FUJITSU Software BS2000 interNet Services

Version 3.4A
*3 May 2016

Readme

All rights reserved, including intellectual property rights.
Technical data subject to modifications and delivery subject to availability. Any liability that the data and illustrations are complete, actual or correct is excluded. Designations may be trademarks and/or copyrights of the respective manufacturer, the use of which by third parties for their own purposes may infringe the rights of such owner.

1 Introduction

This Readme file contains the changes and extensions for interNet Services V3.4, which were implemented after the guides had been published.

*1 Changes to 1st release level May 2014 are marked with *1
*2 Changes to 2nd release level April 2015 are marked with *2
*3 Changes to 3rd release level November 2015 are marked with *3

1.1 Affected guides

The changes described here concern the following guides:

[1] interNet Services V3.4A
Administrator Guide
Order number U41095-J-Z125-5-76
Issued: December 2010

[2] interNet Services V3.4A
User Guide
Order number U41096-J-Z125-5-76
Issued: December 2010

2 Software extensions

2.1 New functionality with internet Services V3.4A10

Support of TLS protocols TLSv1.1 and TLSv1.2

The TLS/SSL support in the services FTP, TELNET, Mail Sender and Mail Reader is extended with the protocols TLSv1.1 and TLSv1.2.

Support of Last Byte Pointer in FTP

Additionally to the default marking of the exact end of a PAM file with the string “C-DATEIENDE“ a method named Last Byte Pointer (short: LBP) is supported. LBP uses data from the file catalog entry and the file itself is not modified. The LBP support has to be activated explicitly with the command setfile and quote site SFIL respectively.

Support of High Availability in FTP and TELNET

Newly with the SDF command SET-FTP-TELNET-PARAMERTERS created start files for FTP and TELNET (default: SYSENT.TCP-IP-AP.nnn.FTPD and .TELNETD respectively) have no longer a dependency on the HSI and are therefore deployable on all Business Servers.

2.2 Changes to the Administration Guide [1]

2.2.1 New functionality with TCP-IP-AP V5.2A10

Section 4.3 Configuring FTP via the option file

Extension/Change of option -tlsProtocol (page 86):

OpenSSL supports Versions 2 and 3 of the SSL protocol and also the TLS protocol in the versions 1, 1.1 and 1.2. Some of these protocols can be activated selectively using the *-tlsProtocol* option.

-tlsProtocol
[**+** | **-**] {**SSLv2 | SSLv3 | TLSv1 | TLSv1.1 | TLSv1.2 | ALL**} …

…

SSLv3
SSL protocol Version 3

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

TLSv1.1
TLS protocol Version 1.1

TLSv1.2
TLS protocol Version 1.2

Extension/Change of option -tlsCipherSuite (page 87):

Additional and extended entries in the „Permissible cipher mnemonics“ list:

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

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

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

AECDH
Anonymous cipher suites with Elliptic Curve Diffie-Hellman key negotiation.

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

aECDSA
Cipher suites using ECDSA authentication, i.e. the certificates carry ECDSA keys.

TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2

TLSv1.2, TLSv1.1, TLSv1, SSLv3 or SSLv2 cipher suites. Remark: There are no TLSv1.1 specific cipher suites.

AES128, AES256, AES
Cipher suites using 128 bit AES, 256 bit AES or either 128 or 256 bit AES.

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

CAMELLIA128, CAMELLIA256, CAMELLIA
Cipher suites using 128 bit CAMELLIA, 256 bit CAMELLIA or either 128 bit or 256 bit CAMELLIA.

SHA1, SHA
Cipher suites using SHA1 hash function.

[i] Because feasible attacks on SHA1 come nearer and nearer, one should change as soon as possible to cipher suites which use e.g. the hash functions SHA256 or SHA384. But this implies generally also the change to TLS protocol version 1.2.

SHA256, SHA384
Cipher suites using SHA256 and SHA384 hash function respectively for the MAC (Message Authentication Code) computation. With cipher suites using AESGCM and therefore AEAD (Authenticated Encryption with Associated Data) as MAC method the SHA256 and SHA384 respectively in the name has a different meaning,

The table of the available cipher suites on page 90 is extended with following entries:

Name	Version	Key exchange	Authentication	Encryption	MAC/ Digest
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

*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2
*2

DHE-RSA-AES128-GCM-SHA256	TLSv1.2	DH	RSA	AESGCM(128)	AEAD
DHE-DSS-AES356-GCM-SHA384	TLSv1.2	DH	DSS	AESGCM(256)	AEAD
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	TLSv1.2	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
ECDHE-ECDSA-AES256-SHA	SSLv3	ECDH	ECDSA	AES(256)	SHA1
ECDHE-ECDSA-AES128-SHA	SSLv3	ECDH	ECDSA	AES(128)	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

Section 5.3 Configuring TELNET using an option file

Extension/Change of option -Z Protocol (page 181):

OpenSSL supports Versions 2 and 3 of the SSL protocol and also the TLS protocol in the versions 1, 1.1 and 1.2. Some of these protocols can be activated selectively using the *-Z Protocol* option.

-Z Protocol
=[**+** | **-**] {**SSLv2** | **SSLv3** | **TLSv1** | **TLSv1.1** | **TLSv1.2** | **ALL**} …

…

SSLv3
SSL protocol Version 3

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

TLSv1.1
TLS protocol Version 1.1

TLSv1.2
TLS protocol Version 1.2

Extension/Change of option -Z CipherSuite (page 174):

Additional and extended entries in the „Permissible cipher mnemonics“ list:

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

*2 kEECDH, kECDHE
*2 Cipher suites with ephemeral Elliptic Curve Diffie-Hellman key negotiation, in-
*2 cluding anonymous suites.

*2

*2 EECDH, ECDHE
*2 Cipher suites with ephemeral Elliptic Curve Diffie-Hellman key negotiation, with-
*2 out anonymous suites.

*2

*2 AECDH
*2 Anonymous cipher suites with Elliptic Curve Diffie-Hellman key negotiation.

*2

*2 ECDH
*2 Cipher suites with Elliptic Curve Diffie-Hellman key negotiation, including
*2 anonymous, ephemeral and fixed ECDH.

*2

*2 aECDSA
*2 Cipher suites using ECDSA authentication, i.e. the certificates carry ECDSA
*2 keys.

*2

*2 TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2
*2 TLSv1.2, TLSv1.1, TLSv1, SSLv3 or SSLv2 cipher suites. Remark: There are no
*2 TLSv1.1 specific cipher suites.

*2

*2 AES128, AES256, AES
*2 Cipher suites using 128 bit AES, 256 bit AES or either 128 or 256 bit AES.

*2

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

*2

*2 CAMELLIA128, CAMELLIA256, CAMELLIA
*2 Cipher suites using 128 bit CAMELLIA, 256 bit CAMELLIA or either 128 bit or
*2 256 bit CAMELLIA.

*2

*2 SHA1, SHA
*2 Cipher suites using SHA1 hash function.

*2

*2 **[i]** Because feasible attacks on SHA1 come nearer and nearer, one should
*2 change as soon as possible to cipher suites which use e.g. the hash
*2 functions SHA256 or SHA384. But this implies generally also the change
*2 to TLS protocol version 1.2.

*2

*2 SHA256, SHA384
*2 Cipher suites using SHA256 and SHA384 hash function respectively for the MAC
*2 (Message Authentication Code) computation. With cipher suites using AESGCM
*2 and therefore AEAD (Authenticated Encryption with Associated Data) as MAC
*2 method the SHA256 and SHA384 respectively in the name has a different
*2 meaning,

*2

*2 For the extension of the table on page 177 with the available cipher suites see the
*2 corresponding table in section 2.2.1 of this Readme.

*2

2.2.2 New functionality with MAIL V3.3A08

*2 **Section 11.2.2 Configuration file for the mail sender backend**

*2

*2 Extension/Change of option tlsProtocol (page 374):

OpenSSL supports versions 2 and 3 of the SSL protocol and also the TLS protocol in the versions 1, 1.1 and 1.2. You can use the *tlsProtocol* option to select which of these protocols are to be enabled.

tlsProtocol
[**+** | **-**] {**SSLv2 | SSLv3 | TLSv1 | TLSv1.1 | TLSv1.2 | ALL**} …

…

SSLv2
SSL protocol Version 2

[i] Version 2 of the SSL protocol displays some security-related deficiencies and should therefore not be used if possible.

SSLv3
SSL protocol Version 3

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

TLSv1.1
TLS protocol Version 1.1

TLSv1.2
TLS protocol Version 1.2

Extension/Change of option tlsCipherSuite (page 375):

Additional and extended entries in the „Permissible cipher mnemonics“ list:

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

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

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

AECDH
Anonymous cipher suites with Elliptic Curve Diffie-Hellman key negotiation.

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

aECDSA
Cipher suites using ECDSA authentication, i.e. the certificates carry ECDSA keys.

TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2
TLSv1.2, TLSv1.1, TLSv1, SSLv3 or SSLv2 cipher suites. Remark: There are no TLSv1.1 specific cipher suites.

AES128, AES256, AES

Cipher suites using 128 bit AES, 256 bit AES or either 128 or 256 bit AES.

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

CAMELLIA128, CAMELLIA256, CAMELLIA
Cipher suites using 128 bit CAMELLIA, 256 bit CAMELLIA or either 128 bit or 256 bit CAMELLIA.

SHA1, SHA
Cipher suites using SHA1 hash function.

[i] Because feasible attacks on SHA1 come nearer and nearer, one should change as soon as possible to cipher suites which use e.g. the hash functions SHA256 or SHA384. But this implies generally also the change to TLS protocol version 1.2.

SHA256, SHA384
Cipher suites using SHA256 and SHA384 hash function respectively for the MAC (Message Authentication Code) computation. With cipher suites using AESGCM and therefore AEAD (Authenticated Encryption with Associated Data) as MAC method the SHA256 and SHA384 respectively in the name has a different meaning,

For the extension of the table on page 378 with the available cipher suites see the corresponding table in section 2.2.1 of this Readme.

One new configuration option is described below:

smtpReadMaxWaitTime

The *smtpReadMaxWaitTime* option specifies how long the mail sender backend shall wait for a response of the SMTP server, if needed. When a mail send order is terminated due to a too long wait time, then it will be repeated after a certain time (see options *smtpRetryTimeBase* and *smtpRetryTimeMaxExp*) as with error messages of the SMTP server hinting to a temporary problem.
Because the commands communicating with the mail sender backend (MODIFY-MAIL-SERVICE-PARAMETER, SHOW-MAIL-SERVICE-PARAMETER and STOP-MAIL-SERVICE) on the one hand and the communication of the backend with the SMTP server on the other hand are serialized, it is advisable for a prompt completion as possible to limit the time of a wait state of the backend due to e.g. a deadlock because of SMTP server problems.
On the other hand this limit should be not too drastically, because otherwise e.g. an overloaded SMTP server will be loaded even more due to transfer abortion and repetition. Times in the single minute range should be normally a good compromise.

smtpReadMaxWaitTime
<time>[s | m | h | d]

<time>
Maximum wait time
Default: 5m

Without specification of a measurement unit the specified <time> is interpreted as minutes. With specification of a measurement unit (s for second, m for minute, h

*2 for hour, d for day) this must be placed directly after <time>, i.e. without separating
*2 blank. Specifying 0 means, that the wait time is not limited.

2.2.3 New functionality with MAIL V3.3A06

Section 11.2.2 Configuration file for the mail sender backend

Two new configuration options are described below:

smtpRetryTimeBase

The *smtpRetryTimeBase* option defines the time base that is used to determine the time according to which a new mail dispatch is attempted if the mail dispatch fails. See *smtpRetryTimeMaxExp* for details.

smtpRetryTimeBase
<value>[s | m | h | d]

<value>
Time base
Default: 15m

If a unit is not specified, the <value> specified is in minutes. If a unit is specified (s for seconds, m for minutes, h for hours, d for days) it must be directly after <value>, i.e. without a blank.

smtpRetryTimeMaxExp

The *smtpRetryTimeMaxExp* option limits the increase in waiting time between two repeats of mail dispatch attempts. The waiting time normally doubles with every failed dispatch attempt in order to restrict CPU usage during persistent dispatch attempt problems. After doubling *smtpRetryTimeMaxExp* the waiting time remains at the value reached.

smtpRetryTimeMaxExp
<value>

<value>
Default: 6

If errors occur during connection setup to the SMTP mail server, double *smtpRetryTimeBase* is continually used as the waiting time until a renewed delivery attempt is made.

If the error only occurs later in the SMTP dialog, which could possibly mean that the problem is not a general server problem that is quickly noticed, but a mail-specific problem that is often only noticed after some time, then the waiting time between two dispatch attempts (beginning with *smtpRetryTimeBase*) doubles with every attempt until doubling of the *smtpRetryTimeMaxExp* is reached.

Maximum waiting time default between two delivery attempts:

Maximum waiting time = *smtpRetryTimeBase* x 2^ *smtpRetryTimeMaxExp*

Maximum wait interval = 15m * 2^6 = 960m = 16h

Scenario 1: Mail server not accessible

Renewed delivery attempts after	30 min	= 2 * 15m

Scenario 2: Connection setup to mail server possible; mail-specific error

Renewed delivery attempt after	15 min	= 15m * 2^0
Renewed delivery attempt after	30 min	= 15m * 2^1
Renewed delivery attempt after	1 h	= 15m * 2^2
Renewed delivery attempt after	2 h	= 15m * 2^3
Renewed delivery attempt after	4 h	= 15m * 2^4
Renewed delivery attempt after	8 h	= 15m * 2^5
All further delivery attempts after	16 h	= 15m * 2^6

until *maxQueueLifeTime* is reached (default: 5 days).

Note:
When reducing the *smtpRetryTimeBase,* the value for *smtpRetryTimeMaxExp* tends to be increased at the same time; otherwise frequent delivery attempt repeats overload the CPU.

2.2.4 New functionality with MAIL V3.3A02

Section 11.2.2 Configuration file for the mail sender backend

The new configuration option is described below:

maxQueueLifeTime

The *maxQueueLifeTime* option defines the maximum life expectancy of a mail, during which a failed mail dispatch is repeated.

maxQueueLifeTime
<lifetime>[s | m | h | d]

<lifetime>
  Default: 5d

If a unit is not specified, the <lifetime> specified is in days. If a unit is specified (s for seconds, m for minutes, h for hours, d for days) it must be directly after <lifetime>, i.e. without a blank.

Note:
The option *retryLimit* option (page 372) becomes invalid with the introduction of *maxQueueLifeTime*.

2.2.5 Corrections

Section 5.3.2 Options for the safe use of TELNET with the aid of authentication and encryption

Supplement:
The equals sign must follow the option name without a blank and there may not be a blank after the equals sign.

Section 5.3.3 -Z option Support of the START-TLS option

Correction to -Z tls-required (page 167)

-Z tls-required
[={yes | no | optional}]

optional
START-TLS support is activated optionally, i.e. TLS security is only performed if requested by the Telnet client.

*3 **Section 4.3 Configuring FTP via the option file**
*3 **Section 5.3 Configuring TELNET using an option file**
*3
*3 Because due to security reasons SSLv2 isn’t supported anymore by the now used
*3 version of the OpenSSL library, the specification of SSLv2 with the
*3 option -tlsProtocol and -Z Protocol is factually ignored.

2.3 Changes to the User Guide [2]

*2 2.3.1 New functionality with MAIL V3.3A08 and TCP-IP-AP V5.2A10

*2 **Section 3.3 Overview of SSL**

*2 Addition of supported TLS versions (page 39).

*2 With introduction of the named MAIL and TCP-IP-AP versions the OpenSSL toolkit
*2 is used in version 1.0.2d. The supported protocol versions are SSLv2, SSLv3,
*2 TLSv1, TLSv1.1 and TLSv1.2.

*2 **Section 3.3.2 SSL and TLS**

*2 Extension of warning (page 40):

*2 The SSL protocol in version 2 and 3 displays some security-related deficiencies
*2 and should therefore not be used if possible.
*2 Some security-related deficiencies are remedied in a seminal way first with TLS
*2 version 1.2, therefore this version should be given preference over older ones if
*2 possible.

*2 **Kapitel 4.1 FTP servers in BS2000/OSD**

*2 Calling the FTP server functions via the FTP partner client (page 65):

*2 *quote site sfil datend on|off|lbp*
*2 Enable/Disable the special EOF marker (default: enabled) or usage of the new
*2 EOF marking method Last Byte Pointer (LBP).

*2 **Kapitel 4.7 Parameter selection using option files**

*2 Extension/Change of option -tlsProtocol (page 93):

*2 OpenSSL supports Versions 2 and 3 of the SSL protocol and also the TLS proto-
*2 col in the versions 1, 1.1 and 1.2. Some of these protocols can be activated selec-
*2 tively using the *-tlsProtocol* option.

-tlsProtocol
[**+** | **-**] {**SSLv2** | **SSLv3** | **TLSv1** | **TLSv1.1** | **TLSv1.2** | **ALL**} …

*2 …

*2 **SSLv3**
*2 SSL protocol Version 3

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

*2 **TLSv1.1**
*2 TLS protocol Version 1.1

*2 **TLSv1.2**
*2 TLS protocol Version 1.2

*2 Extension/Change of option -tlsCipherSuite (page 94):
*2
*2 Additional and extended entries in the „Permissible cipher mnemonics“ list:
*2
*2 kEDH, kDHE
*2 Cipher suites with ephemeral Diffie-Hellman key negotiation, including anony-
*2 mous suites.
*2
*2 kEECDH, kECDHE
*2 Cipher suites with ephemeral Elliptic Curve Diffie-Hellman key negotiation, in-
*2 cluding anonymous suites.
*2
*2 EECDH, ECDHE
*2 Cipher suites with ephemeral Elliptic Curve Diffie-Hellman key negotiation, with-
*2 out anonymous suites.
*2
*2 AECDH
*2 Anonymous cipher suites with Elliptic Curve Diffie-Hellman key negotiation.
*2
*2 ECDH
*2 Cipher suites with Elliptic Curve Diffie-Hellman key negotiation, including
*2 anonymous, ephemeral and fixed ECDH.
*2
*2 aECDSA
*2 Cipher suites using ECDSA authentication, i.e. the certificates carry ECDSA
*2 keys.
*2
*2 TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2
*2 TLSv1.2, TLSv1.1, TLSv1, SSLv3 or SSLv2 cipher suites. Remark: There are no
*2 TLSv1.1 specific cipher suites.
*2
*2 AES128, AES256, AES
*2 Cipher suites using 128 bit AES, 256 bit AES or either 128 or 256 bit AES.
*2
*2 AESGCM
*2
*2 Cipher suites using AES in Galois Counter Mode (GCM). These cipher suites
*2 are only supported in TLSv1.2.
*2
*2 CAMELLIA128, CAMELLIA256, CAMELLIA
*2 Cipher suites using 128 bit CAMELLIA, 256 bit CAMELLIA or either 128 bit or
*2 256 bit CAMELLIA.
*2
*2 SHA1, SHA
*2 Cipher suites using SHA1 hash function.
*2
*2 **[i]** Because feasible attacks on SHA1 come nearer and nearer, one should
*2 change as soon as possible to cipher suites which use e.g. the hash
*2 functions SHA256 or SHA384. But this implies generally also the change
*2 to TLS protocol version 1.2.
*2
*2 SHA256, SHA384
*2 Cipher suites using SHA256 and SHA384 hash function respectively for the MAC
*2 (Message Authentication Code) computation. With cipher suites using AESGCM
*2 and therefore AEAD (Authenticated Encryption with Associated Data) as MAC
*2 method the SHA256 and SHA384 respectively in the name has a different
*2 meaning,
*2
*2 For the extension of the table on page 97 with the available cipher suites see the
*2 corresponding table in section 2.2.1 of this Readme.
*2

Section 4.10 Overview of commands (FTP-Client)

setfile - Enable/Disable file marker (page 186):

There are two methods to mark the exact end of a PAM file. The conventional method uses for this a special string, which contains amongst others "C-DATEIENDE". The new method named Last Byte Pointer (short: LBP) uses information from the file catalog entry, the file itself isn't modified. Shall the file be processed by programs having difficulties with the marking string, then the attaching of a marker has to be disabled or alternatively the LBP method has to be used.

setfile
[datend on **|** off **|** lbp] [pademptyrec on **|** off]

datend on **|** off **|** lbp
Enables/disables the usage of the end of file marker „C-DATEIENDE“ or enables the usage of the new end of file marking method LBP respectively.

Section 6.1.3.3 START-TLS option

Extension/Change of option -Z Protocol (page 294):

OpenSSL supports Versions 2 and 3 of the SSL protocol and also the TLS protocol in Versions 1, 1.1 and 1.2. Some of these protocols can be activated selectively using the *-Z Protocol* option.

-Z Protocol
=[**+** | **-**] {**SSLv2 | SSLv3 | TLSv1 | TLSv1.1 | TLSv1.2 | ALL**} …

…

SSLv3
SSL protocol Version 3

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

TLSv1.1
TLS protocol Version 1.1

TLSv1.2
TLS protocol Version 1.2

Extension/Change of option -Z CipherSuite (page 288):

Additional and extended entries in the „Permissible cipher mnemonics“ list:

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

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

EECDH, ECDHE

*2 Cipher suites with ephemeral Elliptic Curve Diffie-Hellman key negotiation, without anonymous suites.

*2 AECDH
Anonymous cipher suites with Elliptic Curve Diffie-Hellman key negotiation.

*2 ECDH
Cipher suites with Elliptic Curve Diffie-Hellman key negotiation, including anonymous, ephemeral and fixed ECDH.

*2 aECDSA
Cipher suites using ECDSA authentication, i.e. the certificates carry ECDSA keys.

*2 TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2
TLSv1.2, TLSv1.1, TLSv1, SSLv3 or SSLv2 cipher suites. Remark: There are no TLSv1.1 specific cipher suites.

*2 AES128, AES256, AES
Cipher suites using 128 bit AES, 256 bit AES or either 128 or 256 bit AES.

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

*2 CAMELLIA128, CAMELLIA256, CAMELLIA
Cipher suites using 128 bit CAMELLIA, 256 bit CAMELLIA or either 128 bit or 256 bit CAMELLIA.

*2 SHA1, SHA
Cipher suites using SHA1 hash function.

*2 **[i]** Because feasible attacks on SHA1 come nearer and nearer, one should change as soon as possible to cipher suites which use e.g. the hash functions SHA256 or SHA384. But this implies generally also the change to TLS protocol version 1.2.

*2 SHA256, SHA384
Cipher suites using SHA256 and SHA384 hash function respectively for the MAC (Message Authentication Code) computation. With cipher suites using AESGCM and therefore AEAD (Authenticated Encryption with Associated Data) as MAC method the SHA256 and SHA384 respectively in the name has a different meaning,

*2 For the extension of the table on page 291 with the available cipher suites see the corresponding table in section 2.2.1 of this Readme.

*2 **Section 8.2.3 POP3/IMAP servers: SERVER parameter section**

*2 Extension/Change of option PROTOCOL (page 370):

*2 **PROTOCOL=<protocol spec>**
You can limit the protocols used. SSL Version 2 and 3 and TLS Version 1, 1.1 and 1.2 are supported.
You can specify SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2 and ALL.

*2 Extension/Change of option CIPHER_SUITE (page 370):

*2 Additional and extended entries in the „Permissible cipher mnemonics“ list:

*2 kEDH, kDHE
*2 Cipher suites with ephemeral Diffie-Hellman key negotiation, including anonymous
*2 suites.
*2
*2 kEECDH, kECDHE
*2 Cipher suites with ephemeral Elliptic Curve Diffie-Hellman key negotiation, including
*2 anonymous suites.
*2
*2 EECDH, ECDHE
*2 Cipher suites with ephemeral Elliptic Curve Diffie-Hellman key negotiation, without
*2 anonymous suites.
*2
*2 AECDH
*2 Anonymous cipher suites with Elliptic Curve Diffie-Hellman key negotiation.
*2
*2 ECDH
*2 Cipher suites with Elliptic Curve Diffie-Hellman key negotiation, including
*2 anonymous, ephemeral and fixed ECDH.
*2
*2 aECDSA
*2 Cipher suites using ECDSA authentication, i.e. the certificates carry ECDSA
*2 keys.
*2
*2 TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2
*2 TLSv1.2, TLSv1.1, TLSv1, SSLv3 or SSLv2 cipher suites. Remark: There are no
*2 TLSv1.1 specific cipher suites.
*2
*2 AES128, AES256, AES
*2 Cipher suites using 128 bit AES, 256 bit AES or either 128 or 256 bit AES.
*2
*2 AESGCM
*2 Cipher suites using AES in Galois Counter Mode (GCM). These cipher suites
*2 are only supported in TLSv1.2.
*2
*2 CAMELLIA128, CAMELLIA256, CAMELLIA
*2 Cipher suites using 128 bit CAMELLIA, 256 bit CAMELLIA or either 128 bit or
*2 256 bit CAMELLIA.
*2
*2 SHA1, SHA
*2 Cipher suites using SHA1 hash function.
*2
*2 **[i]** Because feasible attacks on SHA1 come nearer and nearer, one should
*2 change as soon as possible to cipher suites which use e.g. the hash
*2 functions SHA256 or SHA384. But this implies generally also the change
*2 to TLS protocol version 1.2.
*2
*2 SHA256, SHA384
*2 Cipher suites using SHA256 and SHA384 hash function respectively for the MAC
*2 (Message Authentication Code) computation. With cipher suites using AESGCM
*2 and therefore AEAD (Authenticated Encryption with Associated Data) as MAC
*2 method the SHA256 and SHA384 respectively in the name has a different
*2 meaning,
*2
*2 For the extension of the table on page 373 with the available cipher suites see the
*2 corresponding table in section 2.2.1 of this Readme.
*2

2.3.2 Corrections

*3 **Section 3.4.1 MAKE.CERT procedure - Generating test certificates and CSRs**

*3 Key length correction (page 45 and 47)

*3 The procedure MAKE.CERT generates for RSA a key pair with 2048 bit key
*3 length, with DSA the key length is still 1024 bit.

Section 7.3.1 scp – Secure copying of files between computers in the network

Extension to option `–X binary`:

In the case of transfers to EBCDIC servers, EBCDIC must also be transformed to ASCII (e.g. with the Posix command iconv -f edf04 -t 8859) either beforehand or afterwards in order to effectively obtain a binary transfer. An additional transformation from ASCII to EBCDIC (e.g. with the Posix command iconv -f 8859 -t edf04) must be accordingly performed for transfers from EBCDIC servers.

*1 **Section 8.1 Starting/stopping the mail reader**
*1
*1 **Stopping the mail reader**
*1
*1 For the /INTR commands, the following conditions apply:
*1 - only allowed via the console interface by the system operator
*1 - only works, if the mail reader is executed as a batch task
*1 - the TSN of batch task is given
*1
*1 The user stops as a batch task executed mail reader with:
*1
*1 /CANCEL-JOB JOB-IDENTIFICATION=*TSN(TSN=<tsn of batch task>)

*1 **Section 8.2 Configuration file**
*1
*1 **Changing the configuration of the mail reader using the configuration file**
*1
*1 For the /INTR command, the following condition holds:
*1 - only allowed via the console interface by the system operator

*2 **Section 8.4.1 Structure of an e-mail**
*2
*2 **Supplement regarding the topic ‘character set conversion’**
*2
*2 The Mail Reader converts all data gotten from the IMAP or POP3 server from ISO-
*2 8859-1 to EDF041. When a MIME part of the mail uses ‘base64’ as Content-
*2 Transfer-Encoding and the Content-Type is ‘text’, then after the base64 decoding
*2 the data is converted again from ISO-8859-1 to EDF041. In all other cases a re-
*2 quired character set conversion has to be done by the processing procedures.