD CMD=(SEND-MAIL -
/                 TO='Heinrich.Schuetz@dresden.example', -
/                 SUBJECT='Opus ultimum', -
/                 WAIT=*YES), -
/            STRUCTURE-OUTPUT=OUT
%  YML0160 MAIL SEND ORDER SUBMITTED WITH ORDER ID
'077AF4BF0000004F'
%  YML0170 MAIL SEND ORDER CARRIED OUT; MAIL SERVER RESULT: '250
OK: QUEUED AS 5720B6EC20'

/SHOW-VARIABLE OUT
OUT(*LIST).ORDER-ID = 077AF4BF0000004F
OUT(*LIST).RETURN-CODE = 0k
OUT(*LIST).RETURN-MSG = 250 Ok: queued as 5720B6EC20
```

REQUEST-MAIL-ORDER-RESULT - Request mail result

Domain:
UTILITIES

Required authorization:
STD-PROCESSING
TSOS

You can use the REQUEST-MAIL-ORDER-RESULT command to query the execution status of the mail send order. The status is returned by the backend task after the order is processed. The order data is deleted from the internal ASTI tables. The associated SYS file under the user ID of the mail sender is deleted.

The REQUEST-MAIL-ORDER-RESULT command does not provide any information on orders that have not yet been processed. In this case, you can use the SHOW-MAIL-ORDER-STATUS command to get information on the order (see page 416). The REQUEST-MAIL-ORDER-RESULT command supports structured output in S variables (see the manual “Commands, Volume 6, S Variables”).

REQUEST-MAIL-ORDER-RESULT
ORDER-ID = *__ANY__ / <x-text 1..16> ,**WAIT-FOR-RESULT** = *__NO__ / ***YES** ,**USER-OPTION-FILE**=__***STD***__ / <filename 1..54 without-gen>

Operands

ORDER-ID = *__ANY__ / <x-text 1..16>
Specifies the order whose execution status is queried.

ORDER-ID = *__ANY__
The execution status of a completed order sent by the command caller is queried. If there are several completed orders, it is not specified which of these orders’ status is queried.

ORDER-ID = <x-text 1..16>
ASTI order ID of the order whose execution status is queried.

WAIT-FOR-RESULT = *__NO__ / *YES
Specifies whether the command waits until the processing of the order is completed.

WAIT-FOR-RESULT = *NO
The command does not wait for the end of an order that is not yet completed. You can query the execution status of the order subsequently by calling REQUEST-MAIL-ORDER-RESULT again.

WAIT-FOR-RESULT = *YES
The command waits until the order is completed. If ORDER-ID=*ANY is specified, waiting is implemented internally by means of regular queries at intervals of 60 seconds, so that waiting is completed an average of 30 seconds after the end of the order.

USER-OPTION-FILE = *STD / <filename 1..54 without-gen>
Specifies a user option file containing default values for different operands. You will find a detailed description of the user option file in the section “Configuration file for the mail sender frontend (user option file)” on page 385.

USER-OPTION-FILE = *STD
The default user options are obtained from the file specified in the *SYSSSI* option file with the *defaultOptionFileName* option (see “interNet Services BS2000, Administrator Guide”). The default setting for this option is *SYSDAT.MAIL.nnn.USER.OPT*.

USER-OPTION-FILE = <filename 1..54 without-gen>
The file specified here is used as the user option file.

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error.
	32	CMD0220	Internal error.
	32	CMD2009	Error during creation of the S variable.
	64	YML0120	ASTI subsystem not available.
	32	YML0176	Unexpected ASTI error.
	64	YML0210	Order was not found.
	128	YML0214	Resources are exhausted.
	64	YML0215	A result was not requested when the order was issued.
	64	YML0216	The order was executed by an external task.
	128	YML0222	Order not completed.

Examples

```
/REQUEST-MAIL-ORDER-RESULT ORDER-ID=02BC49BB0000000D
%  ORDER-ID:              02BC49BB0000000D
%  RETURN CODE:           0k
%  RETURN MESSAGE:        250 Ok: queued as 0E7026E860
```

```
/EXECUTE-CMD CMD=(REQUEST-MAIL-ORDER-RESULT ORDER-ID= 02BC49BB00000017),-
/            STRUCTURE-OUTPUT=OUT
%  ORDER-ID:              02BC49BB00000017
%  RETURN CODE:           Ok
%  RETURN MESSAGE:        250 Ok: queued as 2B9AE6E860

/SHOW-VARIABLE OUT
OUT(*LIST).ORDER-ID =      02BC49BB00000017
OUT(*LIST).RETURN-CODE =   Ok
OUT(*LIST).RETURN-MSG =    250 Ok: queued as 2B9AE6E860
```

Output data

When an order is terminated, the command outputs three lines containing information on the execution status.

- The first line (ORDER-ID) specifies the ID of the order for which the data is specified. This is relevant, above all, if ORDER-ID=*ANY was specified.
- The second line (RETURN CODE) specifies whether an error has occurred and, if so, which one. The possible values are:

  OK
  : The e-mail was sent successfully.

  Error during SMTP protocol
  : The mail server reported an error to the backend task.

  Error during S/MIME operation
  : An error occurred during S/MIME processing (e.g. certificate problems).

  Internal error
  : In the case of all other errors.

- The third line (RETURN MESSAGE) contains the following information:
  - In the event of an error:
    Additional textual information on the error.
  - In the event of a successfully terminated order:
    Concluding message of the mail server. This message generally contains the (partial) message ID, indicating that the e-mail was rejected by the server. If an error occurs subsequently in the course of mail transfer, the administrator can use this ID to trace the transfer of the e-mail across the chain of mail servers.

The data can also be transferred to an S variable structured in accordance with the output of the command.

DELETE-MAIL-ORDER - Delete mail

Domain:
UTILITIES

Required authorization:
STD-PROCESSING
TSOS

You can use the DELETE-MAIL-ORDER command to delete mail send orders transferred to the ASTI subsystem that have not yet been processed. For orders that have already been processed, the DELETE-MAIL-ORDER command deletes the information on the associated execution status in the ASTI-internal tables. The associated SYS files under the user ID of the mail sender are also deleted.

DELETE-MAIL-ORDER
ORDER-ID = *__ALL__ / <x-text 1..16> ,**SENDER-USERID** = *__OWN__ / ***ANY** / <name 1..8>

Operands

ORDER-ID = *__ALL__ / <x-text 1..16>
Specifies which orders are to be deleted.

ORDER-ID = *__ALL__
All orders of the user ID specified under SENDER-USERID are deleted. It can happen that new e-mails are also deleted that are sent by another task of the same user ID parallel to the execution of DELETE-MAIL-ORDER.

If DELETE-MAIL-ORDER returns an error, it may not have been possible to delete all orders. In this case, repeat the command in order to delete the remaining orders.

ORDER-ID = <x-text 1..16>
The order with the specified ASTI order ID is deleted if the specified user is the owner of the order. Only users with TSOS authorization can delete orders with a different user ID.

SENDER-USERID = *__OWN__ / *ANY / <name 1..8>
Specifies the users whose orders are to be deleted.

SENDER-USERID = *OWN
Only the orders of the command caller are deleted.

SENDER-USERID = *ANY
For users without TSOS authorization, this has the same effect as specifying *OWN. For users with TSOS authorization, it means that the orders of all users are deleted.

SENDER-USERID = <name 1..8>
User ID of the owner of the orders to be deleted. Only users with TSOS privileges can delete orders of another user ID.

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error.
	64	CMD0216	The user does not have the required authorization for the command.
	32	CMD0220	Internal error.
	64	YML0120	The ASTI subsystem is not available.
	32	YML0176	Unexpected ASTI error.
	64	YML0210	Order not found.
	64	YML0216	Order executed by external task.
	64	YML0220	The user is not entitled to delete this order.
	64	YML0221	The user is not authorized to delete orders of other users.

SHOW-MAIL-ORDER-STATUS - Query information on mail orders

Domain:
UTILITIES

Required authorization:
STD-PROCESSING
TSOS

You can use the SHOW-MAIL-ORDER-STATUS command to query the status of the mail send orders sent.

For all current orders, SHOW-MAIL-ORDER-STATUS indicates whether they:

– Are waiting to be executed
– Are currently being executed or
– Have already been executed

The SHOW-MAIL-ORDER-STATUS command supports structured output in S variables (see the manual “Commands, Volume 6, S Variables”).

SHOW-MAIL-ORDER-STATUS
ORDER-ID = ***ALL** / list-poss(30): <x-text 1..16> ,**SENDER-USERID** = ***OWN** / ***ANY** / <name 1..8> ,**INFORMATION** = ***SUMMARY** / ***ALL**

Operands

ORDER-ID = *ALL / list-poss(30): <x-text 1..16>
Specifies which orders are selected.

ORDER-ID = *ALL
All orders of the users specified by means of the SENDER-USERID operand are selected.

ORDER-ID = list-poss(30): <x-text 1..16>
Here you can specify up to 30 orders to be selected. Only users with TSOS authorization can specify orders that have a different user ID.

SENDER-USERID = *OWN / *ANY / <name 1..8>
Specifies the users whose orders are to be selected.

SENDER-USERID = *OWN
Only the orders of the command caller are selected.

SENDER-USERID = *ANY
For users without TSOS authorization, this has the same effect as specifying *OWN. For users with TSOS authorization, it means that the orders of all users are selected.

SENDER-USERID = <name 1..8>
User ID whose orders are to be selected. Only users with TSOS authorization can specify orders of another user ID.

INFORMATION = *SUMMARY / *ALL
Specifies which information on the orders is output.

INFORMATION = *SUMMARY
Only the summaries of the order categories are output.

INFORMATION = *ALL
All the available information on the selected orders is output.

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error.
	64	CMD0216	The user does not have the required authorization for the comand.
	32	CMD0220	Internal error.
	32	CMD2009	Error during creation of the output variable.
	64	YML0120	The ASTI subsystem is not available.
	32	YML0176	Unexpected ASTI error.
	64	YML0210	Order not found.
	128	YML0214	Resource utilization.
	64	YML0216	Order executed by another application.
	64	YML0218	Order not a mail send order.

Output data

- With the operand INFORMATION=*SUMMARY, the mail send orders are selected on the basis of the ORDER-ID and SENDER-USERID operands and grouped into categories. The total is output. Distinctions are drawn between five categories:

WAITING	The order is waiting to be executed by the backend task.
DEFERRED	During execution by the backend task, a time-limited error occurred (when the mail server was shut down, for example). After a certain time, another attempt is made to execute the order.
ACTIVE	The order is currently being executed by the backend task. The application does not support parallel processing. The number of orders in this category can therefore only be 0 or 1, except when an error occurs.
SENT	The order was successfully executed.
FAILED	An error occurred during the processing of the order.

  The data can be stored in an S variable containing the names of the categories as components.

- If INFORMATION=*ALL is specified, several lines are output for each order selected.

The following table gives the labels of the SYSOUT lines and the corresponding component names of the S variables. If a certain value does not exist, the associated line or variable component is suppressed.

SYSOUT	**S variable**	**Meaning**
ORDER-ID	ORDER-ID	Order ID
STATE	STA	Order status (WAITING, DEFERRED, ACTIVE, SENT, FAILED)
SEND TIME	SEND-TIME	Send time of the e-mail
SENDER	SENDER	User ID of the mail sender
RETURN CODE	RETURN-CODE	Return code of the backend task when STATUS = SENT or FAILED is specified
RETURN MESSAGE	RETURN-MSG	End-of-order message of the mail server or plain-text error message when STATUS = SENT or FAILED is sent
FROM	FROM	Value of the SEND-MAIL operand FROM
TO	TO	Value of the SEND-MAIL operand TO
CC	CC	Value of the SEND-MAIL operand CC
BCC	BCC	Value of the SEND-MAIL operand BCC

Examples

```
/EXECUTE-CMD CMD=(SHOW-MAIL-ORDER-STATUS INFORMATION=*SUMMARY),-
/            STRUCTURE-OUTPUT=OUT
%  # ORDERS
%    WAITING:     1
%    DEFERRED:    0
%    ACTIVE:      0
%    SENT:        1
%    FAILED:      0
%    TOTAL:       2


/SHOW-VARIABLE OUT
OUT(*LIST).WAITING = 1
OUT(*LIST).DEFERRED = 0
OUT(*LIST).ACTIVE = 0
OUT(*LIST).SENT = 1
OUT(*LIST).FAILED = 0


/EXECUTE-CMD CMD=(SHOW-MAIL-ORDER-STATUS ORDER=*ALL,INFORMATION=*ALL),-
/            STRUCTURE-OUTPUT=OUT
%  ORDER_ID:            01E7E48600000025
%  STATUS:              Waiting
%  SUBMISSION TIME:     2010-02-10 17:23:18
%  SUBMITTER:           CLAUDIO
%  FROM:                Claudio.Monteverdi@mantova.example
%  TO:                  Heinrich.Schuetz@dresden.example
%  CC:                  John.Bull@london.example
%  BCC:                 William.Byrd@london.example
%  ---
%  ORDER_ID:            01E7E48600000013
%  STATUS:              Sent
%  SUBMISSION TIME:     2010-02-10 11:05:54
%  SUBMITTER:           HEINRICH
%  RETURN CODE:         Ok
%  RETURN MESSAGE:      250 Ok: queued as BDB6F6EA2F
%  FROM:                Heinrich.Schuetz@dresden.example
%  TO:                  Claudio.Monteverdi@mantova.example


/SHOW-VARIABLE OUT
OUT(*LIST).ORDER-ID = 01E7E48600000025
OUT(*LIST).STA = Waiting
OUT(*LIST).SEND-TIME = 2010-02-10 17:23:18
OUT(*LIST).SENDER = CLAUDIO
OUT(*LIST).FROM = Claudio.Monteverdi@mantova.example
OUT(*LIST).TO = Heinrich.Schuetz@dresden.example
```

```
OUT(*LIST).CC = John.Bull@london.example
OUT(*LIST).BCC = William.Byrd@london.example
OUT(*LIST).ORDER-ID = 01E7E48600000013
OUT(*LIST).STA = Sent
OUT(*LIST).SEND-TIME = 2010-02-10 11:05:54
OUT(*LIST).SENDER = HEINRICH
OUT(*LIST).RETURN-CODE = Ok
OUT(*LIST).RETURN-MSG = 250 Ok: queued as BDB6F6EA2F
OUT(*LIST).FROM = Heinrich.Schuetz@dresden.example
OUT(*LIST).TO = Claudio.Monteverdi@mantova.example
```

9.3 Subprogram interface of the mail sender frontend

The subprogram interface of the mail sender frontend supports the programming languages ASSEMBLER and C.

This section describes the following:

- The ASSEMBLER macro interface
- Function calls in C

9.3.1 ASSEMBLER macro interface

The ASSEMBLER subprogram interface of the mail sender frontend is implemented by the ASSEMBLER macro interface of the DSSM subsystem MAILCLNT.

9.3.1.1 Properties

The following table shows the properties of the macro interface that apply to all macro calls:

Interface type:	CALL
Linkage:	ISL linkage
SVC number	SVC 20 (decimal)
Function area:	TU (**T**ask **U**nprivileged) / TPR (**T**ask **Pr**ivileged)
Macro type:	S
MF formats supported:	MF format 3: MF = {C | D | E | L | M}
ASSEMBLER	`[ PREFIX = Y ]` `[ MACID = MLS ]`

9.3.1.2 Macro calls (overview)

The following macro calls are available:

Macro call	Function
YMLSML	Send e-mail
YMLCML	Get result
YMLDML	Delete e-mail
YMLGML	Query information on the e-mail

9.3.1.3 Format for describing the macro calls

The descriptions of the macro calls all have the same structure:

Macro name - brief description of functionality

Description of functionality.

Entry names or SVC number(s)

Description of the entry names and SVC numbers.

Macro call format and operand description

Macro name
Operands ... . . .

Operand description

Return codes

SRC2	SRC1	MRC	MRC name	Meaning
00	00	0000	Identifier	Meaning of the return code
. . .	. . .	. . .	. . .	...

SRC1/2 = sub-return code 1/2 in hexadecimal notation
MRC = main return code in hexadecimal notation

Macro call parameters

Description of the data structure(s) for the macro call parameters.

Mail parameters

Description of the data structure(s) for the mail parameters.

9.3.1.4 Description of the macro calls

The various ASSEMBLER macro calls are described below.

YMLSML - Send mail

You can use this macro to send e-mails.

Entry names or SVC number(s)

SVC 20 (decimal)

UNIT=940, FUNCTION=20, VERSION=1 | 2

Macro call format and operand description

YMLSML
FL= ***TU** / ***TPR**
,**VERSION**=**1** / **2**
,**XPAND**=**PARAM** / **GENERAL** / **ADD_HEADER** / **DATA_SPEC** / **CHARSET** / **ENCODING** / **CONT_DISP** / **MSG_ATT**
,**MAILP**=<var: pointer>
,**MAILPL**=<integer: 1..32767> / <var: int:4>
,**OPTFILE**= ***NONE** / <var: char:54>
,**ENCRYPT**= ***NO** / ***YES** / ***OPTFILE**
,**SIGN**= ***NO** / ***YES** / ***OPTFILE**
,**SECPROT**= ***SMIME**
,**CIPHER**= ***3DES** / ***DES** / ***RC2-40** / ***RC2-64** / ***RC2-128** / ***AES-128** / ***AES-192** / ***AES-256** / ***OPTFILE**
,**WAIT**= ***NO-DISCARD** / ***NO** / ***YES**
,**WAITTIM=*UNLIM /** <integer: 1..65535> **/<**var: int:4>

FL=
Function area

***TU**
An SVC interface is generated.

***TPR**
A CALL interface is generated.

VERSION=
Selects the interface version.

1
Selects the old interface version.
This is the default.

2
Selects the new interface version, which offers the operand WAITTIM and additional return codes.

XPAND=
These parameters control the expansion of the data structures that describe the parameter list of the macro and the layout of the mail parameters. These data structures are referenced by the mail parameter MAILP. You will find a description of the data structures starting on page 429.

PARAM
A macro parameter list is generated.

GENERAL
A data layout is generated for the general mail parameters.

ADD_HEADER
A data layout is generated for the additional header lines.

DATA_SPEC
A data layout is generated for the specification of the mail data.

CHARSET
A data layout is generated for the specification of the character set.

ENCODING
A data layout is generated for the specification of the encoding.

CONT_DISP
A data layout is generated for the specification of the internal structure of the content.

MSG_ATT
A data layout is generated for the nesting of messages/attachments.

MAILP=
Parameter section for describing the e-mail. This parameter is mandatory.

IDENTIFIER
Variable in which the address of the parameter section is stored or register containing the address of the parameter section.

MAILPL=
Length of the parameter section for describing the e-mail. This parameter is mandatory.

INTEGER 1, 32767
Length of the parameter section.

IDENTIFIER
Variable in which the length of the parameter section is stored or register containing the length of the parameter section.

OPTFILE=
Name of the user option file (see page 385). The options in this file can be overwritten or options can be added by using macro parameters.

***NONE**
No file defined.

IDENTIFIER
Variable in which the name of the option file is stored or register containing the address of the name of the option file.

ENCRYPT=
Specifies whether the user wants to encrypt the message to be sent.

***NO**
The e-mail is not encrypted.

***YES**
The e-mail is encrypted.

***OPTFILE**
The specification is taken from the user option file (see page 385).

SIGN=
This parameter specifies whether the caller wants to sign the message to be sent.

***NO**
The e-mail is not signed.

***YES**
The e-mail is signed.

***OPTFILE**
The specification is taken from the user option file (see page 385).

SECPORT=
This parameter specifies the encryption/signing method. Currently, S/MIME is the only method supported.

***SMIME**
S/MIME is used.

CIPHER=
Symmetrical encryption code.

***3DES**
Triple DES

***DES**
DES

***RC2-40**
RC2 with 40-bit key length.

***RC2-64**
RC2 with 64-bit key length.

***RC2-128**
RC2 with128-bit key length.

***AES-128**
AES with 128-bit key length.

***AES-192**
AES with 192-bit key length.

***AES-256**
AES with 256-bit key length.

***OPTFILE**
The specification in the user option file (see page 385) is taken.

WAIT=
Specifies whether the caller wants to wait for the mail send order to be completed by the mail sender backend.

***NO-DISCARD**
Do not wait or instruct ASTI to reject the result status of the mail sender backend.

***NO**
Do not wait, but instruct ASTI to save the result status of the mail sender backend.

***YES**
Wait until the mail sender backend has completed the send order or reports an error sending the e-mail.

WAITTIME=
With this operand the wait time can be limited in the case of WAIT=*YES. The operand is only available with VERSION=2. When the maximal wait time is reached, the call is terminated with an appropriate return code.

***UNLIM**
Unlimited wait time. With this operand value the behaviour is as hitherto.

INTEGER 1,65535
Maximal wait time in seconds.

IDENTIFIER
Variable or register containing the wait time (in seconds).

Return codes

SRC2	SRC1	MRC	MRC name	Meaning
00	00	0000	YMLSSUCC	No error found.
00	01	0001	YMLSPARE	Parameter error.
00	20	0002	YMLSINTE	Internal error.
00	40	0003	YMLSMSYN	Syntax error in the mail parameter.
00	40	0004	YMLSOFNA	Option file not available.
00	40	0005	YMLSMFNA	Message or attachment not available.
00	40	0006	YMLSSFNA	File for SMIME not available.
00	40	0007	YMLSPTBG	Mail parameter too large.
00	40	0008	YMLSBACK	Backend error.
00	80	0009	YMLSMORD	Maximum number of orders exceeded.
00	80	0010	YMLSTIME	Maximal wait time reached.
00	80	000A	YMLSSNAV	Mail client service not available.
00	40	000B	YMLSAINV	Mail parameter address invalid.
00	40	000C	YMLSRSRC	Resources exhausted.
00	40	000D	YMLSANAV	ASTI subsystem not available.
00	40	000E	YMLSNFRA	No FROM address specified.
00	20	000F	YMLSASTI	Unexpected ASTI error.

SRC1/2 = sub-return code 1/2 in hexadecimal notation
MRC = main return code in hexadecimal notation

Macro call parameters

The data structure of the macro call parameters of YMLSML is composed as follows:

VERSION=1:

Distance	Identifier	Value	Meaning
	YMLSPARL		Parameter section
000	YMLSHDR		Function header
008	YMLSIND		Input parameter
008	YMLSMPAR		Address of the mail parameter area
00C	YMLSMPRL		Length of the mail parameter area
010	YMLSWAIT		Wait for the end of the mail send order?
	YMLSWYES	1	Yes
	YMLSWNO	2	No; ASTI discards order execution status.
	YMLSWRES	3	No; ASTI does not discard order execution status.
011	YMLSSPRO		Protocol for encryption and signing
	YMLSSMIM	1	S/MIME
012	YMLSENC		Encryption?
	YMLSGYES	1	Yes
	YMLSGNO	2	No
	YMLSGOPT	3	Value from user option file
013	YMLSSIGN		Signing?
	YMLSGYES	1	Yes
	YMLSGNO	2	No
	YMLSGOPT	3	Value from user option file
014	YMLSCPHR		Encryption algorithm:
	YMLSRC2O	1	RC2-40
	YMLSRC24	2	RC2-64
	YMLSRC28	3	RC2-128
	YMLSDES	4	DES
	YMLS3DES	5	3DES
	YMLSAES8	6	AES-128
	YMLSAES2	7	AES-192
	YMLSAES6	8	AES-256
	YMLSFOPT	127	Value from user option file
015	YMLSOPTF		Name of user option file

Distance	Identifier	Value	Meaning
04B	YMLSRSV1		Reserved area
04C	YMLSOUTD		Output parameter
04C	YMLSOID		Order ID for referencing the mail send order if the caller does not wait for the transfer to be completed.
05C	YMLSRETC		Return code from backend
	YMLSBOK	00	Ok
	YMLSBPER	01	Parameter error
	YMLSBRSC	02	Resource saturation
	YMLSBSMT	03	SMTP error
	YMLSBSMI	04	S/MIME error
	YMLSBINT	0A	Internal error
060	YMLSRETM		May contain error messages in text format (e.g. SMTP error messages of the SMTP server).
100	YMLSARET		If the YMLSML call returns the return code YMLSASTI, this field contains the ASTI return code.

VERSION=2:

Distance	Identifier	Value	Meaning
	YMLSPARL		Parameter area
000	YMLSHDR		Function header
008	YMLSIND		Input parameters
008	YMLSMPAR		Adress of mail parameter area
00C	YMLSMPRL		Length of mail parameter area
010	YMLSWTTM		Maximal wait time
014	YMLSWAIT		Wait for completion of mail send order?
	YMLSWYES	1	Yes
	YMLSWNO	2	No; ASTI discards order execution status.
	YMLSWRES	3	No; ASTI does not discard order execution status.
015	YMLSSPRO		Protocol for encryption and signing
	YMLSSMIM	1	S/MIME
016	YMLSENC		Encryption?
	YMLSGYES	1	Yes
	YMLSGNO	2	No
	YMLSGOPT	3	Value from user option file

Distance	Identifier	Value	Meaning
017	YMLSSIGN		Signing?
	YMLSGYES	1	Yes
	YMLSGNO	2	No
	YMLSGOPT	3	Value from user option file
018	YMLSCPHR		Encryption algorithm:
	YMLSRC20	1	RC2-40
	YMLSRC24	2	RC2-64
	YMLSRC28	3	RC2-128
	YMLSDES	4	DES
	YMLS3DES	5	3DES
	YMLSAES8	6	AES-128
	YMLSAES2	7	AES-192
	YMLSAES6	8	AES-256
	YMLSFOPT	127	Value from user option file
019	YMLSOPTF		Name of user option file
04F	YMLSRSV1		Reserved area
050	YMLSOUTD		Output parameters
050	YMLSOID		Order ID for referencing the mail send order if the caller does not wait for the transfer to be completed.
060	YMLSRETC		Returncode from backend
	YMLSBOK	00	Ok
	YMLSBPER	01	Parameter error
	YMLSBRSC	02	Resource saturation
	YMLSBSMT	03	General SMTP error
	YMLSBSMI	04	S/MIME error
	YMLSBSMM	05	Error with SMTP MAIL command
	YMLSBSMR	06	Error with SMTP RCPT command
	YMLSBSMD	07	Error with SMTP DATA command
	YMLSBFAC	08	Error at access to user option file
	YMLSBMTL	09	Mail too large
	YMLSBINT	0A	Internal error
064	YMLSRETM		May contain error messages in text format (e.g. SMTP error messages of the SMTP server).

Distance	Identifier	Value	Meaning
104	YMLSARET		This field contains the ASTI return code in case the YMLSSML call is rejcected with the YMLSASTI return code.
108	YMLSMID		Message ID
10F	YMLSRSV2		Reserved area

Supplementary notes to the output parameters:

YMLSRETC

Some failure situations, where with VERSION=1 only a collective return code is delivered, are covered with VERSION=2 by specific return codes for finding out the real error reason more quickly.

For instance the return code YMLSBMM is a strong hint that there is an error in the sender address. Accordingly the return code YMLSBMR indicates an error in at least one of the recipient addresses. A YMLSBMD can e.g. occur when the SMTP server checks the violation of certain rules only after the DATA command, the reason for the rule violation can be related to the specified addresses. In any case the field YMLSRETM should give further (text) information related to the error cause.

YMLSRETM

If the SMTP server used supports RFC 2034, then this field contains a machine interpretable error indicator, which in general is more specific than the SMTP related return codes in the YMLSRETC field.

YMLSMID

This field contains a YML message key for a message describing the occured error. This is normally the same message as a corresponding SEND-MAIL command call would deliver.

Listing of the expansion of the data structure for the macro call parameters

Expansion:

XPAND= PARAM

```
                         YMLSML MF=D,XPAND=PARAM
                  1      MFTST MF=D,PREFIX=Y,MACID=MLS,ALIGN=F,
                  1            DMACID=MLS,SUPPORT=(E,D,C,M,L),DNAME=MLSPARL
000000            2 YMLSPARL DSECT ,
                  1 *   Parameter area
                  1 YMLSHDR  FHDR  MF=(C,YMLS),EQUATES=NO                        FHDR
                  1 *   main return codes
         00000000 1 YMLSSUCC EQU   0                       No error detected
         00000001 1 YMLSPARE EQU   1                       Parameter error
         00000002 1 YMLSINTE EQU   2                       Internal error
         00000003 1 YMLSMSYN EQU   3                       Syntax error
         00000004 1 YMLSOFNA EQU   4                       Option file not
                                                           accessible
         00000005 1 YMLSMFNA EQU   5                       Message or attachment
                                                           file
                  1 *                                      not accessible
         00000006 1 YMLSSFNA EQU   6                       SMIME related file not
                  1 *                                      accessible
         00000007 1 YMLSPTBG EQU   7                       Mail parameter too big
         00000008 1 YMLSBACK EQU   8                       Back-end error
         00000009 1 YMLSMORD EQU   9                       Max number of orders
                                                           exceeded
         0000000A 1 YMLSSNAV EQU   10                      Mailclient Service not
                  1 *                                      available
         0000000B 1 YMLSAINV EQU   11                      Mailpar address invalid
         0000000C 1 YMLSRSRC EQU   12                      Resource saturation
         0000000D 1 YMLSANAV EQU   13                      Subsystem ASTI not
                                                           available
         0000000E 1 YMLSNFRA EQU   14                      No FROM address specified
         0000000F 1 YMLSASTI EQU   15                      Unexpected ASTI error
                  1 *
                  1 *
000008            1 YMLSIND  DS    0XL68                  Input parameters
000008            1 YMLSMPAR DS    A                      Mail parameter area
00000C            1 YMLSMPRL DS    F                      Mail parameter area
                                                           length
000010            1 YMLSWAIT DS    FL1                    wait
                  1 *   Wait operand values
         00000001 1 YMLSWYES EQU   1                       YES
         00000002 1 YMLSWNO  EQU   2                       NO-DISCARD
         00000003 1 YMLSWRES EQU   3                       NO
                  1 *
000011            1 YMLSSPRO DS    FL1                    Protocol for encryption
                                                           and
                  1 *                                      signing
                  1 *   Mail encryption/signing protocol
         00000001 1 YMLSSMIM EQU   1                       SMIME
```

```
                    1 *
000012              1 YMLSENC  DS    FL1                       Encryption
                    1 *   General yes or no selection
        00000001    1 YMLSGYES EQU   1                         YES
        00000002    1 YMLSGNO  EQU   2                         NO
        00000003    1 YMLSGOPT EQU   3                         OPTFILE
                    1 *
000013              1 YMLSSIGN DS    FL1                       Signing
000014              1 YMLSCPHR DS    FL1                       Cipher
                    1 *   Cipher operand values
        00000001    1 YMLSRC20 EQU   1                         RC2-40
        00000002    1 YMLSRC24 EQU   2                         RC2-64
        00000003    1 YMLSRC28 EQU   3                         RC2-128
        00000004    1 YMLSDES  EQU   4                         DES
        00000005    1 YMLS3DES EQU   5                         3DES
        00000006    1 YMLSAES8 EQU   6                         AES-128
        00000007    1 YMLSAES2 EQU   7                         AES-192
        00000008    1 YMLSAES6 EQU   8                         AES-256
        0000007F    1 YMLSFOPT EQU   127                       OPTFILE
                    1 *
000015              1 YMLSOPTF DS    CL54                      Option file
00004B              1 YMLSRSV1 DS    CL1                       Reserved
                    1 *
                    1 *
00004C              1 YMLSOUTD DS    0XL184                    Output parameters
00004C              1 YMLSOID  DS    CL16                      Order Id
00005C              1 YMLSRETC DS    F                         Return code
                    1 *   rc
        00000000    1 YMLSBOK  EQU   0                         OK
        00000001    1 YMLSBPER EQU   1                         Parameter error
        00000002    1 YMLSBRSC EQU   2                         Resource saturation
        00000003    1 YMLSBSMT EQU   3                         SMTP error
        00000004    1 YMLSBSMI EQU   4                         SMIME error
        0000000A    1 YMLSBINT EQU   10                        Internal error
                    1 *
000060              1 YMLSRETM DS    CL160                     Return message
000100              1 YMLSARET DS    F                         Return code from ASTI
                    1 *
        00000104    1 YMLS#    EQU   *-YMLSHDR
```

Mail parameters

An e-mail consists of many constituent parts, some of which are optional, whose length is generally variable. Data structures based on tuples of a certain format (type, length, value) are therefore used to specify the mail parameters.

Each parameter is a 2-byte integer field. The field contains one of the values listed in the table below.

Value name	Tag #	Meaning
YMLSFROM	1	Envelope sender address (is also placed in the FROM header line if YMLSFFRM is not specified).
YMLSFFRM	2	Sender address in the FROM header line.
YMLSTO	3	List of envelope recipient addresses separated by commas (is also placed in the TO header line if YMLSFTO is not specified).
YMLSFTO	4	List of recipient addresses separated by commas. Is placed in the To header line.
YMLSCC	5	List of envelope recipient addresses separated by commas (is also placed in the CC header line if YMLSFCC is not specified).
YMLSFCC	6	List of recipient addresses separated by commas. Is placed in the CC header line.
YMLSBCC	7	List of envelope recipient addresses separated by commas.
YMLSRPLT	8	Address in the REPLY-TO header line.
YMLSSBJT	9	Text for the SUBJECT header line.
YMLSAHDR	10	Additional header lines in which the caller can define the field names.
YMLSMBEG	11	Beginning of the definition of the message text.
YMLSMEND	12	End of the definition of the message text.
YMLSABEG	13	Beginning of the definition of the attachment.
YMLSAEND	14	End of the definition of the attachment.
YMLSDSPC	15	Data definition for message/attachment.
YMLSCHST	16	Character set definition for header/message/attachment.
YMLSTRCD	17	Transfer encoding of message/attachment.
YMLSCTTT	18	Content type of message/attachment.
YMLSTCTD	19	Presentation note for message/attachment.
YMLSKEYF	21	Key file (see the user option file “privateKeyFile” on page 388).
YMLSSGNF	22	File containing an X.509 certificate for signing e-mails with S/MIME (see the user option “signerCertificateFile” on page 388).

Value name	Tag #	Meaning
YMLSASCF	23	File containing additional X.509 certificates for signing e-mails with S/MIME (see the user option “addSignerCertificatesFile” on page 389).
YMLSRCTF	24	File containing X.509 certificates for encrypting e-mails with S/MIME (see the user option “recipientCertificatesFile” on page 389).
YMLSCRLF	25	File containing certificate revocation lists (CRLs) for X.509 certificates (see the user option “certificateRevocationListFile” on page 390).

Notes on defining the input section for the mail parameters

– If one of the parameters *YMLSKEYF*, *YMLSSGNF*, *YMLSASCF* or *YMLSCRLF* is specified, the corresponding option in the user option file is ignored. This option is deactivated by specifying the value *NONE. This is the case, for example, if an X.509 certificate other than the one specified in the option file is used to sign e-mails with S/MIME (*YMLSSGNF*), and for which, in contrast to the X.509 certificate in the option file, no additional X.509 certificates are required (YMLSASCF:*NONE).

– If the *YMLSRCTF* parameter is specified, the file specified there is placed above the files specified in *recipientCertificatesFile* options in the search sequence.

– The implementation of the ASTI subsystem limits the size of the mail parameters. An exact value cannot be specified here, but a value of around 32 KB is about right, minus the bytes reserved for additional parameters and internal administration data. This limit is only likely to be reached if the text of the e-mail or data in the attachments is inserted directly in the e-mail’s parameter section. If this error occurs, write the e-mail text in a (temporary) file, and specify only the file name in the parameter section.

– The data of a variable length begins immediately after the associated header. The headers must be designed for full word length. You therefore have to insert filler bytes when the variable part contains a number of bytes that is not a multiple of 4.

– The field name of the mail header must be entered in the data structure for additional headers without the subsequent colon.

– Unless otherwise indicated, the order in which the data structures occur is of no significance.

– The definition of the mail text or an attachment must be bracketed with two *YMLSATT* data structures in which *YMLSMBEG*/*YMLSMEND* (e-mail) or *YMLSABEG*/*YMLSAEND* is assigned to the tag field.

  The definition of the e-mail text or an attachment can contain data structures with the following tags:

  YMLSDSPC, YMLSCHST, YMLSTRCD, YMLSTCTD

If a *YMLSCSET* data structure (with a YMLSCHST tag) is outside a *YMLSATT* data structure, it is applied to the following header lines (see RFC 2047) until the occurrence of a further YMLSCSET data structure.

– All the specified files must remain unchanged until the e-mail is sent.

Listing of the expansion of the data structures for the mail parameters

Expansion:

XPAND= GENERAL, ADD_HEADER, DATA_SPEC, CHARSET, ENCODING, CONT_DISP, MSG_ATT

```
                              YMLSML MF=D,XPAND=GENERAL
                    1          MFTST MF=D,PREFIX=Y,MACID=MLS,ALIGN=F,
                    1                DMACID=MLS,SUPPORT=(E,D,C,M,L),DNAME=MLSGNRL
000000              2 YMLSGNRL DSECT ,
                    1 *   Input parameters
000000              1 YMLSGTAG DS    H                        tag of general parameter
                    1 *   Tag value(s) for struct _general
        00000001    1 YMLSFROM EQU   1                        from
        00000002    1 YMLSFFRM EQU   2                        fake_from
        00000003    1 YMLSTO   EQU   3                        to
        00000004    1 YMLSFTO  EQU   4                        fake_to
        00000005    1 YMLSCC   EQU   5                        cc
        00000006    1 YMLSFCC  EQU   6                        fake_cc
        00000007    1 YMLSBCC  EQU   7                        envelope recipient mail
                    1 *                                      addresses
        00000008    1 YMLSRPLT EQU   8                        reply_to
        00000009    1 YMLSSBJT EQU   9                        subject
        00000012    1 YMLSCTTT EQU   18                       content_type
        00000014    1 YMLSOPFI EQU   20                       option file
        00000015    1 YMLSKEYF EQU   21                       key file
        00000016    1 YMLSSGNF EQU   22                       signer certificate file
        00000017    1 YMLSASCF EQU   23                       additional signer
                    1 *                                      certificates file
        00000018    1 YMLSRCTF EQU   24                       recipient certificate
                                                             file
        00000019    1 YMLSCRLF EQU   25                       CRL file
                    1 *
000002              1 YMLSGRS1 DS    XL2                      reserved
000004              1 YMLSGLEN DS    F                        general parameter length
        00000008    1 YMLSGNRL# EQU   *-YMLSGTAG


                              YMLSML MF=D,XPAND=ADD_HEADER
                    1          MFTST MF=D,PREFIX=Y,MACID=MLS,ALIGN=F,
                    1                DMACID=MLS,SUPPORT=(E,D,C,M,L),DNAME=MLSHEAD
000000              2 YMLSHEAD DSECT ,
                    1 *   Input parameters
000000              1 YMLSHTAG DS    H                        tag of add. header
                    1 *   Tag value(s) for struct _add_header
        0000000A    1 YMLSAHDR EQU   10                       add. header
                    1 *
000002              1 YMLSHRS1 DS    XL2                      reserved
000004              1 YMLSHLNN DS    F                        add. header name length
```

```
000008            1 YMLSHLNB DS    F                        add. header body length
        0000000C  1 YMLSHEAD# EQU   *-YMLSHTAG


                            YMLSML MF=D,XPAND=DATA_SPEC
                  1         MFTST MF=D,PREFIX=Y,MACID=MLS,ALIGN=F,
                  1               DMACID=MLS,SUPPORT=(E,D,C,M,L),DNAME=MLSDTSP
000000            2 YMLSDTSP DSECT ,
                  1 *   Data specification
000000            1 YMLSDTAG DS    H                        tag of add. header
                  1 *   Tag value(s) for struct _data_spec
        0000000F  1 YMLSDSPC EQU   15                       dataspec
                  1 *
000002            1 YMLSDTYP DS    FL1                      reserved
                  1 *   Type of data specification
        00000001  1 YMLSFILE EQU   1                        file
        00000002  1 YMLSDATA EQU   2                        data
                  1 *
000003            1 YMLSDRS1 DS    XL1                      reserved
000004            1 YMLSDLEN DS    F                        length of data. spec.
        00000008  1 YMLSDTSP# EQU   *-YMLSDTAG


                            YMLSML MF=D,XPAND=CHARSET
                  1         MFTST MF=D,PREFIX=Y,MACID=MLS,ALIGN=F,
                  1               DMACID=MLS,SUPPORT=(E,D,C,M,L),DNAME=MLSCSET
000000            2 YMLSCSET DSECT ,
                  1 *   Data specification
000000            1 YMLSCTAG DS    H                        tag of add. header
                  1 *   Tag value(s) for struct _charset
        00000010  1 YMLSCHST EQU   16                       character set
                  1 *
000002            1 YMLSCBNR DS    FL1                      binary data
                  1 *   Binary data selection
        00000001  1 YMLSBYES EQU   1                        YES
        00000002  1 YMLSBNO  EQU   2                        NO
                  1 *
000003            1 YMLSCRS1 DS    XL1                      reserved
000004            1 YMLSCLEN DS    F                        length
000008            1 YMLSCSCX DS    CL8                      src_charset_XHCS
000010            1 YMLSCDCX DS    CL8                      dest_charset_XHCS
        00000018  1 YMLSCSET# EQU   *-YMLSCTAG


                            YMLSML MF=D,XPAND=ENCODING
                  1         MFTST MF=D,PREFIX=Y,MACID=MLS,ALIGN=F,
                  1               DMACID=MLS,SUPPORT=(E,D,C,M,L),DNAME=MLSENCD
000000            2 YMLSENCD DSECT ,
                  1 *   Encoding
000000            1 YMLSETAG DS    H                        tag of encoding
                  1 *   Tag value(s) for struct _encoding
        00000011  1 YMLSTRCD EQU   17                       transfer encoding
```

```
                    1 *
000002              1 YMLSEMCH DS    FL1                        encoding mechanism
                    1 *   Type of encoding mechanism
        00000001    1 YMLSE7BT EQU   1                          7 bit
        00000002    1 YMLSE8BT EQU   2                          8 bit
        00000003    1 YMLSEBIN EQU   3                          binary
        00000004    1 YMLSEQP  EQU   4                          quoted printable
        00000005    1 YMLSEB64 EQU   5                          base64
                    1 *
000003              1 YMLSERS1 DS    XL1                        reserved
        00000004    1 YMLSENCD# EQU   *-YMLSETAG


                              YMLSML MF=D,XPAND=CONT_DISP
                    1         MFTST MF=D,PREFIX=Y,MACID=MLS,ALIGN=F,
                    1               DMACID=MLS,SUPPORT=(E,D,C,M,L),DNAME=MLSDISP
000000              2 YMLSDISP DSECT ,
                    1 *   Content disposition
000000              1 YMLSITAG DS    H                          tag of content
                                                               disposition
                    1 *   Tag value(s) for struct _cont_disp
        00000013    1 YMLSTCTD EQU   19                         content disposition
                    1 *
000002              1 YMLSISPT DS    FL1                        content disposition
                    1 *   Disposition type
        00000001    1 YMLSINLN EQU   1                          inline
        00000002    1 YMLSIATT EQU   2                          attachment
                    1 *
000003              1 YMLSIRS1 DS    XL1                        reserved
        00000004    1 YMLSDISP# EQU   *-YMLSITAG


                              YMLSML MF=D,XPAND=MSG_ATT
                    1         MFTST MF=D,PREFIX=Y,MACID=MLS,ALIGN=F,
                    1               DMACID=MLS,SUPPORT=(E,D,C,M,L),DNAME=MLSATT
000000              2 YMLSATT  DSECT ,
                    1 *   Bracketing message or attachment specification
000000              1 YMLSATAG DS    H                          tag of msg | att
                    1 *   Tag value(s) for struct _msg_att
        0000000B    1 YMLSMBEG EQU   11                         msg_begin
        0000000C    1 YMLSMEND EQU   12                         msg_end
        0000000D    1 YMLSABEG EQU   13                         att_begin
        0000000E    1 YMLSAEND EQU   14                         att_end
                    1 *
000002              1 YMLSARS1 DS    XL2                        reserved
        00000004    1 YMLSATT# EQU   *-YMLSATAG
```

YMLCML - Get mail result

You can use this macro to check whether a mail order is completed. If the order has been carried out, the execution status is sent to the caller and removed from the ASTI queues.

Entry names or SVC number(s)

SVC 20 (decimal)

UNIT=940, FUNCTION=21, VERSION=1 | 2

Macro call format and operand description

YMLCML
FL= ***TU** / ***TPR** ,**VERSION**=**1** / **2** ,**ORDER**= ***ANY** / ***SINGLE** ,**ORDERID**= <var: char:16> ,**WAIT**= ***NO** / ***YES** ,**WAITTIM=*UNLIM** / <integer: 1..65535> /<var: int:4> ,**OPTFILE=*NONE** / <var: char:54>

FL=
Function area

***TU**
An SVC interface is generated.

***TPR**
A CALL interface is generated.

VERSION=
Selects the interface version.

1
Selects the old interface version.

2
Selects the new interface version, which offers the operands WAITTIM and OPTFILE and additional return codes.

ORDER=
Specifies the mail send orders for which the result is to be requested.

***ANY**
The execution of a completed order sent by the macro caller is queried. If there are several completed orders, it is not specified which of these orders is to have its status queried.

***SINGLE**
The result of the mail send order identified by the ORDERID parameter is queried.

ORDERID=
If ORDER=*SINGLE is specified, this parameter specifies the ID of the mail send order to be checked.

IDENTIFIER
Variable in which the order ID is stored or register containing the address of a variable in which the order ID is stored.

WAIT=
Specifies whether the caller wants to wait until processing of the mail send order by the mail sender backend is completed.

***NO**
Do not wait.

***YES**
Wait until the mail sender backend reports that processing of the send order is completed or that there was an error sending the e-mail.

WAITTIME=
With this operand the wait time can be limited in the case of WAIT=*YES. The operand is only available with VERSION=2. When the maximal wait time is reached, the call is terminated with an appropriate return code.

***UNLIM**
Unlimited wait time. With this operand value the behaviour is as hitherto.

INTEGER 1,65535
Maximal wait time in seconds.

IDENTIFIER
Variable or register containing the wait time (in seconds).

OPTFILE=
With this operand a user option file can be specified. The operand is available only with VERSION=2. The user option file can influence the macro behaviour by settings regarding the user specific logging (see logItems option).

***NONE**
No file defined.

IDENTIFIER
Variable containing the name of the user option file or register containing the address of the name of the user option file.

Return codes

SRC2	SRC1	MRC	MRC name	Meaning
00	00	0000	YMLCSUCC	E-mail concluded without an error.
00	01	0001	YMLCPARE	Parameter error.
00	20	0002	YMLCINTE	Internal error.
00	40	0003	YMLCONTF	Order not found.
00	40	0004	YMLCFTSK	Order sent by external task.
00	40	0005	YMLCANAV	ASTI subsystem not available.
00	40	0006	YMLCNORR	No result requested.
00	40	0007	YMLCONCM	Order not concluded.
00	40	0008	YMLCASTI	Unexpected ASTI error.
00	40	0009	YMLCTIME	Maximal wait time reached.
00	80	000A	YMLCOFNA	Error at access to user option file.
00	80	000B	YMLCRSRC	Resource saturation.
00	40	000C	YMLCSNAV	Service MAILCLNT not available.

SRC1/2 = sub-return code 1/2 in hexadecimal notation
MRC = main return code in hexadecimal notation

Macro call parameters

The data structure for the macro call parameters of YMLCML is composed as follows:

VERSION=1:

Distance	Identifier	Value	Meaning
	YMLCPARL		Parameter area.
000	YMLCHDR		Function header.
008	YMLCIND		Input parameters.
008	YMLCOIDI		ID of the mail send order in the case of YMLCOSNG.
018	YMLCORDS		Which mail send order's result is to be queried?
	YMLCOANY	1	Any of the caller's orders.
	YMLCOSNG	2	The order with the ID YMLCOIDI.
019	YMLCWAIT		Wait for the completion of the mail send order?
	YMLCWYES	1	Yes
	YMLCWNO	2	No
01A	YMLCRSV1		Reserved section.
01C	YMLCOUTD		Output parameters.
01C	YMLCOOID		Order ID of selected order in the case of YMLCOANY.
02C	YMLCRETC		Return code from backend
	YMLCBOK	00	Ok
	YMLCBPER	01	Parameter error
	YMLCBRSC	02	Resource saturation
	YMLCBSMT	03	SMTP error
	YMLCBSMI	04	S/MIME error
	YMLCBINT	0A	Internal error
030	YMLCRETM		May contain error messages in text format (e.g. SMTP error messages of the SMTP server).
0D0	YMLCARET		If the YMLCML call returns the return code YMLCASTI, this field contains the ASTI return code.

VERSION=2:

Distance	Identifier	Value	Meaning
	YMLCPARL		Parameter area
000	YMLCHDR		Function header
008	YMLCIND		Input parameters
008	YMLCOIDI		ID of mail send order in case YMLCOSNG.
018	YMLCWTTM		Maximal wait time
01C	YMLCORDS		Which mail send order's result is to be queried?
	YMLCOANY	1	Any of the caller's orders.
	YMLCOSNG	2	The order with the ID YMLCOIDI.
01D	YMLCWAIT		Wait for the completion of the mail send order?
	YMLCWYES	1	Yes
	YMLCWNO	2	No
01E	YMLCOPTF		User option file
054	YMLCOUTD		Output parameters
054	YMLCOID		Order ID of selected order in the case of YMLCOANY.
064	YMLCRETC		Return code from backend
	YMLCBOK	00	Ok
	YMLCBPER	01	Parameter error
	YMLCBRSC	02	Resource saturation
	YMLCBSMT	03	General SMTP error
	YMLCBSMI	04	S/MIME error
	YMLCBSMM	05	Error with SMTP MAIL command
	YMLCBSMR	06	Error with SMTP RCPT command
	YMLCBSMD	07	Error with SMTP DATA command
	YMLCBFAC	08	Error at access to user option file
	YMLCBMTL	09	Mail too large
	YMLCBINT	0A	Internal error
068	YMLCRETM		May contain error messages in text format (e.g. SMTP error messages of the SMTP server).
108	YMLCARET		If the YMLCML call returns the return code YMLCASTI, this field contains the ASTI return code.
10C	YMLCMID		Message ID
113	YMLCRSV1		Reserved area

Supplementary notes to the output parameters:

YMLCRETC

Some failure situations, where with VERSION=1 only a collective return code is delivered, are covered by specific return codes with VERSION=2 for finding out the real error reason more quickly. For instance the return code YMLCBMM is a strong hint that there is an error in the sender address. Accordingly the return code YMLCBMR indicates an error in at least one of the recipient addresses. A YMLCBMD can e.g. occur, when the SMTP server checks the violation of certain rules only after the DATA command, the reason for the rule violation can be related to the specified addresses. In any case the field YMLCRETM should give further (text) information related to the error cause.

YMLCRETM

If the SMTP server used supports RFC 2034, then this field contains a machine interpretable error indicator, which in general is more specific than the SMTP related return codes in the YMLCRETC field.

YMLCMID

This field contains a YML message key for a message describing the occured error. This is normally the same message as a corresponding REQUEST-MAIL-ORDER-RESULT command call would deliver.

Listing of the expansion of the data structure for the macro call parameters

Expansion:

XPAND= PARAM

```
                        YMLCML MF=D
                   1          MFTST MF=D,PREFIX=Y,MACID=MLC,ALIGN=F,
                   1                DMACID=MLC,SUPPORT=(E,D,C,M,L),DNAME=MLCPARL
000000             2 YMLCPARL DSECT ,
                   1 *   rc
         00000000  1 YMLCBOK  EQU   0                       OK
         00000001  1 YMLCBPER EQU   1                       Parameter error
         00000002  1 YMLCBRSC EQU   2                       Resource saturation
         00000003  1 YMLCBSMT EQU   3                       SMTP error
         00000004  1 YMLCBSMI EQU   4                       SMIME error
         0000000A  1 YMLCBINT EQU   10                      Internal error
                   1 *
                   1 *   Parameter area
                   1 YMLCHDR  FHDR  MF=(C,YMLC),EQUATES=NO                              FHDR
                   1 *   main return codes
         00000000  1 YMLCSUCC EQU   0                       No error detected
         00000001  1 YMLCPARE EQU   1                       Parameter error
         00000002  1 YMLCINTE EQU   2                       Internal error
         00000003  1 YMLCONTF EQU   3                       Order not found
         00000004  1 YMLCFTSK EQU   4                       Order issued by foreign task
         00000005  1 YMLCANAV EQU   5                       Subsystem ASTI not available
         00000006  1 YMLCNORR EQU   6                       No result requested
         00000007  1 YMLCONCM EQU   7                       Order not completed
         00000008  1 YMLCASTI EQU   8                       Unexpected ASTI error
                   1 *
                   1 *
000008             1 YMLCIND  DS    0XL20                   Input parameters
000008             1 YMLCOIDI DS    CL16                    Order Id
000018             1 YMLCORDS DS    FL1                     Order specification
                   1 *   order
         00000001  1 YMLCOANY EQU   1                       ANY
         00000002  1 YMLCOSNG EQU   2                       SINGLE
                   1 *
000019             1 YMLCWAIT DS    FL1                     Wait
                   1 *   wait
         00000001  1 YMLCWYES EQU   1                       YES
         00000002  1 YMLCWNO  EQU   2                       NO
                   1 *
00001A             1 YMLCRSV1 DS    CL2                     Reserved
                   1 *
                   1 *
00001C             1 YMLCOUTD DS    0XL168                  Output parameters
00001C             1 YMLCRETC DS    F                       Return code
000020             1 YMLCRETM DS    CL160                   Return message
0000C0             1 YMLCARET DS    F                       Return code from ASTI
                   1 *
         000000C4  1 YMLC#    EQU   *-YMLCHDR
```

YMLDML - Delete mail

You can use this macro to delete e-mails that have not yet been sent by the mail sender backend.

Entry names or SVC number(s)

SVC 20 (decimal)

UNIT=940, FUNCTION=22, VERSION=1 | 2

Macro call format and operand description

YMLDML
FL= ***TU** / ***TPR** ,**VERSION**=**1** / **2** ,**ORDER**= ***ALL** / ***SINGLE** ,**ORDERID**= <var: char:16> ,**OWNER**= ***OWN** / ***ALL** / ***OTHER** ,**USERID**= <var: char:8> ,**OPTFILE=*NONE** / <var: char:54>

FL=
Function area

***TU**
An SVC interface is generated.

***TPR**
A CALL interface is generated.

VERSION=
Selects the interface version.

1
Selects the old interface version.

2
Selects the new interface version, which offers the operand OPTFILE and an additional return code.

ORDER=
Specifies which mail order is to be deleted.

***ALL**
All the user's e-mails that have not yet been sent are deleted.

***SINGLE**
The e-mail specified in the ORDERID parameter is deleted.

ORDERID=
If ORDER=*SINGLE is specified, this parameter determines the ID of the mail order to be deleted.

IDENTIFIER
Variable in which the order ID is stored or register containing the address of a variable in which the order ID is stored.

OWNER=
Specifies the user whose mail orders are deleted.

***OWN**
Only delete the caller's mail orders.

***ALL**
Delete the mail orders of all users. Callers without TSOS authorization get the same result here as with *OWN.

***OTHER**
Delete the mail orders of the user specified in the USERID parameter (only permitted for callers with TSOS authorization).

USERID
Specifies the user ID whose mail orders are deleted.

IDENTIFIER
Variable in which the user ID is stored or register containing the address of a variable in which the user ID is stored.

OPTFILE=
With this operand a user option file can be specified. The operand is available only with VERSION=2. Currently there is no user option which would influence the behaviour of YMLDML, but because this may change in the future it is recommended to supply by now the OPTFILE operand of the YDLDML call with the same value as done with the YMLSML and YMLCML calls.

***NONE**
No file defined.

IDENTIFIER
Variable containing the name of the user option file or register containing the address of the name of the user option file.

Return codes

SRC2	SRC1	MRC	MRC name	Meaning
00	00	0000	YMLDSUCC	No error found.
00	01	0001	YMLDPARE	Parameter error.
00	20	0002	YMLDINTE	Internal error.
00	40	0003	YMLDONTF	Order not found.
00	40	0004	YMLDIPRV	Insufficient authorization.
00	40	0005	YMLDWROW	Order does not belong to the specified owner.
00	40	0006	YMLDFTSK	Order sent by external task.
00	40	0007	YMLDANAV	ASTI not available.
00	20	0008	YMLDASTI	Unexpected ASTI error.
00	40	0009	YMLDSNAV	Service MAILCLNT not available.

SRC1/2 = sub-return code 1/2 in hexadecimal notation
MRC = main return code in hexadecimal notation

Macro call parameters

The data structure for the macro call parameters of YMLDML is composed as follows:

VERSION=1:

Distance	Identifier	Value	Meaning
	YMLDPARL		Parameter area.
000	YMLDHDR		Function header.
008	YMLDIND		Input parameters.
008	YMLDOIDI		ID of the order to be deleted if YMLDORDS=YMLDOSNG is specified.
018	YMLDORDS		Specifies which orders are to be deleted.
	YMLDOALL	1	All orders that have not yet been sent are deleted.
	YMLDOSNG	2	The order specified under YMLDOIDI is deleted.
019	YMLDOWNS		Specifies the user whose orders are to be deleted.
	YMLDWOWN	1	Only mail orders of the caller are deleted.
	YMLDWALL	2	All orders that have not yet been sent are deleted.
	YMLDWOTH	3	Orders of the user specified under YMLUSID are deleted.
01A	YMLDRSV1		Reserved area.
01C	YMLDUSID		User ID whose orders are to be deleted.
024	YMLDOUTD		Output parameters.
024	YMLDARET		If the YMLDML call returns the return code YMLDASTI, this field contains the ASTI return code.

VERSION=2:

Distance	Identifier	Value	Meaning
	YMLDPARL		Parameter area
000	YMLDHDR		Function header
008	YMLDIND		Input parameters
008	YMLDOIDI		ID of the order to be deleted if YMLDORDS=YMLDOSNG
018	YMLDORDS		Specifies which orders are to be deleted.
	YMLDOALL	1	All orders that have not yet been sent are deleted.
	YMLDOSNG	2	The order specified under YMLDOIDI is deleted.
019	YMLDOWNS		Specifies the user whose orders are to be deleted.
	YMLDWOWN	1	Only mail send orders of the caller are deleted.
	YMLDWALL	2	All orders that have not yet been sent are deleted.

Distance	Identifier	Value	Meaning
	YMLDWOTH	3	Orders of the user specified under YMLDUSID are deleted.
01A	YMLDUSID		User id whose orders are to be deleted.
022	YMLDOPTF		User option file
058	YMLDOUTD		Output parameters
058	YMLDARET		If the YMLDML call returns the return code YMLDASTI, this field contains the ASTI return code.
05C	YMLDMID		Message ID
063	YMLDRSV1		Reserved area

Supplementary notes to the output parameters:

YMLDMID

This field contains a YML message key for a message describing the occured error. This is normally the same message as a corresponding DELETE-MAIL-ORDER command call would deliver.

Listing of the expansion of the data structure for the macro call parameters

Expansion:

XPAND= PARAM

```
                                        YMLSML MF=D,XPAND=PARA
                                 YMLDML MF=D
                           1     MFTST MF=D,PREFIX=Y,MACID=MLD,ALIGN=F,
                           1           DMACID=MLD,SUPPORT=(E,D,C,M,L),DNAME=MLDPARL
000000                     2 YMLDPARL DSECT ,
                           1 *   Parameter area
                           1 YMLDHDR  FHDR  MF=(C,YMLD),EQUATES=NO    FHDR
                           1 *   main return codes
          00000000         1 YMLDSUCC EQU   0                         No error detected
          00000001         1 YMLDPARE EQU   1                         Parameter error
          00000002         1 YMLDINTE EQU   2                         Internal error
          00000003         1 YMLDONTF EQU   3                         Order not found
          00000004         1 YMLDIPRV EQU   4                         Insufficient privileges
          00000005         1 YMLDWROW EQU   5                         Order not owned by specified
                           1 *                                        owner
          00000006         1 YMLDFTSK EQU   6                         Order issued by foreign task
          00000007         1 YMLDANAV EQU   7                         Subsystem ASTI not available
          00000008         1 YMLDASTI EQU   8                         Unexpected ASTI error
                           1 *
                           1 *
000008                     1 YMLDIND  DS    0XL28                     Input parameters
000008                     1 YMLDOIDI DS    CL16                      Order Id
000018                     1 YMLDORDS DS    FL1                       Order specification
                           1 *   order
          00000001         1 YMLDOALL EQU   1                         ALL
          00000002         1 YMLDOSNG EQU   2                         SINGLE
                           1 *
000019                     1 YMLDOWNS DS    FL1                       Owner specification
                           1 *   owner
          00000001         1 YMLDWOWN EQU   1                         OWN
          00000002         1 YMLDWALL EQU   2                         ALL
          00000003         1 YMLDWOTH EQU   3                         OTHER
                           1 *
00001A                     1 YMLDRSV1 DS    CL2                       Reserved
00001C                     1 YMLDUSID DS    CL8                       User id of owner
                           1 *
                           1 *
000024                     1 YMLDOUTD DS    0XL4                      Output parameters
000024                     1 YMLDARET DS    F                         Return code from ASTI
                           1 *
          00000028         1 YMLD#    EQU   *-YMLDHDR
```

YMLGML - Query information about the mail

You can use this macro to query information about the mail orders.

Entry names or SVC number(s)

SVC 20 (decimal)

UNIT=940, FUNCTION=23, VERSION=1 | 2

Macro call format and operand description

YMLGML
FL= ***TU** / ***TPR** ,**VERSION**=**1** / **2** ,**XPAND**=**PARAM** / **OUTPAR** ,**ORDER**=***SUM** / ***ALL** ,**ORDERID**= <var: char:16> ,**OWNER**= ***OWN** / ***ALL** / ***OTHER** ,**USERID**= <var: char:8> ,**OUTPAR**= <var: pointer> ,**OUTPARL**=<integer: 1..32767> / <var: int:4> ,**OPTFILE=*NONE** / <var: char:54>

FL=
Function area

***TU**
An SVC interface is generated.

***TPR**
A CALL interface is generated.

VERSION=
Selects the interface version.

1
Selects the old interface version.

2
Selects the new interface version, which offers the operand OPTFILE and an additional return code.

XPAND=
This parameter controls the expansion of the data structures that describe the parameter list of the macro and the layout of the output section.

PARAM
Creates parameter list.

OUTPAR
Create data layout for the output section.

ORDER=
Specifies which information about the e-mails is output in the queue.

***SUM**
Queries the number of e-mails in the user queue.

***ALL**
Queries the order ID of all the e-mails in the user queue.

***SINGLE**
Queries the parameters of the e-mail specified in the ORDERID parameter.

It is advisable to call the macro first with ORDER=*ALL to get all the e-mails of the calling user. You can call the macro separately for each order ID with ORDER=*SINGLE to get the data of the mail parameters.

The second call may fail with YMLGONTF if, for example, another task has issued a YMCML/YMCMLC or YMLDML/YMLDMCL call between the two calls.

If the output section is too small for the parameter data of the e-mail, the call is terminated with YMLGOSML. In this case, repeat the call with a larger output section.

ORDERID=
If ORDER=*SINGLE is specified, this parameter specifies the ID of the mail order for which information is requested.

IDENTIFIER
Variable in which the order ID is stored or register containing the address of a variable in which the order ID is stored.

OWNER=
Specifies the user about whose mail orders information is to be queried.

***OWN**
Only queries mail orders of the calling user.

***ALL**
Queries mail orders of all users. Callers without TSOS authorization get the same result here as with *OWN.

***OTHER**
Queries mail orders of the user specified by the USERID parameter (only permissible for callers with TSOS authorization).

USERID
If OWNER=*OTHER is specified, this parameter specifies the user ID of the owner of the mail orders.

IDENTIFIER
Variable in which the user ID is stored or register containing the address of a variable in which the user ID is stored.

OUTPAR=
Specifies the output section. This parameter is required when ORDER=*ALL or ORDER=*SINGLE is specified.
ORDER=*ALL:
The output section contains a list of the IDs of the mail send orders.
ORDER=*SINGLE:
The output section contains the mail parameter section of the order ID specified in the ORDERID parameter.

IDENTIFIER
Variable in which the address of the output section is stored or register containing the address of a variable in which the address of the output section is stored.

OUTPARL=
Length of the output section.

INTEGER (1,32767)
Length of the output section.

IDENTIFIER
Variable in which the length of the output section is stored or register containing the length of the output section.

OPTFILE=
With this operand a user option file can be specified. The operand is available only with VERSION=2. Currently there is no user option which would influence the behaviour of YMLGML, but because this may change in the future it is recommended to supply by now the OPTFILE operand of the YDLGML call with the same value as done with the YMLSML and YMLCML calls.

***NONE**
No file defined.

IDENTIFIER
Variable containing the name of the user option file or register containing the address of the name of the user option file.

Return codes

SRC2	SRC1	MRC	MRC name	Meaning
00	00	0000	YMLGSUCC	No error found.
00	01	0001	YMLGPARE	Parameter error.
00	20	0002	YMLGINTE	Internal error.
00	40	0003	YMLGONTF	Order not found.
00	40	0004	YMLGOSML	Output section too small.
00	40	0005	YMLGONTO	Order does not belong to the caller.
00	40	0006	YMLGRSRC	Resources are exhausted.
00	40	0007	YMLGFTSK	Order executed by external task.
00	40	0008	YMLGANAV	ASTI subsystem not available.
00	40	0009	YMLGANAV	No order for sending e-mails.
00	20	000A	YMLGASTI	Unexpected ASTI error.
00	40	000B	YMLGSNAV	Service MAILCLNT not available.

SRC1/2 = sub-return code 1/2 in hexadecimal notation; MRC = main return code in hexadecimal notation

Macro call parameters and output section

The YMLGML macro call uses the following data structures:

– Data structure for input and output parameters of the macro call YMLGML (XPAND=PARAM)
– Output data structure for the information supplied by YMLGML (XPAND=OUTPAR)

The data structure for the macro call parameters of YMLGML is composed as follows:

VERSION=1:

Distance	Identifier	Value	Meaning
	YMLGPARL		Parameter area.
000	YMLGHDR		Function header.
008	YMLGIND		Input parameters.
008	YMLGOIDI		ID of the order about which information is to be queried if YMLGORDS=YMLGOSNG is specified.
018	YMLGORDS		Specifies which information is queried.
	YMLGOSUM	1	Queries the number of e-mails in the user queue.
	YMLGOALL	2	Queries the order ID of all the e-mails in the user queue.
	YMLGOSNG	3	Queries parameters of the order specified in YMLGOIDI.
019	YMLGOWNS		Specifies the user about whose orders information is to be queried.
	YMLGWOWN	1	Only orders of the calling user are queried.
	YMLGWALL	2	Orders of all users are queried.
	YMLGWOTH	3	Orders of the user specified in YMLGUSID are queried.
01A	YMLGRSV1		Reserved area.
01C	YMLGUSID		User ID whose orders are to be deleted.
024	YMLGOUT		Address of the output area.
028	YMLGOUTL		Size of the output area.
02C	YMLGOUTD		Output parameters.
02C	YMLGSUM		Number of e-mails in the queue.
030	YMLGARET		If the YMLGML call returns the return code YMLGASTI, this field contains the ASTI return code.

VERSION=2:

Distanz	Identifier	Wert	Bedeutung
	YMLGPARL		Parameter area
000	YMLGHDR		Function header
008	YMLGIND		Input parameters
008	YMLGOIDI		ID of the order about whichinformation is to be queried if YMLGORDS=YMLGOSNG is specified.
018	YMLGORDS		Specifies which information is queried.
	YMLGOSUM	1	Queries the number of mails in the user wait queue.
	YMLGOALL	2	Queries the order ID of all the mails in the user wait queue.
	YMLGOSNG	3	Queries parameters of the order specified in YMLGOIDI.
019	YMLGOWNS		Specifies the user about whose orders information is to be queried.
	YMLGWOWN	1	Only orders of the calling user are queried.
	YMLGWALL	2	Orders of all users are queried.
	YMLGWOTH	3	Orders of the user specified in YMLGUSID are queried.
01A	YMLGRSV1		Reserved area
01C	YMLGUSID		User ID whose orders are to be deleted.
024	YMLGOUT		Adress of output area
028	YMLGOUTL		Size of output area
02C	YMLGOPTF		User option file
062	YMLGRSV2		Reserved area
064	YMLGOUTD		Output parameters
064	YMLGSUM		Number of mails in the wait queue.
068	YMLGARET		If the YMLGML call returns the return code YMLGASTI, this field contains the ASTI return code.
06C	YMLGMID		Message ID
073	YMLGRSV3		Reserved area

Supplementary notes to the output parameters:

YMLGMID

This field contains a YML message key for a message describing the occured error. This is normally the same message as a corresponding SHOW-MAIL-ORDER-STATUS command call would deliver.

Listing of the expansion of the data structures for macro call parameters and the output section

Expansion:

XPAND= PARAM, OUTPAR

```
                         YMLGML MF=D,XPAND=PARAM
                  1      MFTST MF=D,PREFIX=Y,MACID=MLG,ALIGN=F,
                  1            DMACID=MLG,SUPPORT=(E,D,C,M,L),DNAME=MLGPARL
000000            2 YMLGPARL DSECT ,
                  1 *   Parameter area
                  1 YMLGHDR  FHDR  MF=(C,YMLG),EQUATES=NO                        FHDR
                  1 *   main return codes
          00000000  1 YMLGSUCC  EQU   0                     No error detected
          00000001  1 YMLGPARE  EQU   1                     Parameter error
          00000002  1 YMLGINTE  EQU   2                     Internal error
          00000003  1 YMLGONTF  EQU   3                     Order not found
          00000004  1 YMLGOSML  EQU   4                     Output area too small
          00000005  1 YMLGONTO  EQU   5                     Order not own
          00000006  1 YMLGRSRC  EQU   6                     Resource saturation
          00000007  1 YMLGFTSK  EQU   7                     Order issued by foreign
                                                            task
          00000008  1 YMLGANAV  EQU   8                     Subsystem ASTI not
                                                            available
          00000009  1 YMLGNMSO  EQU   9                     Not a mail send order
          0000000A  1 YMLGASTI  EQU   10                    Unexpected ASTI error
                  1 *
                  1 *
000008            1 YMLGIND   DS    OXL36                 Input parameters
000008            1 YMLGOIDI  DS    CL16                  Order Id
000018            1 YMLGORDS  DS    FL1                   Order specification
                  1 *   order
          00000001  1 YMLGOSUM  EQU   1                     sum
          00000002  1 YMLGOALL  EQU   2                     all
          00000003  1 YMLGOSNG  EQU   3                     single
                  1 *
000019            1 YMLGOWNS  DS    FL1                   Owner specification
                  1 *   owner
          00000001  1 YMLGWOWN  EQU   1                     OWN
          00000002  1 YMLGWALL  EQU   2                     ALL
          00000003  1 YMLGWOTH  EQU   3                     OTHER
                  1 *
00001A            1 YMLGRSV1  DS    CL2                   Reserved
00001C            1 YMLGUSID  DS    CL8                   User id of owner
000024            1 YMLGOUT   DS    A                     Output area
000028            1 YMLGOUTL  DS    F                     Output area length
                  1 *
                  1 *
00002C            1 YMLGOUTD  DS    OXL8                  Output parameters
00002C            1 YMLGSUM   DS    F                     Number of queued mails
```

```
000030             1 YMLGARET  DS    F                        Return code from ASTI
                   1 *
        00000034   1 YMLG#     EQU   *-YMLGHDR

                             YMLGML MF=D,XPAND=OUTPAR
                   1         MFTST MF=D,PREFIX=Y,MACID=MLG,ALIGN=F,
                   1               DMACID=MLG,SUPPORT=(E,D,C,M,L),DNAME=MLGOUTPAR
000000             2 YMLGOUTPAR DSECT ,
                   1 *   rc
        00000000   1 YMLGBOK   EQU   0                        OK
        00000001   1 YMLGBPER  EQU   1                        Parameter error
        00000002   1 YMLGBRSC  EQU   2                        Resource saturation
        00000003   1 YMLGBSMT  EQU   3                        SMTP error
        00000004   1 YMLGBSMI  EQU   4                        SMIME error
        0000000A   1 YMLGBINT  EQU   10                       Internal error
                   1 *
                   1 *   STRUCT Output parameters
000000             1 YMLGOUTP  DS    0XL192                   UNION Output parameters
                   1 *
000000             1 YMLGMDAT  DS    0XL192                   mail data
000000             1 YMLGSTAT  DS    F                        mail status
                   1 *   order status
        00000001   1 YMLGWAIT  EQU   1                        waiting
        00000002   1 YMLGDEFE  EQU   2                        deferred
        00000003   1 YMLGACTV  EQU   3                        active
        00000004   1 YMLGSENT  EQU   4                        send successful
        00000005   1 YMLGFAIL  EQU   5                        send failed
                   1 *
000004             1 YMLGTIME  DS    F                        submission time
000008             1 YMLGUSER  DS    CL8                      submitter
000010             1 YMLGSLCT  DS    F                        data selector
                   1 *   Data type selector
        00000001   1 YMLGSORD  EQU   1                        order data
        00000002   1 YMLGSDAT  EQU   2                        result data
                   1 *
000014             1 YMLGDATA  DS    0XL164                   UNION data
                   1 *
000014             1 YMLGRDAT  DS    0XL164                   result data
000014             1 YMLGRETC  DS    F                        return code
000018             1 YMLGRETM  DS    CL160                    return message
                   1 *
0000B8  00000014   1          ORG   YMLGDATA
                   1 *
000014             1 YMLGODAT  DS    0XL4                     order data
000014             1 YMLGCNTT  DS    F                        # sending tries
                   1 *
000018  000000B8   1          ORG   YMLGDATA+164
0000B8             1 YMLGMPLN  DS    F                        mail parameter length
0000BC             1 YMLGMP    DS    CL4                      mail parameters, real
                                                              size:
                   1 *                                        mail_par_len
                   1 *
```

```
0000C0  00000000   1          ORG   YMLGOUTP
                   1 *
000000             1 YMLGOIDS  DS    0XL20                  order Ids
000000             1 YMLGNORD  DS    F                      number of orders
000004             1 YMLGOID   DS    1CL16                  Array of order ids, real
                   1 *                                      array size: num_order
        00000001   1 YMLGOID#  EQU   1
                   1 *
000014  000000C0   1          ORG   YMLGOUTP+192
        000000C0   1 YMLGOUTPAR# EQU  *-YMLGSTAT
```

9.3.2 Function calls in C

The C subprogram interface of the mail sender frontend supports the following function calls:

Function call	Function	Associated include file with function declaration and data structures
YMLSML()	Send mail	YMLSML.H
YMLCML()	Get mail result	YMLCML.H
YMLDML()	Delete mail	YMLDML.H
YMLGML()	Get information about the mail	YMLGML.H

C function calls and include files used

The extension of the subprogram interface with some additional parameters and return-codes described in the section “ASSEMBLER macro interface” on page 422 affects also the C include files.

The selection between the old and the new interface version is done with the definition of a preprocessor symbol before the inclusion of the particular include file.

If one wants to use the VERSION=1 variant of the include files, the following has to be specified:

```
#define _YMLCML_H_VERSION_1
#include "YMLCML.H"

#define _YMLDML_H_VERSION_1
#include "YMLDML.H"

#define _YMLGML_H_VERSION_1
#include "YMLGML.H"

#define _YMLSML_H_VERSION_1
#include "YMLSML.H"
```

Correspondingly one gets the VERSION=2 variant of the include files with:

```
#define _YMLCML_H_VERSION_2
#include "YMLCML.H"

#define _YMLDML_H_VERSION_2
#include "YMLDML.H"

#define _YMLGML_H_VERSION_2
#include "YMLGML.H"

#define _YMLSML_H_VERSION_2
#include "YMLSML.H"
```

For provisioning of the function header with unit, function and version values corresponding preprocessor symbols are defined in the include files, for instance the symbols YMLSML_UNIT, YMLSML_FUNCTION and YMLSML_VERSION for the YMLSML interface. The latter symbol naturally depends on the selected interface version.

The C include files listed below declare the functions and the data structures used by them.

9.3.2.1 Include file YMLSML.H for YMLSML() - Send mail

The include file YMLSML.H is listed below. You will find a program example that shows how to use the YMLSML() function call on page 478.

```
#ifndef _YMLSML_H
#define _YMLSML_H

#if 0
/*****************************************************************
 BEGIN-INTERFACE   YMLSML

 TITLE             (/ Submit send mail order /)
 NAME              YMLSML.H
 DOMAIN            MAIL
 LANGUAGE          C
 COPYRIGHT         (C) Fujitsu Technology Solutions GmbH 2010
                       ALL RIGHTS RESERVED
 COMPILATION-SCOPE USER
 INTERFACE-TYPE    CALL,
                      ORDER
 RUN-CONTEXT       TU
 PURPOSE           (/ Submit an order to send mail
                   /)

 END-INTERFACE     YMLSML.
*****************************************************************/
#endif

/* Wait operand values                                          */
/* ENUM wait_s                                                  */
#define YMLSML_yes 1                  /* YES                    */
#define YMLSML_no 2                   /* NO-DISCARD             */
#define YMLSML_res 3                  /* NO                     */

/* Mail encryption/signing protocol                             */
/* ENUM sec_prot_s                                              */
#define YMLSML_smime 1                /* SMIME                  */

/* General yes or no selection                                  */
/* ENUM yesno_s                                                 */
#define YMLSML_yes 1                  /* YES                    */
```

```
#define YMLSML_no 2                     /* NO                          */
#define YMLSML_optfile 3                /* OPTFILE                     */

/* Cipher operand values                                              */
/* ENUM cipher_s                                                      */
#define YMLSML_rc2_40 1                 /* RC2-40                      */
#define YMLSML_rc2_64 2                 /* RC2-64                      */
#define YMLSML_rc2_128 3                /* RC2-128                     */
#define YMLSML_des 4                    /* DES                         */
#define YMLSML_des3 5                   /* 3DES                        */
#define YMLSML_aes_128 6                /* AES-128                     */
#define YMLSML_aes_192 7                /* AES-192                     */
#define YMLSML_aes_256 8                /* AES-256                     */
#define YMLSML_cipher_from_optfile 127  /* OPTFILE                     */

/* main return codes                                                  */
/* mret_code                                                          */
#define YMLSML_successful 0             /* No error detected           */
#define YMLSML_parameter_error 1        /* Parameter error             */
#define YMLSML_int_error 2              /* Internal error              */
#define YMLSML_syntax_error 3           /* Syntax error                */
#define YMLSML_opt_file_error 4         /* Option file not accessible  */
#define YMLSML_msgatt_file_error 5      /* Message or attachment file  */
                                        /* not accessible              */
#define YMLSML_smime_file_error 6       /* SMIME related file not      */
                                        /* accessible                  */
#define YMLSML_param_too_big 7          /* Mail parameter too big      */
#define YMLSML_backend_error 8          /* Back-end error              */
#define YMLSML_max_order 9              /* Max number of orders        */
                                        /* exceeded                    */
#define YMLSML_serv_not_avail 10        /* Mailclient Service not      */
                                        /* available                   */
#define YMLSML_addr_invalid 11          /* Mailpar address invalid     */
#define YMLSML_resource_sat 12          /* Resource saturation         */
#define YMLSML_asti_not_avail 13        /* Subsystem ASTI not          */
                                        /* available                   */
#define YMLSML_no_from_addr 14          /* No FROM address specified   */
#define YMLSML_asti_error 15            /* Unexpected ASTI error       */

/* Parameter area                                                     */
struct YMLSML_pl_mdl {

        /* FHDR                                                       */
        struct ESMFHDR hdr;

        /* Input parameters                                           */
        struct {
                void* mail_par;         /* Mail parameter area        */
                unsigned long mail_par_len;
                                        /* Mail parameter area length */
                unsigned char wait;     /* wait                       */
                unsigned char sec_prot; /* Protocol for encryption    */
```

```
                                        /* and signing                */
                unsigned char encrypt;  /* Encryption                 */
                unsigned char sign;     /* Signing                    */
                unsigned char cipher;   /* Cipher                     */
                char optfile[54];       /* Option file                */
                char reserved1[1];      /* Reserved                   */
        } in_data;

        /* Output parameters                                         */
        struct {
                char order_id[16];      /* Order id                   */
                char error_message[160];
                                        /* Error message              */
                unsigned long asti_rc;  /* Return code from ASTI      */
        } out_data;
};

/*  Entry for YMLSML  */
#ifdef __SNI_HOST_BS2000
#ifdef __cplusplus
extern "C" void _SVC(int, void*);
inline void YMLSML(struct YMLSML_pl_mdl& param)
{_SVC(20, &param);}
#else
void _SVC(int, void*);
#define YMLSML(p) _SVC(20, &p)
#endif
#endif

/* Tag value(s) for struct _general                                 */
/* ENUM tag_s                                                       */
#define YMLSML_tag_from 1               /* from                     */
#define YMLSML_tag_fake_from 2          /* fake_from                */
#define YMLSML_tag_to 3                 /* to                       */
#define YMLSML_tag_fake_to 4            /* fake_to                  */
#define YMLSML_tag_cc 5                 /* cc                       */
#define YMLSML_tag_fake_cc 6            /* fake_cc                  */
#define YMLSML_tag_bcc 7                /* envelope recipient mail  */
                                        /* addresses                */
#define YMLSML_tag_replyto 8            /* reply_to                 */
#define YMLSML_tag_subject 9            /* subject                  */
#define YMLSML_tag_cont_type 18         /* content_type             */
#define YMLSML_tag_optfile 20           /* option file              */
#define YMLSML_tag_keyfile 21           /* key file                 */
#define YMLSML_tag_signfile 22          /* signer certificate file  */
#define YMLSML_tag_scertfile 23         /* additional signer        */
                                        /* certificates file        */
#define YMLSML_tag_rcertfile 24         /* recipient certificate file */
#define YMLSML_tag_crlfile 25           /* CRL file                 */

/* Input parameters                                                 */
struct YMLSML_general {
```

```
        unsigned short tag;             /* tag of general parameter   */
        char reserved[2];               /* reserved                   */
        unsigned long len;              /* general parameter length   */
};

/* Tag value(s) for struct _add_header                                */
/* ENUM tag_s                                                         */
#define YMLSML_tag_add_header 10        /* add. header                */

/* Input parameters                                                   */
struct YMLSML_add_header {
        unsigned short tag;             /* tag of add. header         */
        char reserved[2];               /* reserved                   */
        unsigned long len_field_name;   /* add. header name length    */
        unsigned long len_field_body;   /* add. header body length    */
};

/* Tag value(s) for struct _data_spec                                 */
/* ENUM tag_s                                                         */
#define YMLSML_tag_data_spec 15         /* dataspec                   */

/* Type of data specification                                         */
/* ENUM type_s                                                        */
#define YMLSML_file 1                   /* file                       */
#define YMLSML_data 2                   /* data                       */

/* Data specification                                                 */
struct YMLSML_data_spec {
        unsigned short tag;             /* tag of add. header         */
        unsigned char type;             /* reserved                   */
        char reserved[1];               /* reserved                   */
        unsigned long len;              /* length of data. spec.      */
};

/* Tag value(s) for struct _charset                                   */
/* ENUM tag_s                                                         */
#define YMLSML_tag_charset 16           /* character set              */

/* Binary data selection                                              */
/* ENUM bnry_s                                                        */
#define YMLSML_yes 1                    /* YES                        */
#define YMLSML_no 2                     /* NO                         */
/* Data specification                                                 */
struct YMLSML_charset {
        unsigned short tag;             /* tag of add. header         */
        unsigned char binary;           /* binary data                */
        char reserved[1];               /* reserved                   */
        unsigned long len;              /* length                     */
        char src_charset_XHCS[8];       /* src_charset_XHCS           */
        char dest_charset_XHCS[8];      /* dest_charset_XHCS          */
};
```

```
/* Tag value(s) for struct _encoding                                  */
/* ENUM tag_s                                                         */
#define YMLSML_tag_encoding 17          /* transfer encoding          */

/* Type of encoding mechanism                                         */
/* ENUM mech_s                                                        */
#define YMLSML_7bit 1                   /* 7 bit                      */
#define YMLSML_8bit 2                   /* 8 bit                      */
#define YMLSML_binary 3                 /* binary                     */
#define YMLSML_qp 4                     /* quoted printable           */
#define YMLSML_b64 5                    /* base64                     */

/* Encoding                                                           */
struct YMLSML_encoding {
        unsigned short tag;             /* tag of encoding            */
        unsigned char mechanism;        /* encoding mechanism         */
        char reserved[1];               /* reserved                   */
};

/* Tag value(s) for struct _cont_disp                                 */
/* ENUM tag_s                                                         */
#define YMLSML_tag_cont_disp 19         /* content disposition        */

/* Disposition type                                                   */
/* ENUM disptype_s                                                    */
#define YMLSML_inline 1                 /* inline                     */
#define YMLSML_attachment 2             /* attachment                 */

/* Content disposition                                                */
struct YMLSML_cont_disp {
        unsigned short tag;             /* tag of content disposition */
        unsigned char disp_type;        /* content disposition        */
        char reserved[1];               /* reserved                   */
};

/* Tag value(s) for struct _msg_att                                   */
/* ENUM tag_s                                                         */
#define YMLSML_tag_msg_begin 11         /* msg_begin                  */
#define YMLSML_tag_msg_end 12           /* msg_end                    */
#define YMLSML_tag_att_begin 13         /* att_begin                  */
#define YMLSML_tag_att_end 14           /* att_end                    */

/* Bracketing message or attachment specification                     */
struct YMLSML_msg_att {
        unsigned short tag;             /* tag of msg | att           */
        char reserved[2];               /* reserved                   */
};

#endif          /* _YMLSML_H */
```

Notes on the definition of the input section for the mail parameters

– The implementation of the subsystem limits the size of the mail parameters. An exact value cannot be specified here, but a value of around 32 KB is about right, minus the bytes reserved for additional parameters and internal administration data. This limit is only likely to be reached if the text of the e-mail or data in the attachments is inserted directly in the e-mail's parameter section. If this error occurs, write the e-mail text in a (temporary) file, and specify only the file name in the parameter section.

– The data of a variable length begins immediately after the associated header. The headers must be designed for full word length. You therefore have to insert filler bytes when the variable part contains a number of bytes that is not a multiple of 4.

– The field name of the mail header must be entered in the data structure for additional headers without the subsequent colon.

– The order in which the data structures occur is irrelevant unless stated otherwise.

– The definition of the mail text or an attachment must be nested with two `YMLSML_msg_att` data structures, where

  ```
  tag = YMLSML_tag_msg_begin / YMLSML_tag_msg_end
  ```

  or

  ```
  tag = YMLSML_tag_att_begin / YMLSML_tag_att_end
  ```

  The definition of the mail text or an attachment can contain data structures with the following tags:

  ```
  YMLSML_tag_{data_spec, charset, encoding, cont_type}
  ```

  If a `YMLSML_charset` data structure is outside the `YMLSML_tag_msg_att` data structure, it is applied to the subsequent header lines (see RFC 2047) and to subsequent messages or attachments without their own character set specification.

– Files must remain unchanged until the e-mail is sent.

9.3.2.2 Include file YMLCML.H for YMLCML() - Get mail result

The include file YMLCML.H is listed below. You will find a program example that shows how to use the YMLCML() function call on page 478.

```
#ifndef _YMLCML_H
#define _YMLCML_H

#if 0
/*****************************************************************
 BEGIN-INTERFACE   YMLCML

 TITLE             (/ Check send mail order /)
 NAME              YMLCML.H
 DOMAIN            MAIL
 LANGUAGE          C
 COPYRIGHT         (C) Fujitsu Technology Solutions GmbH 2010
                       ALL RIGHTS RESERVED
 COMPILATION-SCOPE USER
 INTERFACE-TYPE    CALL
 RUN-CONTEXT       TU
 PURPOSE           (/ Check, whether a submitted send mail order has completed
                   /)

 END-INTERFACE     YMLCML.
*****************************************************************/
#endif

/* order                                                        */
/* ENUM order_s                                                 */
#define YMLCML_any 1                    /* ANY                  */
#define YMLCML_single 2                 /* SINGLE               */

/* wait                                                         */
/* ENUM wait_s                                                  */
#define YMLCML_yes 1                    /* YES                  */
#define YMLCML_no 2                     /* NO                   */

/* rc                                                           */
/* ENUM rc_s                                                    */
#define YMLCML_be_ok 0                  /* OK                   */
#define YMLCML_be_param_error 1         /* Parameter error      */
#define YMLCML_be_resource_sat 2        /* Resource saturation  */
#define YMLCML_be_smtp_error 3          /* SMTP error           */
#define YMLCML_be_smime_error 4         /* SMIME error          */
#define YMLCML_be_int_error 10          /* Internal error       */

/* main return codes                                            */
/* mret_code                                                    */
#define YMLCML_successful 0             /* No error detected    */
#define YMLCML_parameter_error 1        /* Parameter error      */
```

```
#define YMLCML_int_error 2              /* Internal error             */
#define YMLCML_order_not_found 3        /* Order not found            */
#define YMLCML_foreign_task 4           /* Order issued by foreign    */
                                        /* task                       */
#define YMLCML_asti_not_avail 5         /* Subsystem ASTI not         */
                                        /* available                  */
#define YMLCML_no_result_req 6          /* No result requested        */
#define YMLCML_order_not_compl 7        /* Order not completed        */
#define YMLCML_asti_error 8             /* Unexpected ASTI error      */

/* Parameter area                                                     */
struct YMLCML_pl_mdl {

        /* FHDR                                                       */
        struct ESMFHDR hdr;

        /* Input parameters                                           */
        struct {
                char order_id[16];      /* Order Id                   */
                unsigned char order;    /* Order specification        */
                unsigned char wait;     /* Wait                       */
                char reserved1[2];      /* Reserved                   */
        } in_data;

        /* Output parameters                                          */
        struct {
                unsigned long rc;       /* Return code                */
                char ret_msg[160];      /* Return message             */
                unsigned long asti_rc;  /* Return code from ASTI      */
        } out_data;
};

/*  Entry for YMLCML  */
#ifdef __SNI_HOST_BS2000
#ifdef __cplusplus
extern "C" void _SVC(int, void*);
inline void YMLCML(struct YMLCML_pl_mdl& param)
{_SVC(20, &param);}
#else
void _SVC(int, void*);
#define YMLCML(p) _SVC(20, &p)
#endif
#endif
#endif          /* _YMLCML_H */
```

9.3.2.3 Include file YMLDML.H for YMLDML() - Delete mail

The include file YMLDML.H is listed below. You will find a program example that shows how to use the YMLDML() function call on page 481.

```
#ifndef _YMLDML_H
#define _YMLDML_H

#if 0
/*****************************************************************
 BEGIN-INTERFACE   YMLDML

 TITLE             (/ Delete send mail order /)
 NAME              YMLDML.H
 DOMAIN            MAIL
 LANGUAGE          C
 COPYRIGHT         (C) Fujitsu Technology Solutions GmbH 2010
                       ALL RIGHTS RESERVED
 COMPILATION-SCOPE USER
 INTERFACE-TYPE    CALL
 RUN-CONTEXT       TU
 PURPOSE           (/ Delete send mail order which is yet uncompleted
                   /)

 END-INTERFACE     YMLDML.
*****************************************************************/
#endif

/* order                                                        */
/* ENUM order_s                                                 */
#define YMLDML_all 1                  /* ALL                    */
#define YMLDML_single 2               /* SINGLE                 */

/* owner                                                        */
/* ENUM owner_s                                                 */
#define YMLDML_own 1                  /* OWN                    */
#define YMLDML_all_users 2            /* ALL                    */
#define YMLDML_other 3                /* OTHER                  */

/* main return codes                                            */
/* mret_code                                                    */
#define YMLDML_successful 0           /* No error detected      */
#define YMLDML_parameter_error 1      /* Parameter error        */
#define YMLDML_int_error 2            /* Internal error         */
#define YMLDML_order_not_found 3      /* Order not found        */
#define YMLDML_insuff_priv 4          /* Insufficient privileges*/
#define YMLDML_wrong_owner 5          /* Order not owned by     */
                                      /* specified owner        */
#define YMLDML_foreign_task 6         /* Order issued by foreign*/
                                      /* task                   */
```

```
#define YMLDML_asti_not_avail 7         /* Subsystem ASTI not         */
                                        /* available                  */
#define YMLDML_asti_error 8             /* Unexpected ASTI error      */
/* Parameter area                                                     */
struct YMLDML_pl_mdl {

        /* FHDR                                                       */
        struct ESMFHDR hdr;

        /* Input parameters                                           */
        struct {
                char order_id[16];      /* Order Id                   */
                unsigned char order;    /* Order specification        */
                unsigned char owner;    /* Owner specification        */
                char reserved1[2];      /* Reserved                   */
                char user_id[8];        /* User id of owner           */
        } in_data;

        /* Output parameters                                          */
        struct {
                unsigned long asti_rc;  /* Return code from ASTI      */
        } out_data;
};

/*  Entry for YMLDML  */
#ifdef __SNI_HOST_BS2000
#ifdef __cplusplus
extern "C" void _SVC(int, void*);
inline void YMLDML(struct YMLDML_pl_mdl& param)
{_SVC(20, &param);}
#else
void _SVC(int, void*);
#define YMLDML(p) _SVC(20, &p)
#endif
#endif

#endif          /* _YMLDML_H */
```

9.3.2.4 Include file YMLGML.H for YMLGML() - Query information about mail

The include file YMLGML.H is listed below. You will find a program example that shows how to use the YMLDML() function call on page 482.

```
#ifndef _YMLGML_H
#define _YMLGML_H

#if 0
/*****************************************************************
 BEGIN-INTERFACE   YMLGML

 TITLE             (/ Get info about send mail orders /)
 NAME              YMLGML.H
 DOMAIN            MAIL
 LANGUAGE          C
 COPYRIGHT         (C) Fujitsu Technology Solutions GmbH 2010
                       ALL RIGHTS RESERVED
 COMPILATION-SCOPE USER
 INTERFACE-TYPE    CALL
 RUN-CONTEXT       TU
 PURPOSE           (/ Get info about submitted send mail order(s)
                   /)

 END-INTERFACE     YMLGML.
*****************************************************************/
#endif

/* order                                                        */
/* ENUM order_s                                                 */
#define YMLGML_sum 1                  /* sum                    */
#define YMLGML_all 2                  /* all                    */
#define YMLGML_single 3               /* single                 */

/* owner                                                        */
/* ENUM owner_s                                                 */
#define YMLGML_own 1                  /* OWN                    */
#define YMLGML_all_users 2            /* ALL                    */
#define YMLGML_other 3                /* OTHER                  */

/* main return codes                                            */
/* mret_code                                                    */
#define YMLGML_successful 0           /* No error detected      */
#define YMLGML_parameter_error 1      /* Parameter error        */
#define YMLGML_int_error 2            /* Internal error         */
#define YMLGML_order_not_found 3      /* Order not found        */
#define YMLGML_out_too_small 4        /* Output area too small  */
#define YMLGML_order_not_own 5        /* Order not own          */
#define YMLGML_resource_sat 6         /* Resource saturation    */
#define YMLGML_foreign_task 7         /* Order issued by foreign*/
                                      /* task                   */
```

```
#define YMLGML_asti_not_avail 8         /* Subsystem ASTI not         */
                                        /* available                  */
#define YMLGML_not_mail_send_order 9    /* Not a mail send order      */
#define YMLGML_asti_error 10            /* Unexpected ASTI error      */

/* Parameter area                                                     */
struct YMLGML_pl_mdl {

        /* FHDR                                                       */
        struct ESMFHDR hdr;

        /* Input parameters                                           */
        struct {
                char order_id[16];      /* Order Id                   */
                unsigned char order;    /* Order specification        */
                unsigned char owner;    /* Owner specification        */
                char reserved1[2];      /* Reserved                   */
                char user_id[8];        /* User id of owner           */
                void* out;              /* Output area                */
                unsigned long outl;     /* Output area length         */
        } in_data;

        /* Output parameters                                          */
        struct {
                unsigned long sum;      /* Number of queued mails     */
                unsigned long asti_rc;  /* Return code from ASTI      */
        } out_data;
};

/*  Entry for YMLGML  */
#ifdef __SNI_HOST_BS2000
#ifdef __cplusplus
extern "C" void _SVC(int, void*);
inline void YMLGML(struct YMLGML_pl_mdl& param)
{_SVC(20, &param);}
#else
void _SVC(int, void*);
#define YMLGML(p) _SVC(20, &p)
#endif
#endif

/* order status                                                       */
/* ENUM status_s                                                      */
#define YMLGML_waiting 1                /* waiting                    */
#define YMLGML_deferred 2               /* deferred                   */
#define YMLGML_active 3                 /* active                     */
#define YMLGML_sent_ok 4                /* send successful            */
#define YMLGML_sent_fail 5              /* send failed                */

/* Data type selector                                                 */
/* ENUM slctr_s                                                       */
#define YMLGML_sel_order 1              /* order data                 */
```

```
#define YMLGML_sel_result 2             /* result data                */

/* rc                                                                 */
/* ENUM rc_s                                                          */
#define YMLGML_be_ok 0                  /* OK                         */
#define YMLGML_be_param_error 1         /* Parameter error            */
#define YMLGML_be_resource_sat 2        /* Resource saturation        */
#define YMLGML_be_smtp_error 3          /* SMTP error                 */
#define YMLGML_be_smime_error 4         /* SMIME error                */
#define YMLGML_be_int_error 10          /* Internal error             */

/* Output parameters                                                  */
struct YMLGML_order {
        unsigned long num_order;        /* number of orders           */
        char order_id[1][16];           /* Array of order ids, real   */
                                        /* array size: num_order      */
};

/* order specific data                                                */
struct YMLGML_order_data {
        unsigned long cnt_tries;        /* # sending tries            */
};

/* result specific data                                               */
struct YMLGML_result_data {
        unsigned long rc;               /* return code                */
        char ret_msg[160];              /* return message             */
};

/* mail parameter                                                     */
struct YMLGML_mail_data {
        unsigned long status;           /* mail status                */
        unsigned long time;             /* submission time            */
        char user[8];                   /* submitter                  */
        unsigned long slctr;            /* data selector              */

        /* UNION data                                                 */
        union /* _data */ {
                struct YMLGML_result_data result_data;
                                        /* result data                */
                struct YMLGML_order_data order_data;
                                        /* order data                 */
        } _data;
        unsigned long mail_par_len;     /* mail parameter length      */
        char mail_par[4];               /* mail parameters, real      */
                                        /* size: mail_par_len         */
};
```

```
/* STRUCT Output parameters                                          */
struct YMLGMLoutpar {

        /* UNION Output parameters                                   */
        union /* _outp */ {
                struct YMLGML_mail_data mail_data;
                                        /* mail data                 */
                struct YMLGML_order order_ids;
                                        /* order Ids                 */
        } _outp;
};

#endif          /* _YMLGML_H */
```

9.3.2.5 C program examples

The C program examples shown below indicate how to use the C function calls YMLSML(), YMLCML(), YMLDML() and YMLDML().

Example 1: YMLSML() and YMLCML() function calls

```
#include <stdio.h>
#include <stdlib.h>
#include "FHDR.H"
#include "YMLSML.H"
#include "YMLCML.H"

#define ALIGNMENT 4
#define PAD(length) ((length + ALIGNMENT - 1) & ~(ALIGNMENT - 1))

main(int argc, char *argv[])
{
  struct YMLSML_pl_mdl sendParam;
  struct YMLCML_pl_mdl checkParam;
  enum {UNIT = 940, SEND_FUNCTION = 20, CHECK_FUNCTION = 21,
        VERSION = 1, parLen = 2048, orderIdLen = 16};
  char sender[] = "Claudio.Monteverdi@mantova.example";
  char recipient[] = "Heinrich.Schuetz@dresden.example";
  char subject[] = "This is a test message";
  char msgText[] = "This is a test line.\n"
                   "And another line.\n";
  char srcCCSN[8 + 1] = "EDFO4DRV";
  char destCCSN[8 + 1] = "ISO88591";
  char charSetName[] = "ISO-8859-1";
  char *ptr;
  struct YMLSML_general *ptrGeneral;
  struct YMLSML_msg_att *ptrMsgAtt;
  struct YMLSML_data_spec *ptrDataSpec;
  struct YMLSML_charset *ptrCharset;
  struct YMLSML_encoding *ptrEncoding;
  char *mailPar;
  char orderId[orderIdLen];

  mailPar = ptr = malloc(parLen);
  /* [Do some error handling] */

  /* Supplying sender address */
  ptrGeneral = (struct YMLSML_general *) ptr;
  ptrGeneral->tag = YMLSML_tag_from;
  ptrGeneral->len = strlen(sender);
  ptr += sizeof(*ptrGeneral);
  memcpy(ptr, sender, ptrGeneral->len);
  ptr += PAD(ptrGeneral->len);
  /* Supplying recipient address */
  ptrGeneral = (struct YMLSML_general *) ptr;
  ptrGeneral->tag = YMLSML_tag_to;
```

```
ptrGeneral->len = strlen(recipient);
ptr += sizeof(*ptrGeneral);
memcpy(ptr, recipient, ptrGeneral->len);
ptr += PAD(ptrGeneral->len);

/* Supplying subject line */
ptrGeneral = (struct YMLSML_general *) ptr;
ptrGeneral->tag = YMLSML_tag_subject;
ptrGeneral->len = strlen(subject);
ptr += sizeof(*ptrGeneral);
memcpy(ptr, subject, ptrGeneral->len);
ptr += PAD(ptrGeneral->len);

/* Supplying message body */
ptrMsgAtt = (struct YMLSML_msg_att *) ptr;
ptrMsgAtt->tag = YMLSML_tag_msg_begin;
ptr += sizeof(*ptrMsgAtt);

ptrDataSpec = (struct YMLSML_data_spec *) ptr;
ptrDataSpec->tag = YMLSML_tag_data_spec;
ptrDataSpec->type = YMLSML_data;
ptrDataSpec->len = strlen(msgText);
ptr += sizeof(*ptrDataSpec);
memcpy(ptr, msgText, ptrDataSpec->len);
ptr += PAD(ptrDataSpec->len);

ptrCharset = (struct YMLSML_charset *) ptr;
ptrCharset->tag = YMLSML_tag_charset;
ptrCharset->binary = YMLSML_no;
memcpy(ptrCharset->src_charset_XHCS, srcCCSN,
       sizeof(ptrCharset->src_charset_XHCS));
memcpy(ptrCharset->dest_charset_XHCS, destCCSN,
       sizeof(ptrCharset->dest_charset_XHCS));
ptrCharset->len = strlen(charSetName);
ptr += sizeof(*ptrCharset);
memcpy(ptr, charSetName, ptrCharset->len);
ptr += PAD(ptrCharset->len);

ptrEncoding = (struct YMLSML_encoding *) ptr;
ptrEncoding->tag = YMLSML_tag_encoding;
ptrEncoding->mechanism = YMLSML_qp;
ptr += sizeof(*ptrEncoding);

ptrMsgAtt = (struct YMLSML_msg_att *) ptr;
ptrMsgAtt->tag = YMLSML_tag_msg_end;
ptr += sizeof(*ptrMsgAtt);
memset(&sendParam, 0x00, sizeof(sendParam));
FHDR_SET_RC_NIL(sendParam.hdr);
FHDR_MOD_IFID(sendParam.hdr, UNIT, SEND_FUNCTION, VERSION);
sendParam.in_data.mail_par = mailPar;
sendParam.in_data.mail_par_len = ptr - mailPar;
sendParam.in_data.wait = YMLSML_res;
```

```
  sendParam.in_data.sec_prot = YMLSML_smime;
  sendParam.in_data.encrypt = YMLSML_no;
  sendParam.in_data.sign = YMLSML_no;
  sendParam.in_data.cipher = YMLSML_cipher_from_optfile;
  memcpy(sendParam.in_data.optfile, "*NONE", 5);
  YMLSML(sendParam);
  if (sendParam.hdr.FHDR_RC_MAINCODE == YMLSML_successful) {
    printf("YMLSML called successfully\n");
    memcpy(orderId, sendParam.out_data.order_id, orderIdLen);
    printf("  Order id: %.16s\n", sendParam.out_data.order_id);
    if (sendParam.in_data.wait == YMLSML_yes) {
      printf("  Return code: %d\n", sendParam.out_data.rc);
      printf("  Return message: %.160s\n", sendParam.out_data.ret_msg);
      exit(0);
    }
  }
  else {
    printf("Error in call of YMLSML: %08X\n", sendParam.hdr.FHDR_RC_NBR);
    printf("  Order id: %.16s\n", sendParam.out_data.order_id);
    printf("  Return code: %d\n", sendParam.out_data.rc);
    printf("  Return message: %.160s\n", sendParam.out_data.ret_msg);
    if (sendParam.hdr.FHDR_RC_MAINCODE == YMLSML_asti_error)
      printf("  ASTI error: %08X\n", sendParam.out_data.asti_rc);
    exit(1);
  }

  memset(&checkParam, 0x00, sizeof(checkParam));
  FHDR_SET_RC_NIL(checkParam.hdr);
  FHDR_MOD_IFID(checkParam.hdr, UNIT, CHECK_FUNCTION, VERSION);
  checkParam.in_data.wait = YMLCML_yes;
  checkParam.in_data.order = YMLCML_single;
  memcpy(checkParam.in_data.order_id, orderId, orderIdLen);
  YMLCML(checkParam);
  if (checkParam.hdr.FHDR_RC_MAINCODE == YMLCML_successful) {
    printf("YMLCML called successfully\n");
    printf("  Order id: %.16s\n", checkParam.out_data.order_id);
    printf("  Return code: %d\n", checkParam.out_data.rc);
    printf("  Return message: %.160s\n", checkParam.out_data.ret_msg);
  }
  else {
    printf("Error in call of YMLCML: %08X\n", checkParam.hdr.FHDR_RC_NBR);
    printf("  Order id: %.16s\n", checkParam.out_data.order_id);
    printf("  Return code: %d\n", checkParam.out_data.rc);
    printf("  Return message: %.160s\n", checkParam.out_data.ret_msg);
    if (checkParam.hdr.FHDR_RC_MAINCODE == YMLCML_asti_error)
      printf("  ASTI error: %08X\n", checkParam.out_data.asti_rc);
  }
}
```

Example 2: YMLDML() function call

```
#include <stdio.h>
#include <stdlib.h>
#include "FHDR.H"
#include "YMLDML.H"

main(int argc, char *argv[])
{
  int c;
  int deleteAll = 0;
  struct YMLDML_pl_mdl deleteParam;
  enum {UNIT = 940, DELETE_FUNCTION = 22,
        VERSION = 1, orderIdLen = 16};
  char orderId[orderIdLen + 1];

  while ((c = getopt(argc, argv, "AO:")) != EOF) {
    if (c == 'O') {
      strupper(optarg, NULL);
      strncpy(orderId, optarg, orderIdLen);
      orderId[orderIdLen] = 0x00;
    }
    else if (c == 'A')
      deleteAll = 1;
  }
  memset(&deleteParam, 0x00, sizeof(deleteParam));
  FHDR_SET_RC_NIL(deleteParam.hdr);
  FHDR_MOD_IFID(deleteParam.hdr, UNIT, DELETE_FUNCTION, VERSION);
  if (deleteAll)
    deleteParam.in_data.order = YMLDML_all;
  else {
    deleteParam.in_data.order = YMLDML_single;
    memcpy(deleteParam.in_data.order_id, orderId, orderIdLen);
  }
  deleteParam.in_data.owner = YMLDML_own;
  YMLDML(deleteParam);
  if (deleteParam.hdr.FHDR_RC_MAINCODE == YMLDML_successful) {
    printf("YMLDML called successfully\n");
  }
  else {
    printf("Error in call of YMLDML: %08X\n",
           deleteParam.hdr.FHDR_RC_NBR);
    if (deleteParam.hdr.FHDR_RC_MAINCODE == YMLDML_asti_error)
      printf("  ASTI error: %08X\n", deleteParam.out_data.asti_rc);
    exit(1);
  }
}
```

Example 3: YMLGML() function call

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "FHDR.H"
#include "YMLGML.H"

main(int argc, char *argv[])
{
  int c;
  int getInfo = 0;
  struct YMLGML_pl_mdl getInfoParam;
  enum {UNIT = 940, GETINFO_FUNCTION = 23, VERSION = 1,
        orderIdLen = 16, maxListSize = 30, mailParSize = 32768};
  char orderId[orderIdLen + 1];
  struct YMLGML_order *orderIdList;
  int orderIdListSize = sizeof(orderIdList->num_order) +
                        sizeof(orderIdList->order_id) * maxListSize;
  struct YMLGMLoutpar *mailPar;
  char *statusTxt[5] =
    {"Waiting", "Deferred", "Active", "Sent", "Failed"};
  char *rcTxt[11] =
    {"Ok", "Parameter error", "Resource saturation", "SMTP error",
     "S/MIME error", NULL, NULL, NULL, NULL, NULL, "Internal error"};
  int i;
  unsigned long time;

  while ((c = getopt(argc, argv, "AO:S")) != EOF) {
    getInfo = c;
    if (getInfo == 'O') {
      strupper(optarg, NULL);
      strncpy(orderId, optarg, orderIdLen);
      orderId[orderIdLen] = 0x00;
    }
  }
  memset(&getInfoParam, 0x00, sizeof(getInfoParam));
  FHDR_SET_RC_NIL(getInfoParam.hdr);
  FHDR_MOD_IFID(getInfoParam.hdr, UNIT, GETINFO_FUNCTION, VERSION);
  if (getInfo == 'A') {
    getInfoParam.in_data.order = YMLGML_all;
    if (!(orderIdList = malloc(orderIdListSize))) {
      printf("malloc failed\n");
      exit(1);
    }
    getInfoParam.in_data.out = orderIdList;
    getInfoParam.in_data.outl = orderIdListSize;
  }
  else if (getInfo == 'S') {
    getInfoParam.in_data.order = YMLGML_sum;
    getInfoParam.in_data.out = NULL;
    getInfoParam.in_data.outl = 0;
  }
```

```
else {
  getInfoParam.in_data.order = YMLGML_single;
  memcpy(getInfoParam.in_data.order_id, orderId, orderIdLen);
  if (!(mailPar = malloc(mailParSize))) {
    printf("malloc failed\n");
    exit(1);
  }
  getInfoParam.in_data.out = mailPar;
  getInfoParam.in_data.outl = mailParSize;
}
getInfoParam.in_data.owner = YMLGML_own;
YMLGML(getInfoParam);
if (getInfoParam.hdr.FHDR_RC_MAINCODE == YMLGML_successful) {
  printf("YMLGML called successfully\n");
  if (getInfo == 'A' || getInfo == 'S')
    printf("  Number of mail orders: %d\n",
           getInfoParam.out_data.sum);
  if (getInfo == 'A') {
    for (i = 0; i < orderIdList->num_order; i++)
      printf("  OrderId[%d]: %.16s\n",
             i + 1, orderIdList->order_id[i]);
  }
  else if (getInfo == 'O') {
    printf("  Mail status: %s\n",
      statusTxt[mailPar->_outp.mail_data.status - 1]);
    printf("  Mail submitter: %.8s\n",
           mailPar->_outp.mail_data.user);
    time = mailPar->_outp.mail_data.time -
            (50 * 365 + 12) * 24 * 60 * 60;
    printf("  Mail send time: %s", ctime(&time));
    if (mailPar->_outp.mail_data.slctr == YMLGML_sel_result) {
      printf("  Mail result: %s\n",
             rcTxt[mailPar->_outp.mail_data._data.result_data.rc]);
      printf("  Mail result message: %s\n",
             mailPar->_outp.mail_data._data.result_data.ret_msg);
    }
    printf("  Mail parameter length: %d\n",
           mailPar->_outp.mail_data.mail_par_len);
    printf("  Mail parameter:");
    for (i = 0; i < mailPar->_outp.mail_data.mail_par_len; i++) {
      if (i % 32 == 0)
        printf("\n    ");
      else if (i % 4 == 0)
        printf(" ");
      printf("%02X", mailPar->_outp.mail_data.mail_par[i]);
    }
  }
}
```

```
  else {
    printf("Error in call of YMLGML: %08X\n",
           getInfoParam.hdr.FHDR_RC_NBR);
    if (getInfoParam.hdr.FHDR_RC_MAINCODE == YMLGML_asti_error)
      printf("  ASTI error: %08X\n", getInfoParam.out_data.asti_rc);
    exit(1);
  }
}
```

10 Frequently asked questions (FAQ)

- **Question:**
  What is the meaning of the output of the following messages after the FTP/TELNET client or FTP/TELNET server has been loaded?

```
BLS0340 UNRESOLVED EXTERNAL REFERENCES
BLS0342 ### 'YS6GSBN ……
BLS0342 ### 'YS6SOCE ……
BLS0342 ### 'YS6CLOS ……
BLS0342 ### 'YS6SHTD ……
BLS0342 ### 'YS6ERRO ……
```

  Answer:
  The programs were started either with START-PROG ..... without PROG-MODE=*ANY or the Socket subsystem SOC6 has not been started.

- **Question:**
  Connection setup with *open* on the FTP or TELNET client takes a very long time. What is the reason?

  Answer:
  For connection setup both the client and the server use DNS functions. If the associated resolver files are not correctly set, this can result in lengthy wait times. This is the case when, for example, the DNS server specified in the resolver files cannot be reached.

  The name of the resolver file in BS2000 is:

  SYSDAT.SOCKETS.nnn.SOC6.RESOLV or
  SYSDAT.LWRESD.nnn.RESOLV.CONF

- **Question:**
  What does the following FTP client message mean?

```
"time limit for server response exceeded"
```

  Answer:
  The client is waiting for an answer from the server which has not arrived after a predefined number of seconds (30 seconds). The reason for this can, for example, be that the network load is too high.
  You can often solve the problem by increasing the predefined timeout value of 30 seconds using the FTP user command *settime* (see page 195).

- **Question:**
  When I transfer a file using FTP and save it as a PAM file in the target system, the string “C-ENDOFFILE” is appended to it. What is the purpose of this, and how can I stop it?

  Answer:
  The string “C-ENDOFFILE” is normally used to mark the exact end of a PAM file. With the FTP user command `setfile datend off` you can stop “C-ENDOFFILE” being appended.

- **Question:**
  In earlier FTP versions (< V4.0) tabulator characters in a text file were automatically converted into the corresponding number of blanks. This is now no longer the case. What is the reason?

  Answer:
  In the course of supporting restart capability in BS2000-FTP as of V4.0 the default for the transfer of text files was changed from `ftyp text` to `ftyp textbin`. Consequently tabulator characters are no longer converted by default.

  If you still want to use conversion of the tabulator characters, you can select this via the *ftyp* command or via the option file:

  – *ftyp* command (see page 136). Specify `ftyp text` on the client or `quote site ftyp text` on the server.

  – Option file of the FTP client (*initialCommand* option, see page 99) or option file of the FTP server (*initialChildCmds* option, see the “interNetServices Administrator Guide”)

- **Question:**
  In earlier FTP versions (< V4.0), SAM files with a fixed record length for which *type binary* was specified were entirely in binary format and stored without end-of-record characters. In the current FTP version the target file contains end-of-record characters. How can I prevent this?

  Answer:
  ftyp settings are now also taken into account for SAM files with a fixed record length. If *ftyp* is not “binary”, the record structure of the original file is retained when the file is transferred. To prevent this, you must select *ftyp binary* explicitly.

- **Question:**
  In BS2000 FTP and TELNET it is possible to generate diagnostic information with the *trace* and *debug* commands (on clients) or with the *-T* and *-D* options (on servers). What is the difference between *trace* and *-T* on the one hand and *debug* and *-D* on the other?

  Answer:
  debug and *-D* generate diagnostic information which concerns the products FTP and TELNET. The highest useful level here is 2.
  trace and *-T*, by contrast, output diagnostic information generated by the Sockets. The highest useful level here is 9.

- **Question:**
  I have started a second FTP or TELNET server but cannot set up a connection to it. What can be the reason?

  Answer:
  The most frequent causes of the problem are:

  – Specification of a port number for the server which had already been allocated.
  – Use of an application name on the server (option *-A*) which had already been allocated.

- **Question:**
  My FTP login is rejected by the (BS2000) FTP server with the message `invalid login`. I cannot find any reason for this behavior, however. What should I do?

  Answer:
  If possible, enable the FTP trace on the FTP server using the following console command:

  ```
  /INFORM-PROGRAM 'debug 2',*TSN(<tsn ftpserver>)
  ```

  Repeat the login and, using the console command

  ```
  /NFORM-PROGRAM 'rdProt',*TSN(<tsn ftpserver>)
  ```

  store the trace in the file SYSOUT.TCP-IP-AP.*nnn*.FTPD.<MMDDHHMMSS>.

  (MMDDHHMMSS is the date and time specification in the format
  Month Day Hour Minute Second).

- **Question:**
  How do I reach a BS2000 FTP server with a Web browser?

  Answer:
  You can access the POSIX directory of the ID <userid> via the following URL:

  ```
  ftp://<userid>,<account-number>@<host-name>:<port-number>/
  ```

  This will at least allow you to output the directories. Access to the BS2000 directory is not possible via a Web browser.

- **Question:**
  In FTP it is possible to use `quote <command>` to send the command <command> to the server. However, there is also `quote site <command>` and `quote site exec <command>`. What is the difference?

  Answer:

  – With `quote <command>` you send FTP commands that conform with the standard to the server.

  – With `quote site <command>` you send BS2000-specific (“proprietary”) commands to the server which are not defined in the standard. These commands include *ftyp*, *cmod*, *modc*, *file*, *setc*, *sfil*.

  – With `quote site exec <command>` you send BS2000 commands to be executed to the server. To prevent misuse of these commands in the target system, this variant is disabled when FTAC is used (option *-FTAClevel* >0) or via the option *-disableSiteExecCommand* (see the “interNet Services Administrator Guide”).

- **Question:**
  When using FTP clients with a graphical user interface (GUI) there is often no way of specifying the account number required for the connection to BS2000. What should I do in this case?

  Answer:
  In cases like this enter the account number when you enter your ID as follows:

  ```
  <userid>,<account>
  ```

- **Question:**
  When fetching a file from a BS2000 FTP server my non-BS2000 FTP client aborts execution after a certain time without data transfer actually having begun.

  Answer:
  Some FTP clients show the progress of the transfer with a progress bar. For this purpose, the clients first of all use the FTP protocol command *SIZE* to query the size of the file from the server. Generally the server must read the relevant file in full to process this command. With very large files this can naturally take quite a time, with the result that the client's timeout monitoring clears the connection.
  Unfortunately on some clients this timeout monitoring cannot be reconfigured to permit longer wait times. In addition, in many cases a SIZE command is sent to the server even when the progress bar has been disabled.

  If an enhancement of the configuration options can be obtained from the vendor of the FTP client, there is an option in the BS2000 FTP server to disable the *SIZE* command using the *-disableSizeCommand* option (see the "interNet Services Administrator Guide").

  As a client cannot require that the server should support the *SIZE* command, transfer should always function. However, you must accept that the restart mechanism no longer functions because the *SIZE* command is needed for this.

- **Question:**
  When an LMS file is transferred from an NK2 pubset to an NK4 pubset, the destination file is no longer a valid LMS file.

  Answer:
  Transfer the file to a non-NK4 pubset on the destination computer, and then copy the LMS library to the NK4 pubset using LMS.

  Alternatively, you can convert the LMS library to an NK4 pubset on the source computer and then use FTP to transfer it to the NK4 pubset on the destination computer.

- **Question:**

  How can I get the Windows GUI FTP client FileZilla to enable TVFS for the respective session without specifying other session default values?

  Answer:

  Unfortunately, FileZilla does not yet provide an option on the graphical user interface that allows you to specify FTP server commands to be sent to the server at the beginning of an FTP session. However, by editing the XML file *%USERPROFILE%\AppData\Roaming\FileZilla\sitemanager.xml* manually, you can configure these types of commands. To do so, first generate a server entry using the graphical user interface, then search for the related <server> entry in the XML file once FileZilla has finished, and insert the *<Command>SITE SVFS ON</Command>* sequence

bracketed with *<PostLoginCommands> ... </PostLoginCommands>*. Using several *<Command> ... </Command>* sequences, you can specify other FTP server commands that are to be executed at the beginning of a session.

For example, to enable TVFS and use LBP (Last Byte Pointer) as the method for marking the EOF of a PAM file:

```
<Server>
   <Host>Testze04.example.com</Host>
   <Port>21</Port>
   ...
   <User>Username,Account</User>
   ...
   <PostLoginCommands>
      <Command>SITE SVFS ON</Command>
      <Command>SITE SFIL DATEND LBP</Command>
   </PostLoginCommands>
   ...
   <Name>TESTZE04</Name>
   ...
   <DirectoryComparison>0</DirectoryComparison>TESTZE04</Server>
```

- **Question:**

  How can I set the file attributes of the remote DMS file via the Windows GUI FTP client FileZilla?

  Answer:

  Generally, the *site file* command is to be used (see page 75). In the log window, select *Enter custom command* via right mouse click and enter the appropriate command in the field, e.g.:

  ```
  site file file-name,recf=f,recsize=1234
  ```

  file-name is the file name which FileZilla will use as target name of the file transfer (i.e. the name of the source file by default).

  One problem may be that FileZilla normally is working with multiple parallel FTP sessions ín order to be able to update remote file lists even during protracted file transfers. However, in this case the *site file* command may not be issued in the same FTP session as the subsequent file transfer. You can avoid this by limiting the maximum number of símultaneous FTP session to 1. To do this, open the server manager of the respective server and set the maximum simultaneous transfers to 1 in the transfer settings.

Related publications

The manuals are available as online manuals, see *http://manuals.ts.fujitsu.com*, or in printed form which must be paid and ordered separately at *http://manualshop.ts.fujitsu.com*.

openNet Server BS2000
BCAM
User Guide

interNet Services BS2000
Administrator Guide

openNet Server BS2000
IPv6 Introduction and Conversion Guide, Stage 1
User Guide

openNet Server BS2000
IPSec
User Guide

openNet Server, interNet Services
SNMP Management for openNet Server and interNet Services
User Guide

openFT (BS2000)
Command Interface
User Guide

CMX(BS2000)
Communication Method in BS2000
User Guide

SNMP Management V5.0
SNMP Management for BS2000/OSD
User Guide

SNMP Management V6.0
SNMP Management for BS2000/OSD
User Guide

C Library Functions
for POSIX Applications
Reference Manual

XHCS (BS2000/OSD)
8-Bit Code Processing in BS2000/OSD
User Guide

LMS (BS2000)
SDF Format
User Guide

BS2000
User Commands (ISP Format)
User Guide

BS2000 OSD/BC
Commands
User Guide

IMON (BS2000/OSD)
Installation Monitor
User Guide

BS2000 OSD/BC
System Administration
User Guide

Additional publications

SSL and TLS
Designing and Building Secure Systems
by Eric Rescorla
ISBN 0-201-61598-3

Contents
Detailed description of SSL and TLS and of the application environment

Secrets & Lies
Digital Security in a Networked World
by Bruce Schneier
ISBN 0-471-25311-1

Contents
Overview of IT security

Postfix
The Definitive Guide
by Kyle D. Dent
ISBN 0-596-00212-2

Contents
Relatively compact and up-to-date introduction to configuring and using the Postfix mail server.

http://shop.oreilly.com/product/9780596002121.do

Postfix
by Richard Blum
ISBN 0-672-32114-9

Contents
Detailed description of how to configure and use the Postfix mail server. The May 2001 edition is no longer quite up-to-date, and there may be slight differences as far as BS2000/OSD porting concerned. In addition, the focus is somewhat Linux-oriented. Nevertheless, the level of detail makes it a noteworthy book.

The Book of Postfix
State-ot-the-Art Message Transport
by Ralf Hildebrandt und Patrick Koetter
ISBN 978-1-59327-001-8

Contents
Extensive treatment of the issues of mail transport constraints and mail filtering. Of the books listed here, this one contains the most detailed description of TLS use. Theory-oriented chapters are complemented by detailed practical examples (generally on the basis of Linux systems).

http://www.postfix-book.com/index.html

RFCs

Comprehensive information on the Requests for Comments (RFCs) is available on the Internet Engineering Task Force's (IETF) home page:

www.ietf.org

Index

S

T

U