**DRAFT AMENDMENT** ISO/IEC DIS 8802-11/Amd.7

Attributed to ISO/IEC JTC 1 by the Central Secretariat (see page iii)

| Voting begins on | Voting terminates on |
|---|---|
| **2005-10-07** | **2006-03-07** |

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОММИСИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

# FAST-TRACK PROCEDURE

# Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements —

## Part 11:
## Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications

## AMENDMENT 7: Specifications for Enhanced Security — WLAN Authentication and Privacy Infrastructure (WAPI)

*Technologies de l'information — Télécommunications et échange d'information entre systèmes — Réseaux locaux et métropolitains — Exigences spécifiques —*

*Partie 11: Spécifications pour le contrôle d'accès au support et la couche physique*

*AMENDEMENT 7: Caractéristiques pour la sécurité améliorée — Authentification de WLAN et infrastructure privée (WAPI)*

ICS 35.110

**In accordance with the provisions of Council Resolution 21/1986 this DIS is circulated in the English language only.**

**Conformément aux dispositions de la Résolution du Conseil 21/1986, ce DIS est distribué en version anglaise seulement.**

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

© International Organization for Standardization, 2005
© International Electrotechnical Commission, 2005

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

**PDF disclaimer**

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

## Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

© ISO/IEC 2005 — All rights reserved

## NOTE FROM ITTF

This draft International Standard is submitted for JTC 1 national body vote under the Fast-Track Procedure.

In accordance with Resolution 30 of the JTC 1 Berlin Plenary 1993, the proposer of this document recommends assignment of ISO/IEC 8802-11 to JTC 1/SC 6.

**"FAST-TRACK" PROCEDURE**

**1** Any P-member and any Category A liaison organization of ISO/IEC JTC 1 may propose that an existing standard from any source be submitted directly for vote as a DIS. The criteria for proposing an existing standard for the fast-track procedure are a matter for each proposer to decide.

**2** The proposal shall be received by the ITTF which will take the following actions.

**2.1** To settle the copyright and/or trade mark situation with the proposer, so that the proposed text can be freely copied and distributed within JTC 1 without restriction.

**2.2** To assess in consultation with the JTC 1 secretariat which SC is competent for the subject covered by the proposed standard and to ascertain that there is no evident contradiction with other International Standards.

**2.3** To distribute the text of the proposed standard as a DIS. In case of particularly bulky documents the ITTF may demand the necessary number of copies from the proposer.

**3** The period for combined DIS voting shall be six months. In order to be accepted the DIS must be supported by 75 % of the votes cast (abstention is not counted as a vote) and by two-thirds of the P-members voting of JTC 1.

**4** At the end of the voting period, the comments received, whether editorial only or technical, will be dealt with by a working group appointed by the secretariat of the relevant SC.

**5** If, after the deliberations of this WG, the requirements of 3 above are met, the amended text shall be sent to the ITTF by the secretariat of the relevant SC for publication as an International Standard.

If it is impossible to agree to a text meeting the above requirements, the proposal has failed and the procedure is terminated.

In either case the WG shall prepare a full report which will be circulated by the ITTF.

**6** If the proposed standard is accepted and published, its maintenance will be handled by JTC 1.

© ISO/IEC 2005 – All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

# Contents


2 Normative references……1

3 Definitions……2

4 Abbreviations and acronyms……3

5 General description……4

5.3 Logical service interfaces……4
5.3.1 SS……5
5.4 Overview of the services……5
5.4.3 Access and confidentiality control services……5
5.4.3.1 ~~Authentication~~**Linkverification**……5
5.4.3.2 ~~Deauthentication~~ Delinkverification……7
5.4.3.3 Privacy……7
5.4.3.4 Authentication……8
5.7.5 Privacy……9
5.7.6 ~~Authentication~~ Linkverification……9
5.7.7 ~~Deauthentication~~ Delinkverification……11
5.8 Reference model……11
5.9 Establishing the security association……12
5.9.1 Infrastructure mode……12
5.9.2 IBSS mode……12

6 MAC service definition……13

6.1 Overview of MAC services……13
6.1.2 Security services……13

7 Frame formats……14

7.1 MAC frame formats……14
7.1.3 Frame fields……14
7.1.3.1 Frame Control field……14
7.1.3.1.2 Type and Subtype fields……15
7.1.3.1.9 ~~WEP~~Protected Frame field……15
7.2 Format of individual frame types……15
7.2.2 Data frames……15
7.2.3 Management frames……15
7.2.3.1 Beacon frame format……15
7.2.3.4 Association Request frame format……15
7.2.3.6 Reassociation Request frame format……16
7.2.3.9 Probe Response frame format……16
7.2.3.10 ~~Authentication~~ Linkverification frame format……16
7.2.3.11 ~~Deauthentication~~ Delinkverification……17




--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3 Management frame body components……17
7.3.1 Fixed fields……17
7.3.1.1 ~~Authentication~~ Linkverification Algorithm Number field……17
7.3.1.2 ~~Authentication~~ Linkverification Transaction Sequence Number field……18
7.3.1.4 Capability Information field……18
7.3.1.7 Reason Code field……19
7.3.1.9 Status Code field……19
7.3.2 Information element……20
7.3.2.25 WAPI information element……21
7.3.2.25.1 Authentication and Key Management Suite……22
7.3.2.25.2 Unicast and Multicast Cipher Suites……23

8 Security ……23

8.1 WAI authentication and key management……23
8.1.1 The structure of the authentication system……24
8.1.1.1 Systems and Ports……24
8.1.1.2 Controlled and Uncontrolled access……25
8.1.2 WAPI security association management……27
8.1.2.1 WAPI security association definitions……27
8.1.2.2 Selection of WAPI security policy……30
8.1.3 Certificate……31
8.1.4 WAI protocol……37
8.1.4.1 Format of WAI protocol packet ……37
8.1.4.2 Certificate Authentication procedure……42
8.1.4.3 Unicast key negotiation procedure……49
8.1.4.4 Multicast key / STAKey announcement process……54
8.1.4.5 STAKey establishment procedure……57
8.1.4.6 Pre-authentication……58
8.1.4.7 Cached BKSAs and WAPI key management……60
8.1.4.8 Rekeying……60
8.1.4.9 Timeout processing……60
8.1.4.10 Key derivation architecture……61
8.1.4.11 WAI protocol packet’s fragmentation and defragmentation……63
8.1.4.12 Port control and data transmission……64
8.2 WPI privacy infrastructure……64
8.2.1 Opertate mode……64
8.2.2 Key……65
8.2.3 Encapsulation and decapsulation……66
8.2.4 Rules for using data packet serial number PN……67
8.3 WAPI Authentication and key management state machine……68
8.3.1 WAPI ASUE Authentication and key management state machine……68
8.3.1.1 ASUE state machine states……70
8.3.1.2 ASUE state machine variables……71
8.3.1.3 ASUE state machine procedures……71



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.3.2 WAPI AE Authentication and key management state machine……71
8.3.2.1 AE state machine states……76
8.3.2.2 AE state machine variables……77
8.3.2.3 AE state machine procedures……78

10. Layer management……78

10.3 MLME SAP interface……78
10.3.2 Scan……78
10.3.2.2 MLME-SCAN.confirm……78
10.3.4 ~~Authenticate~~ Linkverification……78
10.3.4.1 MLME-~~AUTHENTICATE~~ LinkVerify .request……78
10.3.4.2 MLME-~~ AUTHENTICATE~~ LinkVerify.confirm……80
10.3.4.3 MLME- ~~AUTHENTICATE~~ LINKVERIFY .indication……81
10.3.5 ~~Deauthenticate~~ DELINKVERIFY……82
10.3.5.1 MLME- ~~DEAUTHENTICATE~~ DELINKVERIFY. request……82
10.3.5.2 MLME- ~~DEAUTHENTICATE~~ DELINKVERIFY.confirm……83
10.3.5.3 MLME- ~~DEAUTHENTICATE~~ DELINKVERIFY. indication……84
10.3.6 Associate……85
10.3.6.1 MLME-ASSOCIATE.request……85
10.3.6.3 MLME-ASSOCIATE.indication……86
10.3.7 Reassociate……86
10.3.7.1 MLME-REASSOCIATE.request……86
10.3.7.3 MLME-REASSOCIATE.indication……87
10.3.17 MLME-SETWPIKEYS……87
10.3.17.1 MLME-SETWPIKEYS.request……87
10.3.17.2 MLME-SETWPIKEYS.confirm……89
10.3.18 MLME-DELETEWPIKEYS……89
10.3.18.1 MLME-DELETEWPIKEYS.request……89
10.3.18.2 MLME-DELETEWPIKEYS.confirm……90
10.3.19 MLME-STAKEYESTABLISHED……91
10.3.19.1 MLME-STAKEYESTABLISHED.indication……91
10.3.20 SetProtection……91
10.3.20.1 MLME-SETPROTECTION.request……91
10.3.20.2 MLME-SETPROTECTION.confirm……93
10.3.21 MLME-PROTECTEDFRAMEDROPPED……93
10.3.21.1 MLME- PROTECTEDFRAMEDROPPED.indication……93

11 MLME ……94

11.3 Association and reassociation……94
11.3.1 Linkverification—originating STA……94
11.3.2 Linkverification—destination STA……94
11.3.3 Delinkverification—originating STA……95
11.3.4 Delinkverification—destination STA……95
11.4 Association, reassociation, and disassociation……95



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.4.1 STA association procedures.....95
11.4.2 AP association procedures.....96
11.4.3 STA reassociation procedures.....97
11.4.4 AP reassociation procedures.....97
11.4.5 STA disassociation procedures.....98
11.4.6 AP disassociation procedures.....98

Annex A (normative) Protocol Implementation Conformance Statements (PICS) .....99

A.4 PICS proforma—ISO/IEC 8802.11,2005 Edition.....99

A.4.4 MAC protocol.....99

Annex C (normative) Formal description of MAC operation.....102

Annex D (normative) ASN.1 encoding of the MAC and PHY MIB.....163

Annex H (informative) Reference implementations of the frame authentication algorithm and the key derivation algorithm and the test vectors.....191

H.1 Frame authentication algorithm.....191

H.1.1 Reference implementation.....191
H.1.2 Test vectors.....193

H.2 Key derivation algorithm.....194

H.2.1 Reference implementation.....194
H.2.2 Test vectors.....195

Annex I (Informative) The example of WAI parameters and WPI block cryptographic algorithm .....199

I.1 Principle.....199

I.2 Algorithm used in China.....199

I.3 ECC parameters used in China.....199



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

## List of figures

Figure 11—Portion of the ISO/IEC basic reference model covered in this amendment.....11
Figure 11a—The establishment of security association under the basic mode.....12
Figure 13—Frame Control field.....14
Figure 24—~~Authentication~~ Linkverification Algorithm Number fixed field.....18
Figure 25—~~Authentication~~ Linkverification Transaction Sequence Number fixed field.....18
Figure 42a—WAPI Information Element format.....21
Figure 42b—Pre-authentication.....22
Figure 42c—Suite selector format.....22
Figure 42d—Suite selector format.....23
Figure 43a—Uncontrolled and controlled ports..... 25
Figure 43b—Authentication state on controlled port..... 26
Figure 43c—Usage of the controlled and uncontrolled port.....27
Figure 43d—ASUE, AE and ASE roles.....27
Figure 43e—Format of certificate..... 32
Figure 43f—Definition of certificate content..... 33
Figure 43g—Extension Attribute format..... 35
Figure 43h—Issue format of the certificate..... 35
Figure 43i—Digest field..... 36
Figure 43j—Attribute field..... 36
Figure 43k—Format of WAPI protocol packet in WAI authentication system.....37
Figure 43l—FLAG.....38
Figure 43m—Certificate..... 39
Figure 43n—Identity..... 39
Figure 43o—ADDID..... 40
Figure 43p—Attribute format..... 40
Figure 43q—Signature attribute..... 40
Figure 43r—Certificate Verification Result..... 41
Figure 43s—Identity List..... 42
Figure 43t—Certificate Authentication procedure..... 42
Figure 43u—The format of the Data field of Authentication Activation packet.....42
Figure 43v—The fomat of the Data field of access authentication request packet.....44
Figure 43w—The format of the Data field of Certificate Authentication Request packet..... 45
Figure 43x—The format of the Data field of Certificate Authentication Response packet..... 46
Figure 43y—The format of the Data filed of Access Authentication Response packet..... 48
Figure 43z—Unicast key negotiation procedure..... 50
Figure 43aa—The format of the Data field of Unicast Key Negotiation Request packet..... 50
Figure 43ab—The format of the Data field of Unicast Key Negotiation Responding packet ..... 51
Figure 43ac—The format of the Data field Unicast Key Negotiation Confirmation packet..... 53
Figure 43ad—Multicast key / inter-station key announcing procedure..... 54
Figure 43ae—The format of the Data field of Multicast key / STAKey announcement packet.....54



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Figure 43af—The format of the Data field of the multicast key/STAKey response packet.....56
Figure 43ag—The flow chart of the STAKey establishment..... 57
Figure 43ah—The format of the Data field in the STAKey establishment request packet..... 58
Figure 43ai—The format of the Data field in the pre-authentication start packet..... 59
Figure 43aj—BK key derivation architecture.....61
Figure 43ak—Unicast key derivation architecture..... 62
Figure 43al—Multicast / STAKey derivation architecture..... 62
Figure 43am—Preshared key derivation architecture..... 63
Figure 43an—Operate modes.....65
Figure 43ao—WPI's MPDU encapsulation structure.....66
Figure 43ap—Integrity check data.....67
Figure 43aq—ASUE state machine, Part 1.....69
Figure 43ar—ASUE state machine, Part 2.....70
Figure 43as—AE state machine, Part 1.....73
Figure 43at—AE state machine, Part 2.....74
Figure 43au—AE state machine, Part 3.....74
Figure 43av—AE state machine, Part 4.....75
Figure 43aw—AE state machine, Part 5.....75



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

## List of tables

Table 1—Valid type and subtype combinations..... 15
Table 5—Beacon frame body.....15
Table 7—Association Request frame body.....16
Table 9—Reassociation Request frame body.....16
Table 12—Proble Response frame body.....16
Table 13—~~Authentication~~ Linkverification frame body.....17
Table 15—~~Deauthentication~~ Delinkverification frame body.....17
Table 18—Reason codes.....19
Table 19—Status codes.....20
Table 20—Element IDs.....21
Table 20f—Authentication and Key Management Suites.....22



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

"The editing instructions are shown in ***bold italic***. Four editing instructions are used: ***change***, ***delete***, ***insert***, and ***replace***. ***Change*** is used to make small corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed either by using ~~strikethrough~~ (to remove old material) or underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instructions. Replace is used to make large changes in existing text, subclauses, tables, or figures by removing existing material and replacing it with new material. Editorial notes will not be carried over into future editions."



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

## 2. Normative references

***Insert the following references at the appropriate locations in Clause 2:***

ANSI X9.62-Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)

FIPS 180-2, Secure Hash Standard (SHS), August 2002.

IETF RFC2104, HMAC: Keyed-Hashing for Message Authentication

ITU-T Recommendation X.509，Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks

ISO/IEC 15946 Information technology - Security techniques - Cryptographic techniques based on elliptic curves

ISO/IEC 10116: 1997 (2nd edition) Information technology - Security techniques - Modes of operation for an n-bit block cipher algorithm.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

## 3. Definitions

***Delete the definition "3.49 WEP."***

***Insert the following definitions in alphabetical order into Clause 3, renumbering as necessary:***

**3.63 authenticator entity (AE):** An entity that provides authentication action for the authentication supplicant before the supplicant accesses to the network. It resides in any STA.

**3.64 authentication and key management (AKM) suite:** A set of methods for authentication and key management.

**3.65 authentication service entity (ASE):** An entity that provides mutual identity authentication for the AE and the ASUE. It resides in any ASU.

**3.66 authentication service unit (ASU):** An important part of WAI authentication framework that is based on public-key cryptography. It manages the certificates and authenticates the identities of users etc.

**3.67 authentication supplicant entity (ASUE):** An entity that requires identity authentication through ASU. It resides in any STA.

**3.68 key encryption key (KEK):** A key used to encrypt the key data field in the key management protocol.

**3.69 message authentication key (MAK):** A key used to authenticate data source and check integrity in key management protocol.

**3.70 multicast session key (MSK):** A random value used to protect multicast MPDUs sent by the node. It is derived from the multicast master key.

**3.71 message integrity code (MIC):** A value generated by using a symmetric key cryptographic algorithm on the input data. If the input data are changed, a new value cannot be correctly computed without knowledge of the symmetric key.

**3.72 multicast master key (MMK):** An auxiliary key used to generate the multicast encryption key and the multicast integrity check key.

**3.73 multicast session key security association (MSKSA):** The result of the multicast key announcement process.

**3.74 preshared key (PSK):** A static key distributed to the STA. The way of the key distribution is beyond this specification.



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

**3.75 STAKey:** A symmetric key used to protect direct communications between two stations in a BSS.

**3.76 STAKey security association (STAKeySA):** The result of the STA-to-STA unicast key negotiation process in a BSS. A STAKeySA includes a STAKey.

**3.77 base key (BK):** An auxiliary key used to generate the unicast session key. It can be negotiated in the certificate authentication process or directly derived from PSK.

**3.78 base key security association (BKSA):** The result of the certificate authentication process or the result directly derived from PSK.

**3.79 unicast session key (USK):** A random value that is derived from the base key with a pseudo-random function. It is composed of four parts: the unicast encryption key, the unicast integrity check key, the message authentication key and the key encryption key.

**3.80 unicast session key security association (USKSA):** The result of the unicast key negotiation process.

**3.81 WAPI key management:** The key management including the unicast key negotiation, the multicast key announcement and the STAKey announcement.

**3.82 WAPI security network:** A network with the WAPI security mechanism. It is identified by the WAPI parameter set in management frame such as the Beacon frame.

**3.83 wireless local area network authentication and privacy infrastructure (WAPI):** A security scheme that is stated to provide identity authentication and data confidentiality for WLAN. WAPI comprises WAI (WLAN Authentication Infrastructure) and WPI (WLAN Privacy Infrastructure).

## 4. Abbreviations and acronyms

***delete the abbreviations "IV,ICV,WEP".***
***Insert the following abbreviations in alphabetical order into Clause 4:***

| | |
|---|---|
| ADDID | address index |
| AE | authenticator entity |
| AKM | authentication and key management |
| AKMP | authentication and key management protocol |
| ASE | authentication service entity |
| ASU | authentication service unit |
| ASUE | authentication supplicant entity |
| BK | base key |
| BKID | base key identification |
| BKSA | base key security association |



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

| | |
|---|---|
| ECDH | Diffie-Hellman exchange in elliptic curve cryptosystem |
| KEK | key encryption key |
| MAK | message authentication key |
| MIC | message integrity check code |
| MSK | multicast session key |
| MSKID | multicast session key index |
| MSKSA | multicast session key security association |
| NMK | notification master key |
| OUI | organization unique identifier |
| PSK | preshared key |
| STAKeyID | STAKey index |
| STAKeySA | STAKey security association |
| UCK | unicast integrity check key |
| UEK | unicast encryption key |
| USK | unicast session key |
| USKID | unicast session key index |
| USKSA | unicast session key security association |
| WAI | WLAN authentication infrastructure |
| WAPI | WLAN authentication and privacy infrastructure |
| WPI | WLAN privacy infrastructure |

# 5. General description

## 5.3 Logical service interfaces

***Change the text in 5.3 as follow:***

The complete set of this part architectural services are as follow:

a) Linkverification
~~a)~~ b) Authentication
~~b)~~ c) Association
~~c)~~ d) Delinkverification
~~d)~~ e) Disassociation
~~e)~~ f) Distribution
~~f)~~ g) Integration
~~g)~~ h) Privacy
~~h)~~ i) Reassociation
~~i)~~ j) MSDU delivery

This set of services is divided into two groups: those that are part of every STA, and those that are part of a DS.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

### 5.3.1 SS

***Change the text in 5.3.1 as follow:***

The SS is as follow:

a) Linkverification
~~a)~~b) Authentication
~~b)~~c) ~~Deauthentication~~ Delinkverification
~~c)~~d) Privacy
~~d)~~e) MSDU delivery

## 5.4 Overview of the services

***Change the first paragraph in 5.4 as follow:***

There are ~~nine~~ ten services (Linkverification, Authentication, Association, Delinkverification, Disassociation, Distribution, Integration, Privacy, Reassociation, and MSDU delivery) specified by ~~IEEE 802.11~~ this standard. Six of the services are used to support MSDU delivery between STAs. ~~Three~~ Four of the services are used to control wireless LAN access and confidentiality.

### 5.4.3 Access and confidentiality control services

***Change the text in 5.4.3 as follow:***

~~Two~~ Three services are required for this standard to provide functionality equivalent to that inherent to wired LANs. The design of wired LANs assumes the physical attributes of wire. In particular, wired LAN design assumes the physically closed and controlled nature of wired media. The physically open medium nature of a wireless LAN violates these assumptions.

~~Two~~ Three services are provided to bring the wireless LAN functionality in line with wired LAN assumptions; Linkverification, Authentication and privacy. Linkverification and Authentication ~~is~~ are used instead of the wired media physical connection. Privacy is used to provide the confidential aspects of closed wired media.

***Change 5.4.3.1 as follow:***

#### 5.4.3.1 ~~Authentication~~Linkverification

~~In wired LANs, physical security can be used to prevent unauthenticated access. This is impractical in wireless LANs due to the essence of wireless medium.~~

~~IEEE 802.11 provides the ability to control LAN access via the authentication service. This service is used by all stations to establish their identity to stations with which they will communicate. This is true for both ESS and~~



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

~~IBSS networks. If a mutually acceptable level of authentication has not been established between two stations, an association shall not be established. Authentication is an SS.~~

~~IEEE 802.11 supports several authentication processes. The IEEE 802.11 authentication mechanism also allows expansion of the supported authentication schemes. IEEE 802.11 does not mandate the use of any particular authentication scheme.~~

~~IEEE 802.11~~ISO/IEC 8802-11 provides link-level ~~authentication~~verification between ~~IEEE 802.11~~ISO/IEC 8802-11 STAs. ~~IEEE 802.11~~ISO/IEC 8802-11 does not provide either end-to-end (frame origin to frame destination) or user-to-user authentication. ~~IEEE 802.11 authentication is used simply to bring the wireless link up to the assumed physical standards of a wired link.~~ (This use of ~~authentication~~linkverification is independent of any authentication process that may be used in higher levels of a network protocol stack.) ~~If authentication other than that described here is desired, it is recommended that IEEE Std 802.10-1998 [B9]~~s be implemented.

~~If desired, a~~An ~~IEEE 802.11~~ISO/IEC 8802-11 network ~~may be~~ is operated using Open System ~~authentication~~linkverification ~~(see 8.1.1)~~. This may violate implicit assumptions made by higher network layers. In an Open System, any station may become ~~authenticated~~linkverified.

~~IEEE 802.11 also supports Shared Key authentication. Use of this authentication mechanism requires implementation of the wired equivalent privacy (WEP) option (see 8.2). In a Shared Key authentication system, identity is demonstrated by knowledge of a shared, secret, WEP encryption key.~~

Management information base (MIB) functions are provided to support the standardized ~~authentication~~linkverification schemes.

~~IEEE 802.11~~ISO/IEC 8802-11 requires mutually acceptable, successful, ~~authentication~~linkverification.

A STA may be ~~authenticated~~linkverified with many other STAs at any given instant.

#### 5.4.3.1.1 ~~Preauthentication~~Prelinkverification

Because the ~~authentication~~linkverification process could be time-consuming (depending on the ~~authentication~~linkverification protocol in use), the ~~authentication~~linkverification service can be invoked independently of the association service.

~~Preauthentication~~Prelinkverification is typically done by a STA while it is already associated with an AP (with which it previously ~~authenticated~~linkverified). ~~IEEE 802.11~~ISO/IEC 8802-11 does not require that STAs ~~preauthenticate~~prelinkverification with APs. However, ~~authentication~~linkverification is required before an association can be established.

If the ~~authentication~~linkverification is left until reassociation time, this may impact the speed with which a STA can reassociate between APs, limiting BSS-transition mobility performance. The use of ~~preauthentication~~prelinkverification takes the ~~authentication~~linkverification service overhead out of the time-critical reassociation process.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

***Change 5.4.3.2 as follow:***

### 5.4.3.2 ~~Deauthentication~~ Delinkverification

The ~~Deauthentication~~ Delinkverification service is invoked when an existing ~~authentication~~linkverification is to be terminated. ~~Deauthentication~~ Delinkverification is an SS.

In an ESS, since ~~Authentication~~ Linkverification is a prerequisite for association, the act of ~~Deauthentication~~ Delinkverification shall cause the station to be disassociated. The ~~Deauthentication~~ Delinkverification service may be invoked by either linkverified party (non-AP STA or AP). ~~Deauthentication~~ Delinkverification is not a request; it is a notification. ~~Deauthentication~~ Delinkverification shall not be refused by either party. When an AP transmits a ~~Deauthentication~~ Delinkverification notice to an associated STA, the association shall also be terminated.

In a WAPI ESS, Open System linkverification is required. In a WAPI ESS, delinkverification results in termination of any association for the delinkverified station. It also results in the WAI Controlled Port for that STA being disabled and deletes the unicast session key security association (USKSA). The delinkverification notification is provided to WAI via the MAC layer.

In WAPI, delinkverification also destroys any related USKSA, multicast session key security association (MSKSA), and STAKey security associations (STAKeySAs) that exist in the STA and closes the associated WAI Controlled Port. If base key (BK) caching is not enabled, delinkverification also destroys the base key security association (BKSA) from which the deleted USKSA was derived.

In a WAPI IBSS, Open System linkverification is optional, but a STA is required to recognize Delinkverification frames. Delinkverification results in the WAI Controlled Port for that STA being disabled and deletes the USKSA.

### 5.4.3.3 Privacy

***Change the text in 5.4.3.3 as follow:***

In a wired LAN, only those stations physically connected to the wire may hear LAN traffic. With a wireless shared medium, this is not the case. Any ~~IEEE 802.11~~ISO/IEC 8802-11-compliant STA may hear all like-PHY ~~IEEE 802.11~~ISO/IEC 8802-11 traffic that is within range. Thus the connection of a single wireless link (without privacy) to an existing wired LAN may seriously degrade the security level of the wired LAN.

To bring the functionality of the wireless LAN up to the level implicit in wired LAN design, ~~IEEE 802.11~~ this amendment provides the ability to encrypt the contents of messages. This functionality is provided by the privacy service. Privacy is an SS.

~~IEEE 802.11~~ This amendment specifies an optional privacy algorithm, ~~WEP~~WPI, that is designed to satisfy the



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

goal of wired LAN "equivalent" privacy. The algorithm is not designed for ultimate security but rather to be "at least as secure as a wire" See Clause 8 for more details.

~~IEEE 802.11~~ This amendment uses the ~~WEP~~WPI mechanism (see Clause 8) to perform the actual encryption of messages. MIB functions are provided to support ~~WEP~~WPI.

Note that privacy may only be invoked for data frames ~~and some Authentication Management frames~~. All stations initially start "in the clear" in order to set up the linkverification, authentication and privacy services.

The default privacy state for all ~~IEEE 802.11~~ this amendment STAs is "in the clear". If the privacy service is not invoked, all messages shall be sent unencrypted. If this default is not acceptable to one party or the other, data frames shall not be successfully communicated between the LLC entities. Unencrypted data frames received at a station configured for mandatory privacy, as well as encrypted data frames using a key not available at the receiving station, are discarded without an indication to LLC (or without indication to distribution services in the case of "To DS" frames received at an AP). These frames are acknowledged on the WM [if received without frame check sequence (FCS) error] to avoid wasting WM bandwidth on retries.

***After 5.4.3.3, insert clause 5.4.3.4 as follow:***

### 5.4.3.4 Authentication

This part supports WAI (WLAN Authentication Infrastructure), which is used to implement mutual authentication between two STAs; it is established after linkverification and association. In BSS, when the authentication is successful, the STA can securely access to the AP; otherwise, both the AP and STA refuse to connect with each other. In IBSS, when the authentication is successful, the STA can securely access to the peer STA; otherwise, both STAs refuse to connect with each other. Detailed definition is in chapter 8.

In BSS or IBSS, WAI supports authentication based on certificate, or preshared keys. Certificate authentication utilizes the public key certificates to authenticate STAs and the AP.

In WAPI, WAI ASUE and AE exchange protocol information via the WAI Uncontrolled Port. The WAI Controlled Port is blocked from passing general data traffic between two STAs until a WAI authentication procedure completes successfully over the WAI Uncontrolled Port.

Authentication of this part is only to make that wireless link have the assumed physical standard of wired link.

#### 5.4.3.4.1 Preauthentication

Because the authentication process could be time-consuming (depending on the authentication protocol in use), the authentication service can be invoked independently of the association service.

Preauthentication is typically done by a STA while it is already associated with an AP (with which it previously authenticated). This amendment does not require that STAs preauthenticate with APs. However, authentication is required before an association can be established.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If the authentication is left until reassociation time, this may impact the speed with which a STA can reassociate between APs, limiting BSS-transition mobility performance. The use of preauthentication takes the authentication service overhead out of the time-critical reassociation process.

### 5.7.5 Privacy

***Change the text in 5.7.5 as follow:***

For a STA to invoke the ~~WEP~~ WPI privacy algorithm (as controlled by the related MIB attributes, see Clause 11), the privacy service causes MAC protocol data unit (MPDU) encryption and sets the ~~WEP~~ WPI frame header bit appropriately (see Clause 7).

***Change 5.7.6 as follow:***

### 5.7.6 ~~Authentication~~ Linkverification

For a STA to ~~authenticate~~linkverify with another STA, the ~~authentication~~linkverification service causes one or more ~~authentication~~ linkverification management frames to be exchanged. The exact sequence of frames and their content is dependent on the ~~authentication~~ linkverification scheme invoked. ~~For all authentication schemes, the authentication~~The linkverification algorithm is identified within the management frame body.

In an IBSS environment, either station may be the initiating STA (STA 1). In an ESS environment, STA 1 is the mobile STA, and STA 2 is the AP.

This amendment defines one subtype of linkverification service: *Open System*. The subtype invoked is indicated in the body of linkverification management frames. Thus linkverification frames are selfidentifying with respect to linkverification algorithm. All management frames of subtype linkverification shall be unicast frames as linkverification is performed between pairs of stations (i.e., multicast linkverification is not allowed). Management frames of subtype Delinkverification are advisory, and may therefore be sent as group-addressed frames. A mutual linkverification relationship shall exist between two stations following a successful linkverification exchange as described below. Linkverification shall be used between stations and the AP in an infrastructure BSS. Linkverification may be used between two STAs in an IBSS.

~~*Authentication (first frame of sequence)*~~

~~— Frame type: Management~~
~~— Frame subtype: Authentication~~
~~— Information items:~~
~~♦ Authentication algorithm identification~~
~~♦ Station identity assertion~~
~~♦ Authentication transaction sequence number~~
~~♦ Authentication algorithm dependent information~~
~~— Direction of frame: First frame in the transaction sequence is always from STA 1 to STA 2.~~



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

~~*Authentication (intermediate sequence frames)*~~

- ~~Frame type: Management~~
- ~~Frame subtype: Authentication~~
- ~~Information items:~~
  - ~~Authentication algorithm identification~~
  - ~~Authentication transaction sequence number~~
  - ~~Authentication algorithm dependent information~~
- ~~Direction of frame:~~
  - ~~Even transaction sequence numbers: From STA 2 to STA 1~~
  - ~~Odd transaction sequence numbers: From STA 1 to STA 2~~

~~*Authentication (final frame of sequence)*~~

- ~~Frame type: Management~~
- ~~Frame subtype: Authentication~~
- ~~Information items:~~
  - ~~Authentication algorithm identification~~
  - ~~Authentication transaction sequence number~~
  - ~~Authentication algorithm dependent information~~
  - ~~The result of the requested authentication. This is an item with values successful and unsuccessful.~~
- ~~Direction of frame: From STA 2 to STA 1~~

Open System linkverification is the simplest of the available linkverification algorithms. Essentially it is a null linkverification algorithm. Any STA that requires linkverification with this algorithm may become linkverified if dot11Linkverification Type at the recipient station is set to Open System linkverification. Open System linkverification is not required to be successful as a STA may decline to linkverification with any particular other STA. Open System linkverification is the default linkverification algorithm. Open System linkverification involves a two-step linkverification transaction sequence. The first step in the sequence is the identity assertion and request for linkverification. The second step in the sequence is the linkverification result. If the result is successful, the STAs shall be mutually linkverified.

Open System linkverification (first frame)

- Message type: Management
- Message subtype: linkverification
- Information items:
  - Linkverification Algorithm Identification = “Open System”.
  - Station Identity Assertion (in SA field of header)
  - Linkverification transaction sequence number = 1
  - Linkverification algorithm dependent information (none)
- Direction of message: From linkverification initiating STA to linkverifying STA

Open System linkverification (final frame)



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

— Message type: Management

—Message subtype: Linkverification

—Information items:

- Linkverification Algorithm Identification = “Open System”.
- Linkverification transaction sequence number = 2
- Linkverification algorithm dependent information (none)
- The result of the requested linkverification as defined in 7.3.1.9

— Direction of message: From linkverifying STA to initiating STA

If dot11 Linkverification Type does not include the value “Open System”, the result code shall not take the value “successful”.

### 5.7.7 ~~Deauthentication~~ Delinkverification

***Change the text in 5.7.7 as follow:***

For a STA to invalidate an active ~~authentication~~ linkverification, the following frame is transmitted:

*~~Deauthentication~~ Delinkverification*

- Frame type: Management
- Frame subtype: ~~Deauthentication~~ Delinkverification
- Information items:
  - IEEE address of the STA that is being ~~deauthenticated~~ delinkverified
  - IEEE address of the STA with which the STA is currently ~~authenticated~~ linkverified
  - This shall be the broadcast address in the case of a STA ~~deauthenticating~~ delinkverifying all STAs currently ~~authenticated~~ linkverified.
- Direction of frame: From STA to STA

## 5.8 Reference model

***Change the Figure 11 as follows:***

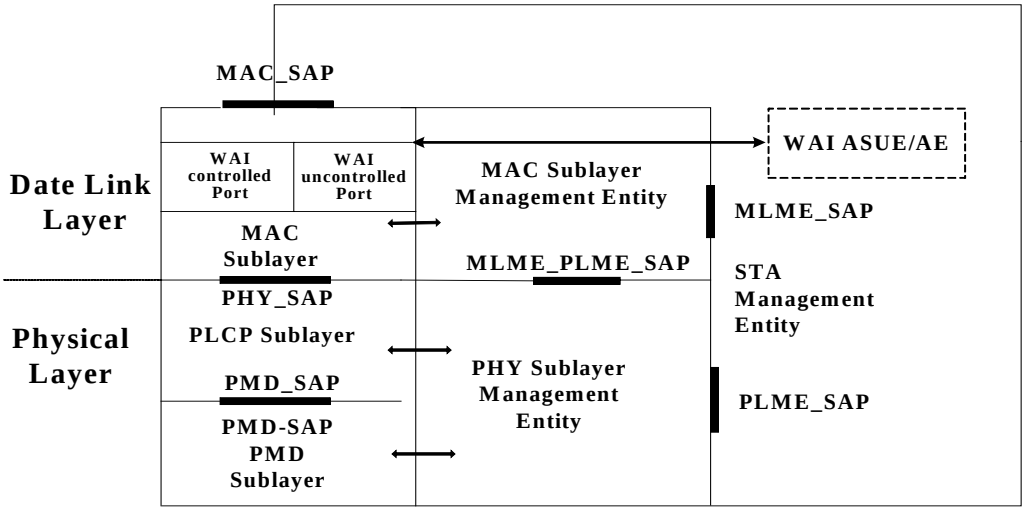



**Figure 11—Portion of the ISO/IEC basic reference model covered in this amendment**



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

*After 5.8, insert 5.9 and renumber the figures as necessary:*

## 5.9 Establishing the security association

### 5.9.1 Infrastructure mode

The STA obtains the security policy of the AP by passively monitoring the Beacon frame or probing the AP on its own initiative (Figure 11a). If the WAI certificate Authentication and Key Management Protocol(AKMP) is applied, AP sends an authentication activation packet to start the certificate authentication process. After this process, the unicast key negotiation process and the multicast key announcement process are executed between AP and STA. If the PSK mechanism is applied, the unicast key negotiation process and the multicast key announcement process are directly executed between AP and STA.

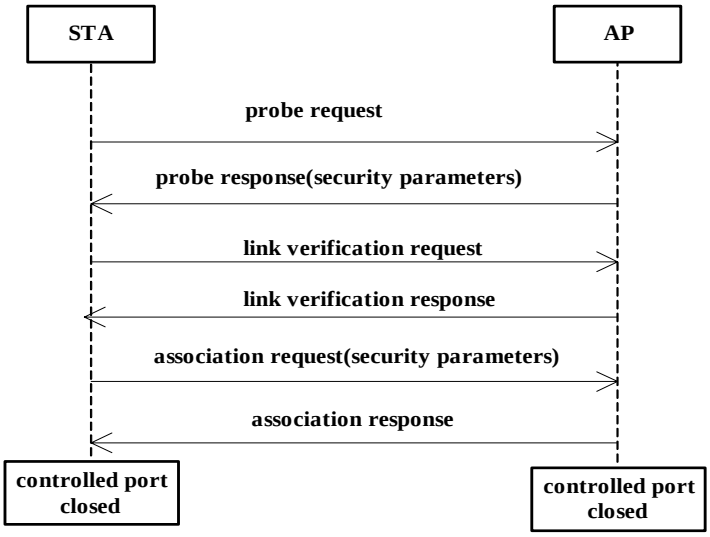



**Figure 11a—The establishment of security association in infrastructure mode**

### 5.9.2 IBSS mode

The initiation of the PSK key negotiation process in IBSS :

In an IBSS, there are two five-step handshakes (the first three steps are the unicast key negotiation process and the last two are the multicast key announcement process) between two STAs. Two independent unicast keys will be negotiated. And their own multicast key will be announced to each other. In the course of practical communication, for a pair of STAs, the unicast data between the two STAs are encrypted or decrypted with the unicast encryption key that is generated by the process initiated by the STA (as AE) with the larger MAC address, whereas the keys generated by the process initiated by the STA (as AE) with the smaller MAC address will not be used. Every STA encrypts its broadcast/multicast data with the multicast key announced by itself before sending them and the other STAs decrypt these data with the same key when receiving them.

The reasons or conditions for the initiation of the unicast key negotiation process based on PSK are listed as below:



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) When a STA receives a Beacon frame or a Probe Response frame from another STA which has not completed a unicast key negotiation process with it.
b) When a STA receives the first protocol packet of the unicast key negotiation process from another STA which has not completed a process of unicast key negotiation with it.
c) When a STA receives a WPI encapsulation broadcast/multicast data, if it has not the corresponding decryption key, it shall notify the station management entity with an MLME-PROTECTEDFRAMEDROPPED.indication primitive. If the entity receives this primitive and has not completed a unicast key negotiation with the peer STA specified in the primitive.

The initiation of authentication based on certificate in IBSS:

When the authentication method based on certificate is chosen between two STAs, each STA initiates its own certificate authentication process, unicast key negotiation process and multicast key announcement process, respectively. After these processes, they establish two independent BKs as well as two independent USKs and announce their own multicast keys to each other. In the course of data communication, for a pair of STAs, the unicast data between the two STAs are encrypted or decrypted with the unicast encryption key that are generated from the process of unicast key negotiation initiated by the STA (as AE) with the larger MAC address, whereas the keys generated by the process initiated by the STA (as AE) with the smaller MAC address will not be used. Every STA encrypts its broadcast/multicast data with the multicast key announced by itself when sending them and the other STAs decrypt these data with the same key when receiving them.

The reasons or conditions of the initiation of the authentication process based on certificate are listed as below:

a) When a STA receives a Beacon frame or a Probe Response frame from another STA which has not completed a certificate authentication process with it.
b) When a STA receives the first protocol packet of the certificate authentication process from another STA which has not completed a process of certificate authentication with it.
c) When a STA receives a WPI encapsulation broadcast/multicast data, if it has not the corresponding decryption key, it shall notify the station management entity with an MLME-PROTECTEDFRAMEDROPPED.indication primitive. If the entity receives this primitive and has not completed a certificate authentication with the STA described in the primitive.

If the WAPI security parameters of the peer STA are unknown to another STA, the STA should send a Probe Request frame to the peer STA before the initiation of unicast key negotiation process. Then the security mechanism can be obtained from the Probe Response frame of the peer STA.

# 6. MAC service definition

## 6.1 Overview of MAC services

### 6.1.2 Security services

***Change the text in 6.1.2 as follows:***



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Security services in this standard are provided by the authentication service, linkverification, and the ~~WEP~~ WPI mechanism. The scope of the security services provided is limited to station-to-station data exchange. The privacy service offered by ~~an IEEE 802.11 WEP~~ this part implementation is the encryption of the MPDU. For the purposes of this standard, ~~WEP~~ WPI is viewed as a logical service located within the MAC sublayer as follows in the reference model, Figure 11 (in 5.8). Actual implementations of the ~~WEP~~ WPI service are transparent to the LLC and other layers above the MAC sublayer.

The security services provided in ~~IEEE 802~~ this part are as follows:

a) Confidentiality;
b) Linkverification;
~~b)~~ c) Authentication; and
~~c)~~ d) Access control in conjunction with layer management.

During the authentication exchange, parties A and B exchange authentication information as described in Clause 8.

The MAC sublayer security services provided by ~~WEP~~ WPI rely on information from nonlayer-2 management or system entities. Management entities communicate information to ~~WEP~~ WPI through a set of MIB attributes

***End of changes to Clause 6.***

# 7. Frame formats

## 7.1 MAC frame formats

### 7.1.3 Frame fields

#### 7.1.3.1 Frame Control field

***Change the text of 7.1.3.1 as follows:***

The Frame Control field consists of the following subfields: Protocol Version, Type, Subtype, To DS, From DS, More Fragments, Retry, Power Management, More Data, ~~WEP~~ Encryption, and Order. The format of the Frame Control field is illustrated in Figure 13.

***Change Figure 13 as shown:***

| B0 B1 | B2 B3 | B4 B7 | B8 | B9 | B10 | B11 | B12 | B13 | B14 | B15 |
|---|---|---|---|---|---|---|---|---|---|---|
| Protocol Version | Type | Subtype | To DS | From DS | More Frag | Retry | Pwr Mgt | More Data | ~~WEP~~ Encryption | Order |
| Bits: 2 | 2 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 13—Frame Control field**



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

#### 7.1.3.1.2 Type and Subtype fields

***Change the item of Table 1 as follows:***

**Table 1—Valid type and subtype combinations**

| Type value b3 b2 | Type description | Subtype value b7 b6 b5 b4 | Subtype description |
|---|---|---|---|
| 00 | Management | 1011 | ~~Authentication~~ Linkverification |
| 00 | Management | 1100 | ~~Deauthentication~~ Delinkverification |

***Change the title and text of 7.1.3.1.9 as follows:***

#### 7.1.3.1.9 ~~WEP~~Protected Frame field

The ~~WEP~~ Encryption field is 1 bit in length. ~~It~~ The Encryption field is set to 1 if the Frame Body field contains information that has been processed by ~~the WEP~~ a cryptographic encapsulation algorithm. The ~~WEP~~ Encryption field is set to 1 only within data frames ~~of type Data and frames of type Management, subtype Authentication~~. The ~~WEP~~ Encryption field is set to 0 in all other frames. ~~When the WEP bit field is set to 1, the Frame Body field is expanded as defined in 8.2.5,~~. In a data frame, the Frame Body field is protected utilizing the cryptographic encapsulation algorithm and expanded as defined in Clause 8.

## 7.2 Format of individual frame types

### 7.2.2 Data frames

***Change the first paragraph after the lettered list in 7.2.2 as follows:***

The frame body consists of the MSDU or a fragment thereof, and a ~~WEP IV and ICV~~ security header and a trailer (if and only if the ~~WEP~~ Encryption subfield in the Frame Control field is set to 1). The frame body is null (0 octets in length) in data frames of subtypes such as Null Function (no data), CF-Ack (no data), CF-Poll (no data), and CF-Ack+CF-Poll (no data).

### 7.2.3 Management frames

#### 7.2.3.1 Beacon frame format

***Insert the order 21 information field in Table 5:***

**Table 5—Beacon frame body**

| | | |
|---|---|---|
| 21 | WAPI Parameter Set | The WAPI Parameter Set information element is only present within Beacon frames generated by STAs when the WAPI mechanism is enabled. |

#### 7.2.3.4 Association Request frame format



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

***Insert the order 8 information field in Table 7:***

**Table 7—Association Request frame body**

| | | |
|---|---|---|
| 8 | WAPI Parameter Set | The WAPI Parameter Set information element is only present within Association request frames generated by STAs in which the WAPI mechanism is enabled. |

### 7.2.3.6 Reassociation Request frame format

***Insert the order 9 information field to Table 9:***

**Table 9—Reassociation Request frame body**

| | | |
|---|---|---|
| 9 | WAPI Parameter Set | The WAPI Parameter Set information element is only present within Reassociation request frames generated by STAs in which the WAPI mechanism is enabled. |

### 7.2.3.9 Probe Response frame format

***Insert the order 21 and 22–n information fields in Table 12:***

**Table 12—Probe Response frame body**

| | | |
|---|---|---|
| 21 | WAPI Parameter Set | The WAPI Parameter Set information element is only present within Probe Response frames generated by STAs that has enabled the WAPI mechanism. |
| 22-n | Requested information elements | Elements requested by the Request information element of the Probe Request frame. |

### 7.2.3.10 ~~Authentication~~ Linkverification frame format

***Change the text in 7.2.3.10 and Table 13 as follows:***

The frame body of a management frame of subtype ~~Authentication~~ Linkverification contains the information shown in Table 13. Only Linkverification frames with the Linkverification algorithm set to Open System Linkverification may be used within a WAPI network.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

**Table 13—~~Authentication~~ Linkverification frame body**

| Order | Information | Notes |
|---|---|---|
| 1 | ~~Authentication~~ Linkverification algorithm number | |
| 2 | ~~Authentication~~ Linkverification transaction sequence number | |
| 3 | Status code | The status code information is reserved and set to 0 in certain ~~Authentication~~ Linkverification frames ~~as defined in Table 14~~. |

#### 7.2.3.11 ~~Deauthentication~~ Delinkverification

***Change the text in 7.2.3.11 and Table 15 as follows:***

The frame body of a management frame of subtype ~~Deauthentication~~ Delinkverification contains the information shown in Table 15.

**Table 15—~~Deauthentication~~ Delinkverification frame body**

| Order | Information |
|---|---|
| 1 | Reason code |

## 7.3 Management frame body components

### 7.3.1 Fixed fields

#### 7.3.1.1 ~~Authentication~~ Linkverification Algorithm Number field

***Change the text in 7.3.1.1 and Figure 24 as follows:***

The ~~Authentication~~ Linkverification Algorithm Number field indicates a single ~~authentication~~ linkverification algorithm. The length of the ~~Authentication~~ Linkverification Algorithm Number field is 2 octets. The ~~Authentication~~ Linkverification Algorithm Number field is illustrated in Figure 24. The following values are defined for ~~Authentication~~ Linkverification algorithm number:

~~Authentication~~ Linkverification algorithm number = 0: Open System
~~Authentication algorithm number = 1: Shared Key~~
All other values of ~~authentication~~ Linkverification number are reserved.



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

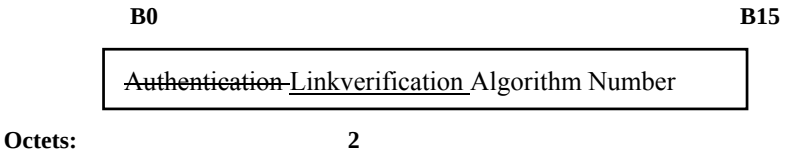



**Figure 24—~~Authentication~~ Linkverification Algorithm Number fixed field**

### 7.3.1.2 ~~Authentication~~ Linkverification Transaction Sequence Number field

***Change the text in 7.3.1.2 and Figure 25 as follows:***

The ~~Authentication~~ Linkverification Transaction Sequence Number field indicates the current state of progress through a multistep transaction. The length of the ~~Authentication~~ Linkverification Transaction Sequence Number field is 2 octets. The ~~Authentication~~ Linkverification Transaction Sequence Number field is illustrated in Figure 25.

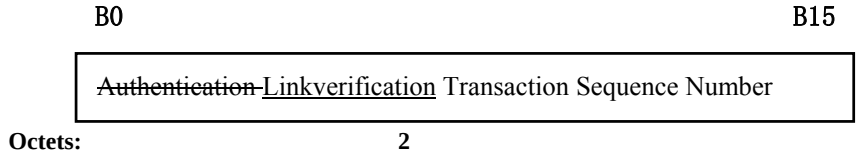



**Figure 25—~~Authentication~~ Linkverification Transaction Sequence Number fixed field**

### 7.3.1.4 Capability Information field

***Change the sixth and seventh paragraphs in 7.3.1.4 as follows:***

APs set the Privacy subfield to 1 within transmitted Beacon, Probe Response, Association Response, and Reassociation Response management frames if ~~WEP encryption~~ data confidentiality is required for all data type frames exchanged within the BSS. If ~~WEP encryption~~ data confidentiality is not required, ~~the Privacy subfield is set to 0~~ APs set the Privacy subfield to 0 within these management frames.

In a WAPI secure network, non-AP STAs in an ESS set the Privacy subfield to 0 within transmitted Association and Reassociation Request management frames. APs ignore the Privacy subfield within received Association and Reassociation Request management frames.

STAs within an IBSS set the Privacy subfield to 1 in transmitted Beacon or Probe Response management frames if ~~WEP encryption~~ data confidentiality is required for all data type frames exchanged within the IBSS. If ~~WEP encryption~~ data confidentiality is not required, STAs in an IBSS set the Privacy subfield ~~is set~~ to 0 within these management frames.

STAs that include the WAPI information element in Beacon and Probe Response frames shall set the Privacy subfield to 1 in any frame that includes the WAPI information element.



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

### 7.3.1.7 Reason Code field

***Change the Table 18 as follows:***

**Table 18—Reason codes**

| Reason code | Meaning |
|---|---|
| 0 | Reserved |
| 1 | Unspecified reason |
| 2 | Previous ~~authentica~~tion Linkverification no longer valid |
| 3 | ~~Deauthenticate~~d Delinkverification because transmitting station is leaving (or has left) IBSS or ESS |
| 4 | Disassociated due to inactivity |
| 5 | Disassociated because AP is unable to handle all currently associated stations |
| 6 | Class 2 frame received from ~~nonauthenticate~~d non-linkverified station |
| 7 | Class 3 frame received from non-associated station |
| 8 | Disassociated because transmitting station is leaving (or has left) BSS |
| 9 | Station requesting (re)association is not ~~authenticate~~d linkverified with responding station |
| 10-~~65 535~~12 | Reserved |
| 13 | Invalid information element |
| 14 | MIC failure |
| 15-19 | Reserved |
| 20 | Invalid AKMP |
| 21-23 | Reserved |
| 24 | Cipher suite rejected per security policy |
| 25 | Unicast Key Negotiation timeout |
| 26 | Multicast Key Announcement timeout |
| 27 | Information element in Unicast Key Negotiation different from (Re)Association Request/Probe Response/Beacon frame |
| 28 | Invalid Multicast cipher |
| 29 | Invalid Unicast cipher |
| 30 | Unsupported WAPI information element version |
| 31 | Invalid WAPI information element capabilities |
| 32 | WAI certificate authentication failed |
| 33-65 535 | Reserved |

### 7.3.1.9 Status Code field

***Change the Table 19 into the following table:***



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

**Table 19—Status codes**

| Status codes | Meaning |
|---|---|
| 0 | Successful |
| 1 | Unspecified failure |
| 2-9 | Reserved |
| 10 | Cannot support all requested capabilities in the Capability Information field |
| 11 | Reassociation denied due to inability to confirm that association exists |
| 12 | Association denied due to reason outside the scope of this standard |
| 13 | Responding station does not support the specified ~~authentication~~ Linkverification algorithm |
| 14 | Received an ~~authentication~~ Linkverification frame with ~~authentication~~ Linkverification transaction sequence number out of expected sequence |
| 16 | ~~Authentication~~ Linkverification rejected due to timeout waiting for next frame in sequence |
| 17 | Association denied because AP is unable to handle additional associated stations |
| 18 | Association denied due to requesting station not supporting all of the data rates in the BSSBasicRateSet parameter |
| 19 | Association denied due to requesting station not supporting the Short Preamble option |
| 20 | Association denied due to requesting station not supporting the PBCC Modulation option |
| 21 | Association denied due to requesting station not supporting the Channel Agility option |
| 22-~~6553539~~ | Reserved |
| 40 | Invalid information element |
| 41-42 | Reserved |
| 43 | Invalid AKMP |
| 44-45 | Reserved |
| 46 | Cipher suite rejected per security policy |
| 47 | Invalid multicast cipher |
| 48 | Invalid unicast cipher |
| 49 | Unsupported WAPI information element version |
| 50 | Invalid WAPI information element capabilities |
| 51-65535 | Reserved |

### 7.3.2 Information element

***Replace element identifiers (IDs) 33through 39 in Table 20 as follows:***



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

**Table 20—Element IDs**

| Information element | Element ID |
|---|---|
| Reserved | 60-67 |
| WAPI | 68 |
| Reserved | 69 |

***After 7.3.2.24, insert 7.3.2.25 through 7.3.2.25.5 and renumber tables and figures as necessary:***

### 7.3.2.25 WAPI information element

The WAPI Parameter Set information element contains authentication and unicast cipher suite selectors, a single multicast cipher suite selector, a WAPI Capabilities field, BK identifier (BKID) count, and BKID list. See Figure 42a. All STAs implementing WAPI shall support this element. The size of the WAPI information element is limited by the size of an information element, which is 255 octets. Therefore, the number of unicast cipher suites, AKM suites, and BKIDs is limited.

| | Element Identifier ID | Length | Version | Authentication And key Management (AKM) Suite Count | Authentication And key Management (AKM) Suite | Unicast Cipher Suite count | Unicast Cipher Suite | Multicast Cipher Suite | WAPI Capability information | BKID count | BKID List |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Octets**: | 1 | 1 | 2 | 2 | 4×m | 2 | 4×n | 4 | 2 | 2 | 16×s |

**Figure 42a—WAPI Information Element format**

In Figure 42a, m denotes AKM suite count, n is the unicast cipher suite count, and s is the BKID count.

— Element Identifier ID should be 68 decimal.
— The length filed gives the number of octets of all the fields except the element identifier ID field and the length field of the information element.
— The version field gives the version number of the WAPI protocol. In this document, this field is set to 1, and other values are reserved.
— The authentication and key management (AKM) suite count field denotes the number of the authentication and key management mechanisms that STA supports.
— The authentication and key management (AKM) suite field contains the authentication and key management mechanisms that STA supports. Here, m is the value of the authentication and key management suite count field.
— The unicast cipher suite count field denotes the number of the unicast cipher algorithms that STA supports.
— The unicast cipher suite field contains the unicast cipher algorithms that STA supports. Here, n is the value of the unicast cipher suite count field.
— The multicast cipher suite field contains the multicast cipher algorithms that STA supports.
— WAPI capability information.

The WAPI capability information field indicates requested or advertised capabilities. Every subfield of the WAPI



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

capability information field can be set to 0. The WAPI capability information field is 2 octets in length, and its subfields are defined as follows:




**Figure 42b—Pre-authentication**

Bit 0: Pre-Authentication. An AP sets the Pre-Authentication subfield of the WAPI Capabilities field to 1 when it supports pre-authentication. Otherwise, it will be set to 0. A non-AP STA sets the Pre-Authentication subfield to 0.

— The BKID count and list fields

The BKID count and list fields is only present within Association or Reassociation Request frames sent to AP. The BKID count field indicates the number of BKIDs in the BKID list field. The BKID list field contains 0 or more valid BKIDs that the STA believes to be valid for the destination AP. The BKIDs can refer to:

a) BKIDs generated by pre-authentication between the current STA and the destination AP
b) BKIDs generated by the certificate authentication
c) BKIDs generated by the preshared key

### 7.3.2.25.1 Authentication and Key Management Suite

A suite selector has the format shown in Figure 42c.

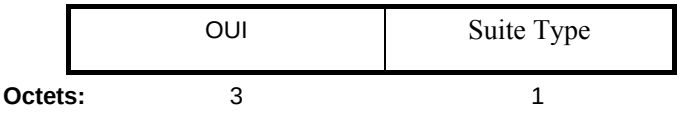



**Figure 42c—Suite selector format**

The suites defined in this specification are shown in Table 20f.

**Table 20f—Authentication and Key Management Suites**

| **OUI**<br>**3** octets | **Suite Type**<br>**1** octet | **Meaning** |
|---|---|---|
| 00-14-72 | 0 | Reserved |
| 00-14-72 | 1 | WAI Certificate Authentication and Key Management |
| 00-14-72 | 2 | WAI Preshared Key Authentication and Key Management |
| 00-14-72 | 3-255 | Reserved |
| Others | 0-255 | Reserved |



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

### 7.3.2.25.2 Unicast and Multicast Cipher Suites

The suite selector format is shown in Figure 42d.

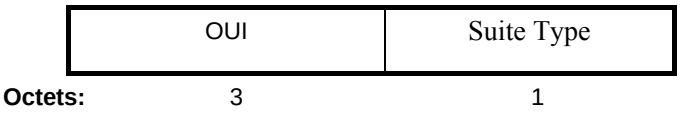



**Figure 42d—Suite selector format**

Note: Cryptographic algorithms to be applied in information security mechanism may be subject to national and regional regulations. An appropriate algorithm shall be adopted here, and which type of algorithm to be adopted is optional. It shall conform to national laws and regulations, and can be chosen according to specific requirements in different countries and regions.

***End of changes to Clause 7.***

***Replace Clause 8 in its entirety with the following text:***

# 8 Security

WAPI mechanism is adopted to implement security access control and data confidentiality. WAPI security is made up of two parts:

— WAI authentication and key management;

— WPI data transport protection.

## 8.1 WAI authentication and key management

The STA supporting the WAI authentication and key management can be implemented by the four ways below:

a) Based on certificate in BSS
   1) STA identifies the WAI authentication and key management suites supported by AP through passively monitoring Beacon frames or through actively probing ( Probe Response frames).
   2) Linkverification is executed between STA and AP.
   3) During (re)association, STA determines the cipher suite by containing the WAPI parameter in the (Re)Association request frame.
   4) STA and AP execute the certificate authentication process to negotiate the BK.
   5) STA and AP execute the unicast key negotiation process and multicast key announcement process.
   6) WPI module is informed of the negotiated keys and cipher suites, and then uses them to protect data.

b) Based on PSK(between STA and AP)in BSS
   1) STA identifies the WAI authentication and key management suites supported by AP through passively monitoring Beacon frame or through actively probing ( Probe Response frame).
   2) Linkverification is executed between STA and AP.
   3) During (re)association, STA determines the cipher suite by containing the WAPI parameter in the



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Re)Association request frame.

4) STA and AP will execute the unicast key negotiation process and multicast key announcement process after BK is derived from PSK .
5) WPI module is informed of the negotiated keys and cipher suites, and then uses them to protect data.

c) Based on certificate in IBSS
   1) STA identifies the WAI authentication and key management suite supported by the peer STA through passively monitoring Beacon frame or through actively probing ( Probe Response frame).
   2) An optional Linkverification is executed between STA and the peer STA.
   3) STA and the peer STA execute the certificate authentication process to negotiate BK.
   4) STA and the peer STA execute the unicast key negotiation process and the multicast key announcement process.
   5) WPI module is informed of the negotiated keys and cipher suites, and then uses them to protect data.

d) Based on PSK in the IBSS
   1) STA identifies the WAI authentication and key management suite supported by the peer STA through passively monitoring Beacon frame or through actively probing ( Probe Response frame).
   2) An optional Linkverification is executed between STA and the peer STA.
   3) STA and the peer STA will execute the unicast key negotiation process and the multicast key announcement process after BK is derived from PSK.
   4) WPI module is informed of the negotiated keys and cipher suites, and then uses them to protect data.

If WAPI security mechanism is negotiated to enable during (re)association between STA and AP /STA, the mutual identify authentication and key negotiation must be executed. Based on certificate, the authentication procedure should include the certificate authentication, unicast key negotiation and multicast key announcement. Based on PSK, the authentication procedure should include the unicast key negotiation and multicast key announcement.

The authentication packets between STA and AP /STA are transported by the WAPI protocol with Ether Type 0x88B4, and those packets between AP/STA and ASU are transported by the UDP socket with port number 3810.

A security association starts when the STA's SME decides to establish a WAPI security network and completes when the MLME-SETPROTECTION.Request primitive is invoked. The time that the security association takes to set up must be less than the MIB variable dot11WAPIConfigSATimeout. If the time is greater than the MIB variable dot11WAPIConfigSATimeout, DeLinkverification frame will be sent.

### 8.1.1 The structure of the authentication system

#### 8.1.1.1 Systems and Ports

STA provides two kinds of logic channels to access LAN. They are defined as two kinds of ports, the controlled



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

port and the uncontrolled port.

STA A provides a port, which is an uncontrolled port, for other STAs to connect to the ASU. STA A also provides the controlled port. It must only be used by those STAs achieving a successful authentication so that they can access network or send/receive data. In the operation of access control based on Port, the following three entities are defined:

a) AE (Authenticator Entity): An entity that provides authentication action for the authentication supplicant before the supplicant access to the service. It resides in any STA.
b) ASUE (Authentication Supplicant Entity): An entity that requests identity authentication through any ASE (authentication service entity). It resides in STA.
c) ASE (Authentication Service Entity): An entity that provides mutual identity authentication for the AE and the ASUE. It resides any ASU (Authentication service unit).

Both AE and ASUE can be called authentication subsystems.

### 8.1.1.2 Controlled and Uncontrolled access

The figure below is a sketch map of an authentication subsystem of access control based on Port.
Through the uncontrolled port, authentication data (WAPI protocol data) can be transported in WLAN. This transmission is not determined by the current authentication state. For the controlled port, the general protocol data (except WAPI protocol data) can only be passed through when the authentication state of this port is authenticated. The controlled port and the uncontrolled may be two logic ports that are connected to the same physical port. All the data through the physical port can reach both of them. Whether or not the data can pass is determined by the current authentication state.

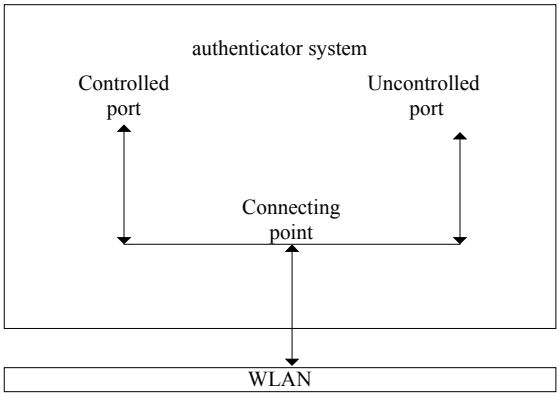



**Figure 43a—Uncontrolled and controlled ports**

Figure 43b shows two different authentication states associated with the controlled port, representing the status as a switch that can be turned on or off. When the ‘switch’ is turned off, the controlled Port is unauthorized and is therefore disabled and prevents the flow of PDUs; When the ‘switch’ is turned on, the controlled port is authorized and is therefore enabled and allows the flow of PDUs. There are two subsystems in Figure 43b. In authentication subsystem 1, the authentication state of the controlled port is unauthenticated and no data is



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

permitted to across this port. In the subsystem 2, the state is authenticated and PDUs are permitted to across the port.

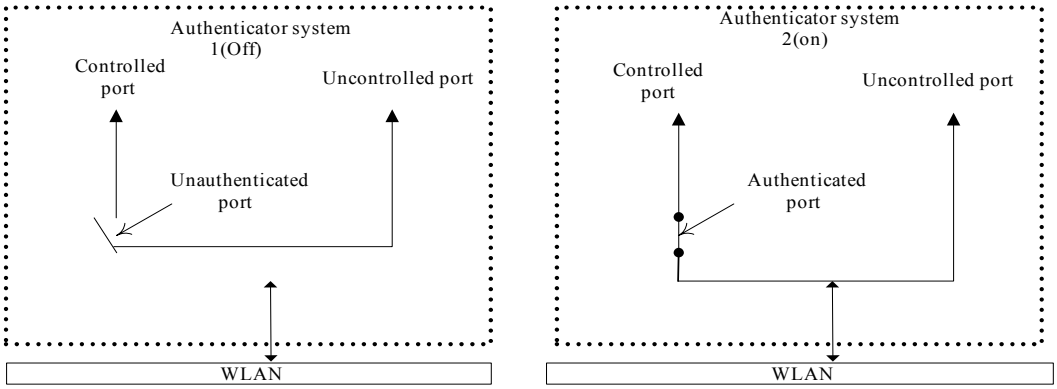



**Figure 43b—Authentication state on controlled port**

The state of any controlled port of the system is determined by the authentication control parameter of the system. The value of the system control parameter may be “authentication disabled” or “authentication enabled”. When the system control parameter is set to “authentication disabled”, the state of all the controlled ports is “authenticated”. When it is set to “authentication enabled”, the authentication state of every controlled port of the system is determined by the following two types of authentication control style:

a) ForceUnauthenticated: the state of a certain controlled port is forced to be unauthenticated. That is, the state of a controlled port is unconditionally set to unauthenticated (i.e. no data is permitted to transit this controlled port)
b) Auto: the state of a controlled port is set according to the result of the mutual authentication between the AE and the ASUE (i.e. the data can pass through the controlled port only if the authentication is successful.).

In the system, the network protocol data except the authentication packet between STAs is tranmitted out through the controlled ports. Figure 43c gives the logic structure of the controlled port and the uncontrolled port. The authentication state of the controlled port in the system is set by AE according to the result of the authentication exchange.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

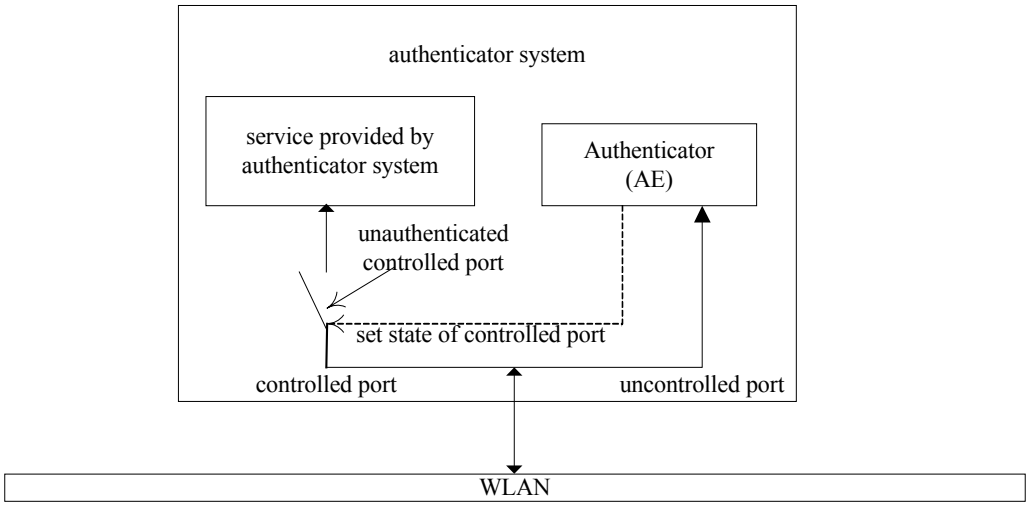



**Figure 43c—Usage of the controlled and uncontrolled port**

Figure 43d shows the relations and the information exchange process among ASUE, AE and ASE. In this figure, the state of the controlled port in AE is unauthenticated, which makes AE system refuse to provide services.

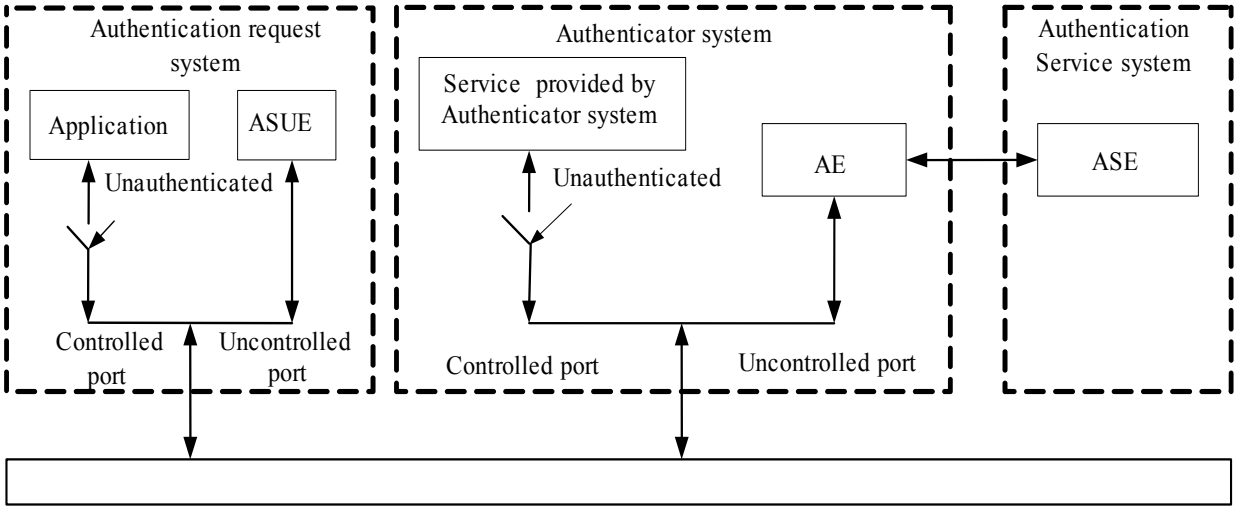



**Figure 43d— ASUE, AE and ASE roles**

### 8.1.2 WAPI security association management

#### 8.1.2.1 WAPI security association definitions

A security association is a set of policy(ies) and key(s) used to protect information. There are 4 kinds of security associations in WAPI:

— BKSA: The base key security association is from the PSK information or the result of a successful WAI



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

certificate authentication exchange.

— USKSA: The unicast session key security association is the result of a successful unicast key negotiation handshake.

— MSKSA: The multicast session key security association is the result of a successful multicast key announcement handshake.

— STAKeySA: STAKey security association is the result of a successful STAKey announcement handshake.

### BKSA

BKSA is bidirectional. After a successful WAI certificate authentication process or PSK configuration, BKSAs are established in ASUE and AE and cached for up to their lifetime. BKSA is used to create the USKSA. BKSA consists of the following elements:

— BKID: the identifier of BKSA
—AE MAC address
—ASUE MAC address
— BK
— Lifetime
— AKMP
— Other security parameters

### USKSA

USKSA, which is the result of a successful unicast key negotiation process, is also bidirectional. Negotiated based on BK, USKSA is independent of BKSA and cached for up to its lifetime. For each pair of ASUE and AE, there are two USKSAs at most. Generally, only one USKSA is active in BSS. However, both of them may be active during rekeying process. After a unicast data MPDU encrypted with a new USKSA is correctly received, the old key is set to invalid. In IBSS, if $MAC_{AE}$ > MACASUE, the negotiated USKSA is used on unicast data MPDU and the key updating state is the same as in BSS. If $MAC_{AE}$ < $MAC_{ASUE}$, the negotiated USKSA can only be used in the multicast key announcement. When a new USKSA is active, the old one is set to invalid at once. The USKSA consists of the following elements:

— USKID
— USK
— The selected unicast cipher suite
— Lifetime
—AE MAC address
—ASUE MAC address
— Other security parameters, such as replay counter used in pre-authentication and STAKey request.

### MSKSA

MSKSA is the result of a successful multicast key announcement. In BSS, only one MSKSA is active. It is used to encrypt broadcast/multicast MPDUs by AP or decrypt received broadcast/multicast MPDUs by STA. In IBSS, every STA has more than one active MSKSAs, one for sending encrypted broadcast/multicast MPDUs and



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

others for decrypting broadcast/multicast MPDUs sent by corresponding peer STAs. During MSKSA refreshing, for every AP or peer STA, STA has two active MSKSAs used to decrypt broadcast/multicast MPDUs. Only after a broadcast/multicast MPDU is correctly decrypted with the new MSKSA, the old ones are set to invalid. However, for the MSKSA used to encrypt broadcast/multicast MPDUs, only one is active. The MSKSA consists of the following elements:

— Direction vector(whether the MSK is used for transmit or receive)
— MSKID
— The selected Multicast cipher suite
— Lifetime
— MSK
— AE MAC address
— Other security parameters

**STAKeySA**

STAKeySA is the result of a successful STAKey negotiation. It is a unidirectional security association from the initiator STA to the peer STA. The MSKSA consists of the following elements:

— STAKeyID
— STAKey
—Unicast cipher suite (adopts multicast cipher suite announced by AP)
— Initiator STA MAC address
— Peer STA MAC address

#### 8.1.2.1.1 Security association in BSS

When the WAPI security mechanism is adopted, STA/AP is identified by the WAPI parameter set which is contained in the Beacon frame or the Probe Response frame. If there is no such information contained in such frame, the WAPI security mechanism is not adopted in that STA/AP.

There are three ways to establish BKSA for a roaming STA in ESS:

— After association, execute the certificate authentication process or directly from PSK to establish BKSA.
— After a certificate authentication process with an AP in the ESS, STA can store that BKSA. When the STA moves to another AP, STA may place BKIDs corresponding to one or more BKSAs in the WAPI security parameter set of the (re) association request packet. If there is a valid corresponding BKSA in the AP, then the unicast key negotiation process can be directly executed without certificate authentication process. If there is no valid corresponding BKSA in the AP or there is no BKID in the WAPI security parameter set of the (re) association request packet of the STA, BKSA must be established from the PSK or a certificate authentication process between the STA and the AP.
— If a STA has completed the establishment of BKSA and USKSA with an AP in a ESS. It may establish a BKSA with other APs in the ESS through a pre-authentication process before association. The STA and the AP store the BKSA after a successful pre-authentication.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Based on BKSA, USKSA can be established with the unicast key negotiation process and MSKSA with the multicast key announcement process.

#### 8.1.2.1.2 Security association in IBSS

If WAPI security mechanism is adopted in IBSS, a security association should be established between every two STAs. A pair of STAs may negotiate a unicast cipher suite supported by both of them. Every STA shall include its list of the unicast cipher suite and the multicast cipher suite in its Beacon frame or Probe Response frame. The two STAs must support a common subset of unicast cipher suites and each STA shall support the multicast cipher suite broadcasted by the peer STA, otherwise BKSA between them can not be established.

In IBSS, if a STA requires establishing a security association with another STA whose security policy is unknown to it. It must send a Probe Request frame to obtain the security policy of the peer STA. The two STAs negotiate unicast cipher algorithms through the unicast key negotiation process. Selected unicast cipher suite is contained in the WAPI parameter set of the unicast key negotiation response. However, the WAPI parameter set contained in the unicast key negotiation acknowledgement is the same as in the Beacon frame or the Probe Response frame. Two STAs will use the unicast cipher algorithm through the unicast key negotiation initiated by the STA (as AE) with the larger MAC address.

If the certificate authentication method is adopted between two STAs, each of them initiates its own certificate authentication process, unicast key negotiation process and multicast key announcement process. Two sets of BKSAs, USKSAs and MSKSAs are established. If PSK method is adopted, BK is derived from PSK, a BKSA is established. Based on the BKSA, each of them initiates their own unicast key negotiation process and multicast key announcement process to establish two USKSAs and two MSKSAs.

#### 8.1.2.1.3 Deletion of WAPI security association

When STA/AP receives the primitive of Association, (Re)Association, Disassociation, Linkverification, De Linkverification, or it believes that it has drifted out of radio range of a STA/AP, some security associations will be deleted.

In BSS, a non-AP STA will delete USKSA and MSKSA, and AP will delete USKSA.

In IBSS, STA will delete the USKSA and the MSKSA used to receive data.

If the lifetime of a security association is run out or the state of a security association is set to invalid, the security association will be deleted. Based on its own management policy, a STA may set the state of some security associations to invalid.

### 8.1.2.2 Selection of WAPI security policy

The selection of the WAPI mechanism is decided by the process of association. STA executes WAPI policy selection by setting WAPI parameter set in the (re-)association frame.

STA/AP identifies its supporting WAPI security policy by the WAPI parameter set contained in the Beacon frame or the Probe Response frame. From the supporting WAPI security policy, the other STAs that require to associate



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

with it should choose the proper WAPI security policy. The association STA identifies the chosen WAPI security policy with the WAPI parameter set field contained in the association request frame. Thus, the selection of security policy is decided. If the association STA does not support the notified policy, the association request is denied.

STA and AP establish secure network by the steps below:

a) STA chooses AP by SSID
b) STA and AP execute linkverification
c) STA and AP execute association process to decide the selection of WAPI security mechanism. Based on the selection, STA and AP execute the WAI protocol described in 8.1.4.

In IBSS, STA chooses the WAPI security policy from the WAPI parameter set contained in the Beacon frame or the Probe Response frame sent by the peer STA. The policy includes the way of authentication, key management and multicast cipher suite. The unicast cipher suite which is negotiated in the process of the unicast key negotiation is based on the authentication and key management method in the chosen WAPI security policy.

### 8.1.3 Certificate

ASU (Authentication Service Unit) is an important component of WAI authentication infrastructure based on public-key cryptography technology. Its fundamental function is to verify the certificate of the user.

User certificate, which is a public-key certificate, is an important part of the construction of WAI system. The identity of the user can be uniquely identified by the certificate and the corresponding secret key. Public-key certificate is the digital identification of user.

Two kinds of public-key certificate format, X.509 v3 and GBW, are supported in this specification. User certificate and ASU certificate are used for signature verification.

#### 8.1.3.1 X.509 v3 Certificate

The format and management of X.509 v3 certificate is defined in ITU-T Recommendation X.509，Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks

The signature algorithm in certificate is 192-bit ECDSA. Hash algorithm is SHA256. The ECDSA algorithm is defined in ANSI X9.62-Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm(ECDSA). The SHA256 algorithm is defined in FIPS 180-2, Secure Hash Standard (SHS), August 2002.
The parameter for the ECDSA is denoted by OID.

Note: One set of appropriate parameters in ECDSA shall be adopted here, and the concrete values of parameters to be adopted are optional.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

### 8.1.3.2 GBW Certificate

#### 8.1.3.2.1 Certificate description

| |
|---|
| Version |
| Serial number |
| Issuer name |
| Validity period |
| Subject name |
| Subject public key information |
| Extension |
| Signature algorithm<br>Signature of the certificate by the issuer |

**Figure 43e—Format of certificate**

**Version**

This field determines the format of the certificate, which makes corresponding protocol be able to extract useful data items from the certificate.

**Serial number**

Each certificate issued by ASU is assigned a unique serial number. Along with the issuer, it can uniquely determine the certificate subject who possesses the certificate.

**Issuer name**

This field designates the identity of the issuer.

**Validity period**

This field prescribes the time interval during which the certificate is valid. It adopts UTC time format, which denotes the number of seconds from 0 o'clock on 1970.1.1 to current time.

**Subject name**

This field specifies the identity of who possesses the certificate.

**Subject public key information**

This field is the public-key information of the subject.

**Extension**

This field is kept for extension applications in the future.

**Signature algorithm**

This field specifies the signature algorithm used by the issuer on Signature of the certificate. Along with the



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

public key contained in the certificate, it can be used to verify a signature of the subject. It should include a hash algorithm a digital signature algorithm and corresponding parameters.

**Signature of the certificate by the issuer**

This field is the certificate issuer (ASU)'s signature of all the fields excluding itself.

### 8.1.3.2.2 Certificate format

The format of GBW certificate is as below:

| | Version | Serial number | Issuer | Validity period | Subject | Subject public key | Extension | Extension Attributes | Signature algorithm | Signature on certificate |
|---|---|---|---|---|---|---|---|---|---|---|
| **Octets:** | 2 | 4 | 6-256 | 8 | 6-256 | variable | 1 | variable | 2 | variable |

**Figure 43f—Definition of certificate content**

— Version field has the length of 2 octets, and the current version is 1

— Serial number field has the length of 4 octets

— The issuer field contains a length field and a content field. The length field denotes the length of the issuer field in octet and the length of itself is 1 octet.

— Validity period field has the length of 8 octets, of which 4 is for beginning time and 4 for ending time. Both the beginning time and the ending time are in UTC format.

— The subject field contains a length field and a subfield. The length field denotes the length of the subject field in octet and the length of itself is 1 octet.

— The subject public-key field is composed of two fields that are length and content. The length field denotes the length of subject public-key field, with the length of 2 octets. The content field contains three subfields that are public-key algorithm identifier, parameter and public key.

a) The Public-key Algorithm Identifier subfield is 1 octet in length, where value 1 stands for ECDSA-192 and other values are reserved.

b) Parameter subfield is composed of Parameter Identifier, Length and Content fields. The Parameter Identifier field is 1 octet in length. The Length field is 2 octets in length and denotes the number of octets in the Content field.

The value of Parameter Identifier field is defined as follow when the value of the Public-key Algorithm Identifier field is 1:

1 The Content field is OID which describes the parameter set. The code rule of OID is based on DER.

2 The Content field is the detailed parameters value. It defines the detailed parameters $q$, $a$, $b$, $h$, $n$, and $G$. A set of EC domain parameters specifies a field $GF(q)$, where $q$ is a positive odd prime integer $p$, $2^m$ for some positive integer $m$, or $p^m$ for an odd prime $p$ and some integer $m \gtrsim 2$; two elliptic curve coefficients $a$ and $b$, elements of $GF(q)$, that define an elliptic curve $E$; a positive prime integer $n$ dividing the number of points on $E$; and a curve point $G$ of order $n$ ($G$ is called the *generator* of a subgroup of order n). The cofactor $h = \#E / r$ (where $\#E$ is the number of points on $E$). The order of the detailed parameters in the



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

field is *q*, *a*, *b*, *h*, *n*, and *G*. Each parameter is described by length and value. The Length subfield is 1 octet in length and gives the number of octets in value subfield.

Other Parameter Identifiers are reserved.

c) Public key Algorithm Identifier subfield denotes the public key value. It has the length of 48 octets when the public key identifier is 1.

— Extension field has the length of 1 octet. It gives the number of the Extension Attributes.

— Extension Attributes field contains the certificate extensions. Every extension is defined as TLV format.

— Signature algorithm field shows which algorithm should be used on signature of the certificate by the issuer. It contains two subfields that are Length and Content. The Length subfield denotes the length of the Content field in octet with the Length of 2 octets. The Content subfield is composed of a one-octet hash algorithm identifier, a one-octet signature algorithm identifier and parameter.

a) Hash algorithm identifier is defined as below:

1 denotes SHA-256.

Other values are reserved.

b) Signature algorithm identifier

1 denotes ECDSA-192.

Other values are reserved.

c) Parameter denotes the parameter information of signature algorithm. It is composed of Parameter Identifier, Length and Content fields. The Parameter Identifier field is 1 octet in length. The Length field is 2 octets in length and denotes the octet number of Content field.

The value of parameter Identifier field is defined as follow when the value of Signature algorithm identifier is 1:

1 The Content field is OID which describes the parameter set. The code rule of OID is based on DER.

2 The Content field is the detailed parameters value. It defines the detailed parameters *q*, *a*, *b*, *h*, *n*, and *G*. A set of EC domain parameters specifies a field $GF(q)$, where $q$ is a positive odd prime integer $p$, $2^m$ for some positive integer $m$, or $p^m$ for an odd prime $p$ and some integer $m \geq 2$; two elliptic curve coefficients $a$ and $b$, elements of $GF(q)$, that define an elliptic curve $E$; a positive prime integer $n$ dividing the number of points on $E$; and a curve point $G$ of order $n$ ($G$ is called the *generator* of a subgroup of order n). The cofactor $h = \#E / r$ (where $\#E$ is the number of points on $E$). The order of the detailed parameters in the field is *q*, *a*, *b*, *h*, *n*, and *G*. Each parameter is described by length and value. The Length subfield is 1 octet in length and gives the number of octets in value subfield.

Other Parameter Identifiers are reserved.

— Signature on certificate field contains Length and Value fields. The Length field denotes the length of the Value field, with the length of 2 octets. Value field is the signature value of all the fields described above (except Signature on certificate field) by the issuer.

The management method of the certificate issuing and revoking, as well as the communication between two ASUs, is beyond this specification.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The Extension Attribute adopts the Type-Length-Value (TLV) format as below:

| | Type | Length | Value |
|---|---|---|---|
| **Octets:** | 1 | 2 | variable |

**Figure 43g—Extension Attribute format**

The 1-octet Type field denotes the type of this attribute.

1 Authority Key Identifier

Other values are reserved

The 2-octet Length field gives the number of octets in the Value field.

The Value field contains the content of this attribute data.

Authority Key Identifier attribute provides a method of identifying the public key corresponding to the private key used to sign a certificate. The Value field of Authority Key Identifier attribute contains an Issuer Name subfield and a Serial Number subfield of issuer certificate. Here, the Issuer Name field consists of a Length subfield and an Issuer Name Content field and the 2-octet Length field gives the number of octets in Issuer Name Content field; the Serial Number field consists of a Length field and a Serial Number Content field and the 2-octet length field gives the number of octets in Serial Number Content field.

#### 8.1.3.2.3 Issue format of the certificate

The default suffix of the issue certificate file is .wcr. It is stored on big endian mode which is network octet order.

The format of the issue certificate is as below:

| | Certificate file identification | Version | Certificate file length | Digest | Attribute1 | Attribute2 | Attribute3 |
|---|---|---|---|---|---|---|---|
| Octets: | 16 | 2 | 2 | 4-65539 | 4-65539 | 4-65539 | 4-65539 |

**Figure 43h—Issue format of the certificate**

The certificate file identification field shows the storage identification of the GBW certificate file. It is used to check whether the file is an issued certificate file of GBW type. If so, the value of this field should be “WAI15629.11-2003” with the form of ASCII code character string (denoted in hex as 0x57 0x41 0x49 0x31 0x35 0x36 0x32 0x39 0x2E 0x31 0x31 0x2D 0x32 0x30 0x30 0x33, with the length of 16 octets.)

The version field is the version of the storage format of the certificate file, and its current value is 1, with the length of 2 octets.

The certificate file length field indicates the length of the digest and all attribute fields in octet, with the length of



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2 octets.

The digest field is the digest of all the fields behind the digest field by certain digest algorithm. The format of the digest field is as below:

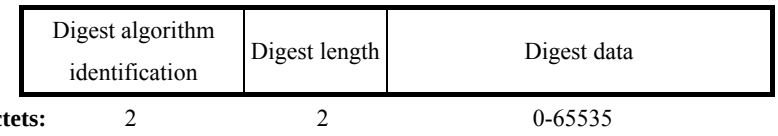



**Figure 43i—Digest field**

Digest algorithm identification field identifies the digest algorithm, with the length of 2 octets. Digest algorithm identification is defined as below:

— 1 denotes SHA-256 hash algorithm. If the value is 1, the digest length value is 36 and digest data is 32-octet digest content.

— Other values are reserved.

Digest length field denotes the length of the digest data field in octet, with the length of 2 octets.

Digest data field is the content of the digest.

There may be several attribute fields in the certificate file. The format of the attribute field is as below:

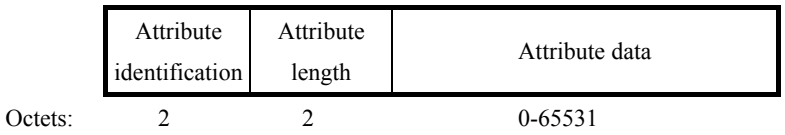



**Figure 43j—Attribute field**

Attribute identification field denotes the type of attribute, with the length of 2 octets. It is defined as below:

— 1 the attribute data is the certificate of the issuer
— 2 the attribute data is the certificate of the user
— 3 the attribute data is the secret key corresponding to user certificate.
— Other values are reserved

The attribute length field shows the length of the attribute data in octet, with the length of 2 octets.

Attribute data field is the content of the attribute.

If a private key attribute according to the user certificate is contained in the certificate file, the last attribute field before this attribute field should be the certificate according to this private key.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

### 8.1.4 WAI protocol

#### 8.1.4.1 Format of WAI protocol packet

The octets in the fields longer than a single octet are sent by the order from the octet containing the highest numbered bits to the octet containing the lowest numbered bits (Network Byte Order).

The format of the WAI protocol packet is defined as below:

| | Version | Type | Sub type | Reserved | Length | Packet sequence number | Fragment sequence number | Flag | Data |
|---|---|---|---|---|---|---|---|---|---|
| **Octets:** | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | variable |

**Figure 43k—Format of WAPI protocol packet in WAI authentication system**

— The Version field denotes the version of authentication infrastructure, with the length of 2 octets. Current version value is 1.

— The Type field denotes the protocol type, with the length of 1 octet. It is defined as below:

1 denotes WAI protocol package.

Other values are reserved.

— The Subtype field is 1 octet in length. If the value of the Type field is not 1, the Subtype field is reserved. Otherwise, the Subtype field is defined as below:

1 pre-authentication start
2 STAKey request
3 authentication activation
4 access authentication request
5 access authentication response
6 certificate authentication request
7 certificate authentication response
8 unicast key negotiation request
9 unicast key negotiation response
10 unicast key negotiation confirmation
11 multicast key/STAKey announcement
12 multicast key/STAKey announcement response

Other values are reserved.

— The Reserved field is 2 octets in length, with the default value 0.

— The Length field denotes the length of all the fields of the WAPI protocol packet in octet, with the length of 2 octets.

— The Packet Sequence Number field denotes the sequence number of the packet, with the length of 2 octets.

— The Fragment Sequence Number denotes the sequence number of the fragment, with the length of 1 octet. For every packet, the sequence number of the first fragment is 0, and that of the following is



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

increased by degree of 1.

— The Flag field is 1 octet in length in which bit 0 is More Frag field, bits 1-7 are reserved. The More Frag field is set to 1 in packet that have another fragment of the current WAI packet to follow. It is set to 0 in all other frames.

The Packet Sequence Number, Fragment Sequence Number and More Frag field is valid only in those packets between ASUE and AE.

— The content of the Data field is determined by the values of the Type field and the Subtype field. Besides the fixed content, some optional attributes are also included in this field.

The maximal length of the WAI protocol packet is 65535 octets.

#### 8.1.4.1.1 Fixed content in the data field of the WAI protocol packet

a) FLAG

This field is 1 octet in length, with the format below:

| B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 |
|---|---|---|---|---|---|---|---|
| BK Rekeying | Pre-Authentication | Certificate Authentication Request | Optional Field | USK Rekeying | STAKey Negotiation | STAKey Revoking | Reserved |

**Figure 43l—FLAG**

— Bit 0: BK Rekeying: A STA sets BK Rekeying subfield to 1 to signal the packet is a BK rekeying packet and set this subfield to 0 when the packet is not a BK rekeying packet.

— Bit 1: Pre-Authentication: A STA sets Pre-Authentication subfield to 1 to signal the packet is a pre-authentication packet and set this subfield to 0 when the packet is not a pre-authentication packet.

— Bit 2: Certificate Authentication Request: A STA sets this subfield to 1 if it requires to verify the peer's certificate, otherwise sets the subfield to 0.

— Bit 3: Optional Field : If there are some optional fields in the packet, this subfield is set to 1. Otherwise this subfield is set to 0.

— Bit 4: USK Rekeying: A STA sets USK Rekeying subfield to 1 to signal the packet is a USK rekeying packet and set this subfield to 0 when the packet is not a USK rekeying packet.

— Bit 5: STAKey Negotiation: A STA sets STAKey Negotiation subfield to 1 to signal the packet is a STAKey negotiation packet and set this subfield to 0 when the packet is not a STAKey negotiation packet.

— Bit 6: STAKey Revoking: This subfield is set to 1 when a STA delete a STAKey and set to 0 otherwise.

— Bit 7: Reserved.

b) BKID

BK identifier, 16 octets in length, is computed as BKID= KD_HMAC_SHA256(BK, $MAC_{AE}$ ||$MAC_{ASUE}$) .

c) USKID

USKID is 1 octet in length, in which bit 0 is significant.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

d) MSKID/STAKeyID

MSKID / STAKeyID is 1 octet in length, in which bit 0 is significant.

e) Result

This field is 1 octet in length. The value may be 0 (success) or other values (reason code).

f) Nonce

This field is 32 octets in length.

g) Key Data

This field is composed of two subfields which are the length subfield and the content subfield. The length subfield is 1 octet in length and denotes the number of octets in the content field.

h) Key Announcement Identifier

This field is 16 octets in length.

i) Data Sequence Number

This field is 16 octets in length.

j) Certificate




| Octets: | 2 | 2 | 0-65535 |
|---|---|---|---|

**Figure 43m—Certificate**

— The Certificate Identifier subfield denotes the type of the certificate, with the length of 2 octets. It is defined as below:

1 denotes X.509 v3 certificate in the Certificate Data subfield.

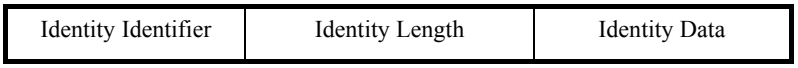
2 denotes GBW certificate in the Certificate Data subfield

Other values are reserved.

— The Certificate Length subfield gives the number of octets in the Certificate Data subfield, with the length of 2 octets.

— The Certificate Data subfield contains the certificate content.

k) Identity




| Octets: | 2 | 2 | 0-65535 |
|---|---|---|---|

**Figure 43n—Identity**

— The Identity Identifier subfield denotes the type of Identity, with the length of 2 octets. It is defined as below:

1 The Identity Data subfield contains three subfields: the Subject subfield, the Issuer subfield and the Serial Number subfield.

Other values are reserved.

— The Identity Length subfield gives the number of octets in the Identity Data subfield, with the length of 2 octets.

— The Identity Data subfield contains the identity content. When the Identity Identifier subfield is set to 1,



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

the Subject subfield contains a length field and a subject content field. The 2-octet length field gives the number of octets in subject content field; the Issuer subfield contains a length field and an issuer content field. The 2-octet length field gives the number of octets in issuer content field; the Serial Number subfield contains a length field and a serial number content field. The 2-octet length field gives the number of octets in serial number content field.

l) ADDID

This field is 12 octets in length.

| | MAC address 1 | MAC address 2 |
|---|---|---|
| **Octets:** | **6** | **6** |

**Figure 43o—ADDID**

### 8.1.4.1.2 Attribute content in the data field of the WAI protocol packet

The attribute content adopts the Type-Length-Value (TLV) format as below:

| | Type | Length | Value |
|---|---|---|---|
| **Octets:** | 1 | 2 | variable |

**Figure 43p—Attribute format**

The 1-octet Type field denotes the type of this attribute.

The 2-octet Length field gives the number of octets in the Value field.

The Value field contains the content of this attribute.

a) Signature attribute

| | Type=1 | Length | Identity | Signature Algorithm | Signature Value |
|---|---|---|---|---|---|
| **Octets:** | 1 | 2 | variable | variable | variable |

**Figure 43q—Signature attribute**

The Identity field has the same definition as 8.1.4.1.1 (k).

The Signature Algorithm field contains two subfields that are the length subfield and the content subfield. The length subfield is 2 octets in length and gives the length of the content subfield in octet. The content subfield contains three subfields: a one-octet hash algorithm identifier, a one-octet signature algorithm identifier and parameter.

1) Hash algorithm identifier is defined as below:

1 SHA-256.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Other values are reserved.

2) Signature algorithm identifier

1 ECDSA-192.

Other values are reserved.

3) Parameter contains the parameter information of the signature algorithm. It is composed of Parameter Identifier, Length and Content fields. The Parameter Identifier field is 1 octet in length. The length field is 2 octets in length and gives the number of octet in Content field.

The value of Parameter Identifier is defined as follow when the value of Signature Algorithm Identifier is 1:

1 The Content field is OID which describes the parameter set. The code rule of OID is based on DER.

2 The Content field is the detailed parameters value. It defines the detailed parameters *q*, *a*, *b*, *h*, *n*, and *G*. A set of EC domain parameters specifies a field $GF(q)$, where $q$ is a positive odd prime integer $p$, $2^m$ for some positive integer $m$, or $p^m$ for an odd prime $p$ and some integer $m \geq 2$; two elliptic curve coefficients $a$ and $b$, elements of $GF(q)$, that define an elliptic curve $E$; a positive prime integer $n$ dividing the number of points on $E$; and a curve point $G$ of order $n$ ($G$ is called the *generator* of a subgroup of order n). The cofactor $h = \#E / r$ (where $\#E$ is the number of points on $E$). The order of the detailed parameters in the field is *q*, *a*, *b*, *h*, *n*, and *G*. Each parameter is described by length and value. The Length subfield is 1 octet in length and gives the number of octets in value subfield.

Other Parameter Identifiers are reserved.

The Signature Value field contains a Length subfield and a Content subfield. The Length field is 2 octets in length and gives the number of octets in the Content field. The Content subfield contains the signature value.

b) Certificate Verification Result

| type=2 (1 octet) | length (2 octets) |
|---|---|
| Nonce 1 (32 octets) | |
| Nonce 2 (32 octets) | |
| verification result (1 octet) | Certificate1 (variable) |
| verification result (1 octet) | Certificate2 (variable) |

**Figure 43r—Certificate Verification Result**

c) Identity List



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

| type=3 (1 octet) | length (2 octets) |
|---|---|
| reserved (1 octet) | number of Identities (2 octets) |
| Identity1 (variable) | |
| Identity2 (variable) | |
| … | |

**Figure 43s—Identity List**

The definition of Identity 1, Identity 2, Identity3, …, should be the same definition as 8.1.4.1.1(k).

### 8.1.4.2 Certificate Authentication procedure

The Certificate Authentication procedure executes the authentication and key negotiation to establish BKSA based on the certificate of the STA/AP. It is depicted by Figure 43t.

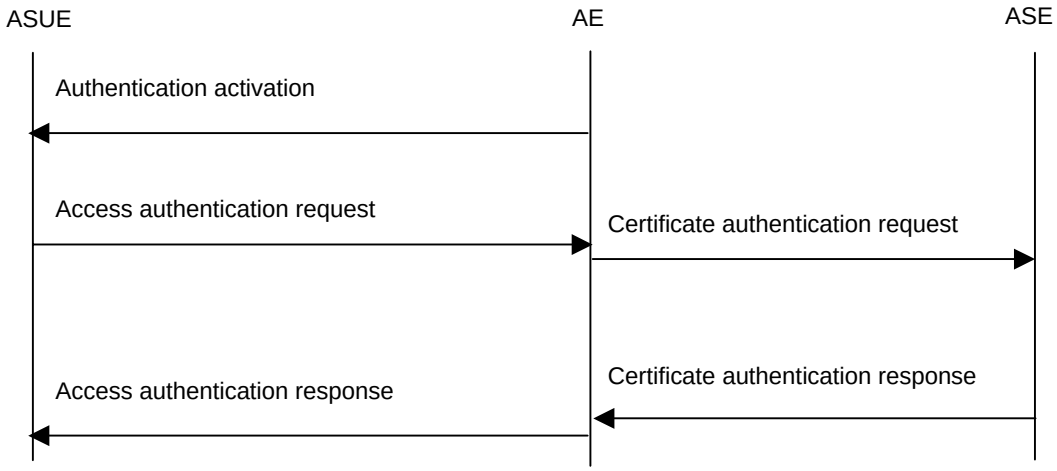



**Figure 43t—Certificate Authentication procedure**

#### 8.1.4.2.1 Authentication Activation

The authentication activation packet is illustrated in Figure 43u.

| | FLAG | Authentication Identifier | Local ASU Identity | $STA_{AE}$ Certificate | ECDH Parameter |
|---|---|---|---|---|---|
| **Octets:** | 1 | 32 | variable | variable | variable |

**Figure 43u—The format of the Data field of Authentication Activation packet**

— The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Here only bit 0 and bit 1 are significant. When the certificate authentication procedure is for the initial BKSA establishment, bit 0 (BK Rekeying) is set to 0. When the certificate authentication procedure is for BK rekeying, bit 0(BK Rekeying) is set to 1. If the certificate authentication procedure is for Pre-Authentication, bit 1 (Pre-Authentication) is set to 1; otherwise, bit 1 is set to 0.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

—Authentication Identifier field is 32 octets in length. If the BK Rekeying is 0, the Authentication Identifier field is a random number generated by AE. If the BK Rekeying is 1, the Authentication Identifier field is equal to the value negotiated in the last certificate authentication procedure.

— The Local ASU Identity field uses the same definition as that in 8.1.4.1.1. It describes the ASU that the AE trusts.

— The $STA_{AE}$ Certificate field contains the certificate of $STA_{AE}$, whose format is the same definition as that in 8.1.4.1.1.

— The ECDH Parameter field is 1 octet in length and indicates the parameter used by ECDH. It is composed of Parameter Identifier, Length and Content fields. The Parameter Identifier field is 1 octet in length. The Length field is 2 octets in length and gives the number of octets in the Content field.

The value of Parameter Identifier field is as follow:

1 The Content field is OID which describes the parameter set. The code rule of OID is based on DER.

2 The Content field is the detailed parameters value. It defines the detailed parameters *q*, *a*, *b*, *h*, *n*, and *G*. A set of EC domain parameters specifies a field $GF(q)$, where $q$ is a positive odd prime integer $p$, $2^m$ for some positive integer $m$, or $p^m$ for an odd prime $p$ and some integer $m \geq 2$; two elliptic curve coefficients $a$ and $b$, elements of $GF(q)$, that define an elliptic curve $E$; a positive prime integer $n$ dividing the number of points on $E$; and a curve point $G$ of order $n$ ($G$ is called the *generator* of a subgroup of order n). The cofactor $h = \#E / r$ (where $\#E$ is the number of points on $E$). The order of the detailed parameters in the field is *q*, *a*, *b*, *h*, *n*, and *G*. Each parameter is described by length and value. The Length subfield is 1 octet in length and gives the number of octets in value subfield.

Other Parameter Identifiers are reserved.

The ECDH algorithm is defined in ISO/IEC 15946 Information technology - Security techniques - Cryptographic techniques based on elliptic curves.

The AE sends an authentication activation packet to ASUE when:

a) certificate authentication and key management method is chosen in (re)association
b) re-authentication and rekeying is required according to the local policy of the AE
c) the AE receives a Pre-authentication Start packet from the ASUE.

After the ASUE receives the Authentication Activation packet from the AE,

a) If the value of bit 0 (BK Rekeying) of the FLAG field in the packet is 1, the ASUE proceeds to Step b).Otherwise, goes to Step c).
b) If the Authentication Identifier of the packet dose not match the one kept in the ASUE which is negotiated in the last Certificate Authentication procedure, the ASUE silently discards the packet. Otherwise the ASUE proceeds to Step c).
c) The ASUE chooses the corresponding $STA_{ASUE}$ certificate according to the ASU Identity in the Authentication Activation packet or local strategies. And it generates temporary secret key x, temporary public key x·P for ECDH exchange and an ASUE challenge. The ASUE constructs the Access Authentication Request packet and sends it to the AE.



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

### 8.1.4.2.2 Access Authentication Request

The Access Authentication Request packet is illustrated in Figure 43v.

| | FLAG | Authentication Identifier | ASUE Challenge | ASUE Key Data | $STA_{AE}$ Identity | $STA_{ASUE}$ Certificate | ECDH Parameter | ASU List trusted by ASUE | ASUE Signature |
|---|---|---|---|---|---|---|---|---|---|
| **Octets:** | 1 | 32 | 32 | variable | variable | variable | variable | variable | variable |

**Figure 43v—The fomat of the Data field of access authentication request packet**

— The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Here only bits 0,1,2 and 3 are significant. Bit 0 and Bit 1 should have the same value as the FLAG filed in the Authentication Activation. The ASUE sets Bit 2 (Certificate Verification Request Identifier) to 1 if it requires to verify the AE certificate; otherwise set it to 0. Bit 2 must be set to 1 if bit 0 (BK Rekeying) is set to 0, i.e., if the packet is not for BK Rekying, the AE's certificate must be verified. If there are some optional fields in the packet, bit 3 (Optional Field) is set to 1. Otherwise set to 0.

— The Authentication Identifier field is 32 octets in length. If the BK Rekeying is 0, it should be equal to the corresponding field in the Authentication Activation packet sent by AE. If the BK Rekeying is 1, it should be equal to the value negotiated in the last certificate authentication procedure.

— The ASUE Challenge field is generated randomly by ASUE, with the length of 32 octets.

— The ECDH Parameter field should be the same as the one in the Authentication Activation packet.

— The ASUE Key Data field adopts the format defined in 8.1.4.1.1. The content of the field is the temporary public key of the ASUE used in ECDH exchange.

— The $STA_{AE}$ Identity field adopts the format defined in 8.1.4.1.1. The field contains the ID of $STA_{AE}$.

— The $STA_{ASUE}$ Certificate field adopts the format defined in 8.1.4.1.1. The field contains the certificate of $STA_{ASUE}$.

— The ASU List trusted by ASUE field is optional and is encapsulated in the Identity attribute defined in 8.1.4.1.2 (c). The content of the field contains the ASUs trusted by $STA_{ASUE}$ except the issuer of the $STA_{ASUE}$ certificate. If the ASUE trusts other entities besides its certificate issuer, this field can be used to inform the ASU about the case.

— The ASUE Signature field is encapsulated in the Signature attribute defined in 8.1.4.1.2 (a). It is a signature on all the fields excluding the packet header. In computing a signature, the value of the Content subfield in the signature attribute is masked with 0.

After receiving an Authentication Activation from the AE or when a BK rekeying is needed, the ASUE sends an Access Authentication Request packet to the AE.

After the AE receives the Access Authentication Request packet from the ASUE,

a) If the AE has not sent an Authentication Activation packet, then it checks whether the Authentication Identification field matches the one negotiated in the last Certificate Authentication procedure. If then, the AE proceeds to Step b). Otherwise the AE silently discards the packet. If the AE has sent an



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Authentication Activation packet and the Authentication Identifier field as well as bit 0 and bit 1 of the FLAG field match those in the Authentication Activation packet sent by the AE, respectively. The AE proceeds to Step b). Otherwise the AE silently discards the packet.

b) In the following cases , the AE silently discards the packet.

— the Identification field of $STA_{AE}$ does not match its own Identity,

— the ECDH Parameter field dose not match the one in the Authentication Activation packet,

— the signature of the ASUE is not valid

In the following cases, AE constructs the Certificate Authentication Request packet and sent it to the ASU.

— bit 2 in the FLAG field is 1

— AE requires to verify the $STA_{ASUE}$ certificate according to its local policy.

In any other case, the AE proceeds to Step c).

c) If the $STA_{ASUE}$ certificate is valid, the AE generates a 32-octet random number $N_{AE}$ as a challenge of AE, as well as its own temporary private key y and temporary public key y·P. Then AE obtains the master key seed (x·y·P) abscissa through ECDH computation. By the formula KD_HMAC_SHA256 ((x·y·P) abscissa, NAE||NASUE|| ”base key expansion for key and additional nonce”), the AE computes a 16-octet base key BK and a 32-octet seed. Then, by a SHA256 computation on the seed, a 32-octet Authentication Identifier used in the next Certificate Authentication procedure is obtained. Finally, the AE constructs the Access Authentication Response packet and sent to the ASUE, in which Access State is set to success and the bit 3 (Optional Field) of FLAG field is set to 0.

On the other hand, if the $STA_{ASUE}$ certificate is not valid, the AE constructs the Access Authentication Response packet and sent to the ASUE, in which Access State is set to failure, the bit 3 (Optional Field) of FLAG field is set to 0, the Challenge $N_{AE}$ field and the AE Key Data (AE temporary public key) field are set to an arbitrary value. Finally, the AE delinkverifies with $STA_{ASUE}$.

#### 8.1.4.2.3 Certificate Authentication Request

The Certificate Authentication Request packet is illustrated in Figure 43w and described after the figure.

| | ADDID | AE Challenge | ASUE Challenge | $STA_{ASUE}$ Certificate | $STA_{AE}$ Certificate | ASU List trusted by ASUE |
|---|---|---|---|---|---|---|
| **Octets:** | 12 | 32 | 32 | variable | variable | variable |

**Figure43 w—The format of the Data field of Certificate Authentication Request packet**

— The ADDID field is assigned by $MAC_{AE}||MAC_{ASUE}$ with the length of 12 octets.

— The AE Challenge field is a random number generated by AE. It is 32 octets in length.

— The ASUE Challenge field is 32 octets in length. It should be equal to the one in the Access authentication Request packet.

—The $STA_{ASUE}$ Certificate field should be equal to the one in the Access Authentication Request packet.

— The $STA_{AE}$ Certificate field adopts the format defined in 8.2.1.4.1.1. The field contains the certificate of $STA_{AE}$.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— The ASU List trusted by ASUE field is optional and should be equal to the one in the Access Authentication Request packet.

If the FLAG field in the Access Authentication Request packet indicates a certificate verification request or the AE itself needs a certificate verification, the AE sends a Certificate Authentication Request to the ASU.

After receiving an Access Authentication Request packet from ASUE and sending a Certificate Authentication Request packet to the ASU, the AE will not handle any Access Authentication Request packet sent by the ASUE until the Certificate Authentication Request packet is timeout.

After the ASU receives the Certificate Authentication Request packet,

a) Cooperating with other entities, the ASU verifies the $STA_{AE}$ certificate and the $STA_{ASUE}$ certificate. If any of the certificate can not be verified, ASU sets the corresponding verification result to 'unknown certificate issuer', and then proceeds to Step b). If both of them can be verified, ASU verifies the state of $STA_{AE}$ certificate and $STA_{ASUE}$ certificate, and then proceeds to Step b).
b) According to the result of the $STA_{AE}$ certificate and $STA_{ASUE}$ certificate verification, the ASU constructs a Certificate Authentication Response packet with the corresponding signature and sends it to the AE.

#### 8.1.4.2.4 Certificate Authentication Response

The Certificate Authentication Response packet is illustrated in Figure 43x.

| | ADDID | Authentication Result of the Certificate | Server Signature trusted by ASUE | Server Signature trusted by AE |
|---|---|---|---|---|
| **Octets:** | 12 | variable | variable | variable |

**Figure 43x—The format of the Data field of Certificate Authentication Response packet**

— The ADDID field is assigned by $MAC_{AE}||MAC_{ASUE}$ with the length of 12 octets. It should be equal to the one in the Certificate Authentication Request packet.

— The Authentication Result of the Certificate is encapsulated in the Certificate Authentication Result attribute defined in 8.2.1.4.1.2 (b). The first nonce in the field is equal to the AE Challenge value in the Certificate Authentication Request packet. The second one is equal to the ASUE challenge value. The first certificate and its result correspond to the $STA_{ASUE}$ certificate. The second certificate and its result correspond to the $STA_{AE}$ certificate. The definition of the certificate result is as follows:

0 the certificate is valid
1 the issuer is unknown
2 the certificate is based on an untrusted root
3 the certificate is not time valid
4 the certificate have not a valid signature
5 the certificate is revoked
6 the certificate is not valid for proposed usage
7 the revocation state of the certificate is unknown



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

8 the certificate has an unknown error

Other values are reserved.

— Server Signature trusted by ASUE field is encapsulated in signature attribute defined in 8.1.1.4.1.2 (a). It signs the Authentication Result of the Certificate field and itself. When the signature is being calculated, the value of the Content subfield in the signature attribute is masked as 0.

— Server Signature trusted by AE field is encapsulated in signature attribute defined in 8.1.1.4.1.2 (a). It signs all the fields except the ADDID field in this packet, excluding the packet header. When the signature is being calculated, the value of the Content subfield in the signature attribute is masked as 0.

NOTE: If the server trusted by ASUE is the same as trusted by AE, then the Certificate Authentication Response packet will not include Server Signature trusted by ASUE field.

When the ASU receives the Certificate Authentication Request packet, it will send the Certificate Authentication Response packet to the AE.

After the AE receives the Certificate Authentication Response packet,

a) According to the ADDID, the AE identifies the corresponding Certificate Authentication Request packet. If the first nonce in the Authentication Result field in the Certificate Authentication Response packet matches the AE Challenge value in the Certificate Authentication Request packet, proceeds to Step b); otherwise, the AE discards this packet.

b) The AE verifies the signature with the signature list of its trusting ASU. If the signature is valid, proceeds to Step c). Otherwise, the AE discards this packet.

c) If the $STA_{ASUE}$ certificate is valid, the AE generates a 32-octet random number $N_{AE}$ as a challenge of AE, as well as its own temporary private key y and temporary public key y·P. Then AE obtains the master key seed (x·y·P) abscissa through ECDH computation. By the formula KD_HMAC_SHA256 ((x·y·P) abscissa, NAE||NASUE|| ”base key expansion for key and additional nonce”), the AE computes out a 16-octet base key BK and a 32-octet seed. Then, by a SHA256 computation on the seed, a 32-octet Authentication Identifier used in the next Certificate Authentication procedure is obtained. Finally, the AE constructs the Access Authentication Response packet sent to the ASUE, in which Access State is set to success. If the ASUE requires to verify the STAAE certificate, the bit 3 (Optional Field) of FLAG field is set to 1 and the Access Authentication Response packet will include Multiple Certificate Verification Result field. Otherwise, the bit 3 (Optional Field) of FLAG field is set to 0 and the Access Authentication Response packet will not include Multiple Certificate Verification Result field.

On the other hand, if the $STA_{ASUE}$ certificate is not valid, the AE constructs the Access Authentication Response packet and sent to the ASUE, in which Access State is set to failure, and the Challenge $N_{AE}$ field and the AE Key Data (AE temporary public key) field are set to an arbitrary value. If the ASUE requires to verify the $STA_{AE}$ certificate, the bit 3 (Optional Field) of FLAG field is set to 1 and the Access Authentication Response packet will include Multiple Certificate Verification Result field. Otherwise, the bit 3 (Optional Field) of FLAG field is set to 0 and the Access Authentication Response packet will not include Multiple Certificate Verification Result field. Finally, the AE delinkverifies with $STA_{ASUE}$.



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

### 8.1.4.2.5 Access Authentication Response

The Access Authentication Response packet is illustrated in Figure 43y and described after the figure.

| | FLAG | ASUE Challenge | AE Challenge | Access Result | ASUE Key Data | AE Key Data | $STA_{AE}$ Identity | $STA_{ASUE}$ Identity | Multiple Certificate Verification Result | AE Signature |
|---|---|---|---|---|---|---|---|---|---|---|
| **Octets:** | **1** | **32** | **32** | **1** | **Variable** | **Variable** | **Variable** | **Variable** | **Variable** | **Variable** |

**Figure 43y—The format of the Data filed of Access Authentication Response packet**

—The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Here only bits 0,1 and 3 are significant. Bit 0 and bit 1 should have the same value as the FLAG filed in the Access Authentication Request sent by the ASUE. The AE sets bit 3 (Optional Field) to corresponding value according to the context.

— The ASUE Challenge field is 32 octets in length. It should be equal to the one in the Access Authentication Request packet.

— The AE Challenge field is 32 octets in length. It should be equal to the one in the Certificate Authentication Request packet.

—Access Result field is 1 octet in length. The definition is as follows:

0 success, corresponding value of certificate verification result is 0

1 cannot verify the certificate, and corresponding value of certificate verification result is 1

2 certificate error, and corresponding value of certificate verification result is other numerical value than 0 and 1

3 prohibition of the local strategy

Other values are reserved.

— The ASUE Key Data field adopts the format defined in 8.2.1.4.1.1. The content of the field is the temporary public key of the ASUE used in ECDH exchange. The field should be equal to the one in the Access Authentication Request packet.

— The AE Key Data field adopts the format defined in 8.2.1.4.1.1. The content of the field is the temporary public key of the AE used in ECDH exchange.

— The $STA_{AE}$ Identity field adopts the format defined in 8.2.1.4.1.1. The field contains the ID of $STA_{AE}$.

— The $STA_{ASUE}$ Identity field adopts the format defined in 8.2.1.4.1.1. The field contains the ID of $STA_{ASUE}$.

— The Multiple Certificate Verification Result field is optional. If exists, it contains all the fields except the ADDID with the same content in the Certificate Authentication Response packet.

— The AE Signature field is encapsulated in the Signature attribute defined in 8.2.1.4.1.2 (a). It is a signature on all the fields excluding the packet header. In computing a signature, the value of the Content subfield in the signature attribute is masked as 0.

When the AE receives the Certificate Authentication Responding packet or the Access Authentication request packet, it will send the Access Authentication Response packet to ASUE.

After the ASUE receives the Access Authentication Response packet,



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

a) If the ASUE conforms that the packet corresponds to the current Access Authentication Requesting packet according to $STA_{AE}$ Identity and $STA_{ASUE}$ Identity, the ASUE proceeds to Step b). Otherwise the ASUE discards the packet.

b) If the bit 0 and bit 1 in the FLAG field match the ones in the Access Authentication Request packet respectively, the ASUE proceeds to Step c). Otherwise the ASUE discards the packet

c) If the ASUE Challenge field and the ASUE Key Data match those in the Access Authentication Request packet respectively, the ASUE proceeds to Step d). Otherwise the ASUE discards the packet.

d) If the ASUE does not require the certificate verification in the Access Authentication Requesting packet, the ASUE proceeds to Step e). Otherwise, the ASUE will verify the ASU signature in the Multiple Certificate Authentication Result field. If the verification is not correct, the ASUE discards the packet. Otherwise, the ASUE will verify the AE certificate authentication result. If it is not valid, the ASUE terminates the linkverification with the $STA_{AE}$. Otherwise, the ASUE proceeds to Step e).

e) If the AE signature is not valid, the ASUE discards the packet. Otherwise, the ASUE will verify the Access Result. If it is not "successful," the ASUE terminates the linkverification with the $STA_{AE}$. Otherwise, the ASUE proceeds to Step f).

f) The ASUE obtains the master key seed (x·y·P) abscissa through ECDH computation. By the formula KD_HMAC_SHA256 ((x·y·P) abscissa, NAE||NASUE|| "base key expansion for key and additional nonce"), the ASUE computes out a 16-octet base key BK and a 32-octet seed. Then, by a SHA256 computation on the seed, a 32-octet Authentication Identifier used in the next Certificate Authentication procedure is obtained.

For the ECDH algorithm, there are some explanations as follows:

a) Temporary private keys x and y are the integers between 1 and n-1. n is the degree of the base point p in the elliptic curve domain parameters.

b) Temporary pubic keys x.p and y.p are the points in the elliptic curve defined in the elliptic curve domain parameters.

c) The key seed $(x \cdot y \cdot P)_{abscissa}$ negotiated by ECDH is the x-coordinate in the x·y·P. x·y·P can not be an infinite point.

#### 8.1.4.3 Unicast key negotiation procedure

Unicast key negotiation procedure uses BK to complete the negotiation of the unicast session key, and constructs the USKSA.



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

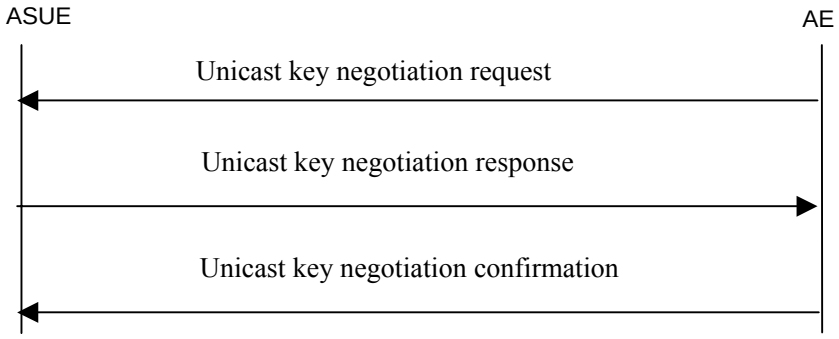



**Figure 43z—Unicast key negotiation procedure**

### 8.1.4.3.1 Unicast Key Negotiation Request

The Unicast Key Negotiation Request packet is illustrated in Figure 43aa.

| | FLAG | BKID | USKID | ADDID | AE Challenge |
|---|---|---|---|---|---|
| **Octets:** | 1 | 16 | 1 | 12 | 32 |

**Figure 43aa—The format of the Data field of Unicast Key Negotiation Request packet**

— The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Here, only bit 4 (USK Rekeying) is significant. When the unicast key negotiation procedure is for the initial USKSA establishment, bit 4 (USK Rekeying) is set to 0. When the unicast key negotiation procedure is for USK rekeying, bit 4 (USK Rekeying) is set to 1

— The BKID field is 16 octets in length, defined in 8.2.1.4.1.1. It indicates the current shared BK.

— The USKID field is 1 octet in length in which the bit 0 is the index of the unicast session key of the negotiation, and all the other bits are reserved. Bit 0 of this field is originally set to 0, and it switches between 0 and 1 after each unicast key negotiation.

— The ADDID field is 12 octets in length, defined in 8.2.1.4.1.1. MAC address 1 in the field is the AE MAC address, and MAC address 2 is the ASUE MAC address.

— The AE Challenge field is 32 octets in length. If the USK Rekeying is 0, it is a random number generated by AE. If the USK Rekeying is 1, it is equal to the value negotiated in the last unicast key negotiation procedure.

The AE will send a unicast key negotiation request packet to ASUE and start the unicast key negotiation with ASUE in the following cases:

— After the AE completes the certificate authentication procedure and constructs the valid BKSA,

— using preshared key authentication method,

— cached BKSA is used,

— rekeying unicast key.



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

After the ASUE receives the Unicast Key Negotiation Request packet,

a) If the BKSA referred by BKID is valid and bit 4 of the FLAG field (USK Rekeying) is 1, the ASUE goes to Step c). If BKSA referred by BKID is valid, bit 4 of the FLAG field (USK Rekeying) is 0 and USKSA referred by USKID is not valid, the ASUE proceeds to Step b). Otherwise the ASUE discards the packet.

b) If the AE Challenge is equal to the one negotiated in the last Unicast Key Negotiation procedure, the ASUE proceeds to Step c). Otherwise the ASUE discards the packet.

c) The ASUE generates a random number as the ASUE challenge. Then the ASUE generates 96 octets through the formula KD_HMAC_SHA256 (BK, ADDID|| N1||N2||"pairwise key expansion for unicast and additional keys and nonce") in which N1 is the AE Challenge and N2 is the ASUE Challenge. The first 64 octets are the unicast session key (the first 16 octets are the unicast encryption key, the second 16 octets are the unicast integrity check key, the third 16 octets are KCK, and the fourth 16 octets are KEK). The last 32 octets are the AE Challenge seed of the next unicast session key negotiation procedure. By a SHA256 computation on the seed, a 32-octet AE Challenge used in the next unicast session key negotiation procedure is obtained.

d) The ASUE calculates the message authentication code by the HMAC_SHA256 algorithm with KCK and constructs the Unicast Key Negotiation Response packet to send to the AE.

e) The ASUE will install the new unicast session key when it is in BSS. When the ASUE is in IBSS, it will install the new unicast session key only if its MAC address is less than the AE's MAC address. The ASUE should only decapsulate the received unicast data frame with the new unicast session key and must not encapsulate the sent unicast data frame with the new key.

#### 8.1.4.3.2 Unicast Key Negotiation Response

The Unicast Key Negotiation Response packet is illustrated in Figure 43ab.

| | FLAG | BKID | USKID | ADDID | ASUE Challenge | AE Challenge | $WIE_{ASUE}$ | Message Authentication Code |
|---|---|---|---|---|---|---|---|---|
| **Octets:** | 1 | 16 | 1 | 12 | 32 | 32 | variable | 20 |

**Figure 43ab—The format of the Data field of Unicast Key Negotiation Responding packet**

—The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Here, only bits 4 (USK Rekeying) is significant. When the unicast key negotiation procedure is for the initial USKSA establishment, bit 4 (USK Rekeying) is set to 0. When the unicast key negotiation procedure is for USK rekeying, bit 4 (USK Rekeying) is set to 1

— The BKID field is 16 octets in length, defined in 8.1.4.1.1. It indicates the current shared BK .

— The USKID field is 1 octet in length in which the bit 0 is the index of the unicast session key of the negotiation, and all the other bits are reserved. Bit 0 of this field is originally set to 0, and it switches between 0 and 1 after each unicast key negotiation.

— The ADDID field is 12 octets in length, defined in 8.1.4.1.1. MAC address 1 in the field is the AE's MAC address, and MAC address 2 is the ASUE's MAC address.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— The AE Challenge field is 32 octets in length. If the USK Rekeying is 0, it is equal to the one in the Unicast Key Negotiation Request packet. If the USK Rekeying is 1, it is equal to the one negotiated in the last unicast key negotiation procedure.

— ASUE Challenge field is 32 octets in length which is a random number generated by the ASUE.

— $WIE_{ASUE}$ field is the WAPI parameter set chosen by the ASUE's SME. In the BSS mode, the field is equal to the WAPI parameter set sent by the association request frame from ASUE. In the IBSS mode, the field contains the unicast cipher algorithm chosen by ASUE, the multicast cipher algorithm announced by AE, and the currently used authentication and key management suite list.

— The Message Authentication Code field is 20 octets in length. Its value is generated by ASUE through the HMAC_SHA256 algorithm. The ASUE calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field. The HMAC algorithm is defined in IETF RFC2104, HMAC: Keyed-Hashing for Message Authentication.

When the ASUE rekeys, or receives the Unicast Key Negotiation Request, the ASUE sends the Unicast Key Negotiation Response packet to the AE.

After the AE receives the Unicast Key Negotiation Response packet,

a) If the bit 4 (USK Rekeying) of the FLAG field is 1, the AE proceeds to Step b); otherwise the AE goes to Step c).

b) If there is a valid USKSA and the USKSA referred by USKID in the packet is invalid, the AE proceeds to Step c); otherwise the AE discards the packet.

c) If the AE Challenge is correct, the AE proceeds to Step d). Otherwise the AE discards the packet.

d) The AE generates 96 octets through the formula KD_HMAC_SHA256 (BK, ADDID|| N1||N2||"pairwise key expansion for unicast and additional keys and nonce") in which N1 is the AE Challenge and N2 is the ASUE Challenge. The first 64 octets are the unicast session key (the first 16 octets are the unicast encryption key, the second 16 octets are the unicast integrity check key, the third 16 octets are KCK, and the fourth 16 octets are KEK). The last 32 octets are the AE Challenge seed of the next unicast session key negotiation procedure. By a SHA256 computation on the seed, a 32-octet AE Challenge used in the next unicast session key negotiation procedure is obtained. The AE calculates the Message Authentication Code locally by the HMAC_SHA256 algorithm. If the calculated value matches that included in the packet, the AE proceeds to Step e); otherwise the AE discards the packet.

e) If the bit 4 (USK Rekeying) of the FLAG field is 1, the AE proceeds to Step f). Otherwise, If the bit 4 (USK Rekeying) of the FLAG field is 0,

— In BSS, if the $WIE_{ASUE}$ field is equal to the WAPI parameter set in the received association request frame, the AE proceeds to Step f); otherwise the AE delinkverifies with $STA_{ASUE}$.

— In IBSS, if the unicast key algorithm in $WIE_{ASUE}$ field is supported, the AE proceeds to Step f); otherwise the AE delinkverifies with $STA_{ASUE}$.

f) The AE calculates Message Authentication Code locally by the HMAC_SHA256 algorithm, and sends the Unicast Key Negotiation Confirmation packet to the ASUE.

g) The AE will install the new unicast session key when it is in BSS. When the AE is in IBSS, it will install the new unicast session key only if its MAC address is greater than the ASUE's MAC address.



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The AE should receive and send the unicast data frame with the new unicast session key. For unicast rekeying, the old unicast session key should be deleted if the received unicast data frame is decapsulated correctly with the new key.

#### 8.1.4.3.3 Unicast Key Negotiation Confirmation

The Unicast Key Negotiation Confirmation packet is illustrated in Figure 43ac and described after the figure.

| | FLAG | BKID | USKID | ADDID | ASUE Challenge | $WIE_{AE}$ | Message Authentication Code |
|---|---|---|---|---|---|---|---|
| **Octets:** | 1 | 16 | 1 | 12 | 32 | variable | 20 |

**Figure 43ac—The format of the Data field of Unicast Key Negotiation Confirmation packet**

— The FLAG field is 1 octet in length, and it is equal to the FLAG field in the Unicast Key Negotiation Response packet.
— The BKID field is 16 octets in length, and it is equal to the BKID field in the Unicast Key Negotiation Response packet.
— The USKID field is 1 octet in length, and it is equal to the USKID field in the Unicast Key Negotiation Response packet.
— The ADDID field is 12 octets in length, and it is equal to the ADDID field in the Unicast Key Negotiation Response packet.
— The ASUE Challenge field is 32 octets in length, and it is equal to the ASUE Challenge field in the Unicast Key Negotiation Response packet.
— The $WIE_{AE}$ field is the WAPI parameter set sent by AE in the Beacon frame and Probe Response frame.
—The Message Authentication Code field is 20 octets in length. Its value is generated by AE through the HMAC_SHA256 algorithm. The AE calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

After the AE receives the Unicast Key Response packet, it will send the Unicast Key Negotiation Confirmation packet to the ASUE.

After the ASUE receives the Unicast Key Negotiation Response packet

a) If the ASUE challenge is equal to the value in the Unicast Key Negotiation Response packet, the ASUE proceeds to Step b). Otherwise the ASUE discards the packet.
b) The ASUE calculates the Message Authentication Code locally by the HMAC_SHA256 algorithm. If the calculated value matches that included in the packet, the ASUE proceeds to Step c); otherwise the ASUE discards the packet.
c) If the bit 4 (USK Rekeying) of the FLAG field is 1, the AE proceeds to Step d). Otherwise, if the bit 4 (USK Rekeying) of the FLAG field is 0, and if the $WIE_{AE}$ field is equal to the WAPI parameter set in the received Beacon and Probe Response frame, the ASUE proceeds to Step d); otherwise the ASUE



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

delinkverifies with the $STA_{AE}$.

d) The sent unicast data frame will be encapsulated with the new key. For unicast rekeying, the old key should be deleted.

#### 8.1.4.4 Multicast key / STAKey announcement process

The keys negotiated in the unicast key negotiation process are used to achieve the multicast key / STAKey announcement and construct MSKSA or STAKeySA.

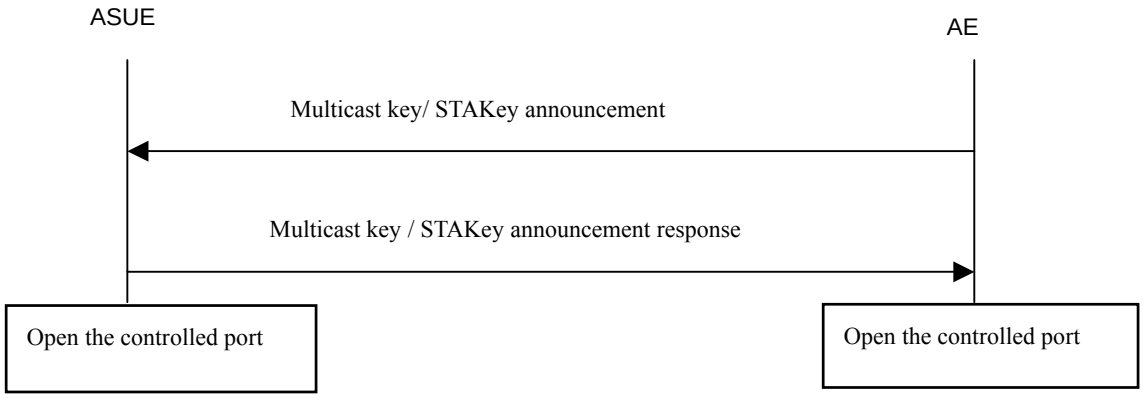



**Figure 43ad—Multicast key / STAKey announcement procedure**

##### 8.1.4.4.1 Multicast key / STAKey announcement packet

After the success of the unicast key negotiation, or when AE needs to update the multicast key, or when AE receives the STAKey request packet, AE should send ASUE the multicast key / STAKey announcement packet to announce the multicast /STAKey master key.

The multicast key / STAKey announcement packet is illustrated in Figure 43ae and described after the figure.

| | FLAG | MSKID/STAKeyID | USKID | ADDID | Data packet number | Key Announcement Identifier | Key Data | Message Authentication Code |
|---|---|---|---|---|---|---|---|---|
| **Octets**: | 1 | 1 | 1 | 12 | 16 | 16 | variable | 20 |

**Figure 43ae—The format of the Data field of Multicast key / STAKey announcement packet**

Here:

— The FLAG field is 1 octet in length, defined as before, and bits 5 and 6 are significant. When the announced key is STAKey, Bit 5 (STAKey negotiation) is set to 1; otherwise it is set to 0. If Bit 6 (STAKey key deletion) is 1, it means to delete the STAKey determined by USKID and ADDID; otherwise it is 0.

— The MSKID/ STAKeyID field is 1 octet in length, in which Bit 0 indicates the currently announced key and all the other bits are reserved. In this field, Bit 0 is initially set to 0, and it will switch between 0 and 1 after each announcement key update.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— The USKID field is 1 octet in length, in which Bit 0 indicates the message authentication key used to calculate the value of the Message Authentication Code field.

— The ADDID field is 12 octets in length. When the announced key is STAKey, the value of this field is: the initiator MAC's address || the peer MAC's address. When announced key is the multicast key, the value of this field is $MAC_{AE}||MAC_{ASUE}$.

— The Data packet Number field is 16 octets in length, and it indicates the serial number (PN in the WPI data packet) of the data packet that have been encrypted and sent. Hereafter, the serial number of the data frame that STA receives should be larger than the value of this field; otherwise the frame will be discarded.

— The Key Announcement Identifier field is 16 octets in length, and its initial value is 0x5C365C365C365C365C365C365C365C36. During every key updating announcement procedure, the value of this field will increase by 1. If the announced key is unchanged, then the value of this field should keep unchanged. When AE finds that the value of the Key Announcement Identifier field has overflowed, it will delinkverify with all related STAs. This field is also used as the IV of the key announcement data.

— The Key Data field is defined as before. Its content is the multicast master key or STAKey master key(no MIC) encrypted with the KEK by the negotiated unicast cipher algorithm. The multicast master key or STAKey master key is a 16-octet random number generated by AE, and the IV is the Key Announcement Identifier field.

— The Message Authentication Code field is 20 octets in length. It is calculated by AE through the HMAC_SHA256 algorithm. AE calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

After the ASUE receives the multicast key / STAKey announcement packet sent by AE it does as follows:

a) If bit 5 of the FLAG field (STAKey negotiation ) is 1, the packet is a STAKey announcement packet; otherwise it is a multicast key announcement packet. If ASUE does not support or not permit STAKey, then when bit 5 of the FLAG field (STAKey negotiation) is 1, this packet is discarded.
   ASUE uses the message authentication key identified by USKID field to calculate a check value, and compares it with the Message Authentication Code field value. If they are equal, goes to b). Otherwise, this packet is discarded.

b) Checks whether the value of the Key Announcement Identifier field value increases monotonically. If then, goes to c); otherwise, this packet is discarded.

c) Gets a 16-octet announcement master key by decrypting the key data. Then, through KD_HMAC_SHA256 algorithm the ASUE uses the master key to generate a 32-octet session key (the leading 16 octets are used as the encryption key, and the following 16 octets are used as the integrity check key). If bit 5 of the FLAG field (STAKey negotiation) is 1, then the above key is the STAKey; otherwise, it is the multicast key.

d) Saves the value of the Key Announcement Identifier field, then constructs the multicast key / STAKey response packet and sends it to AE.

e) Installs or deletes the key.
   If this procedure is the multicast key announcement procedure, the new MSK will be installed to receive the multicast data frame. For multicast rekeying, the old MSK should be deleted when a



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

multicast data frame is decapsulated correctly with the new key.

If this procedure is the STAKey announcement procedure, the initiator/peer will install the new key to send/receive the STA-to-STA data frame. For STAKey rekeying, the old key should also be deleted by the initiator immediately or by the peer when a STA-to-STA data frame is decapsulated correctly with the new key.

If this procedure is the STAKey deletion procedure, the peer will delete the corresponding STAKey.

If AE announces the new multicast key, ASUE will save the old multicast key. When it receives the multicast data frame, it will choose the multicast decryption key according to the KeyID field. Once ASUE receives the multicast data frame from AE with the last announced multicast key, and verifies as well as decrypts it correctly, the ASUE should discard the old multicast key.

#### 8.1.4.4.2 Multicast key / STAKey response packet

The authentication packet is illustrated in Figure 43af.

| | FLAG | MSKID/ STAKeyID | USKID | ADDID | Key Announcement Identifier | Message Authentication Code |
|---|---|---|---|---|---|---|
| **Octets**: | 1 | 1 | 1 | 12 | 16 | 20 |

**Figure43af—The format of the Data field of the multicast key/STAKey response packet**

Here:

— The FLAG field is 1 octet in length, which is defined as before. This field should be equal to the FLAG field in the multicast key / STAKey announcement packet sent by AE.

— The MSKID/ STAKeyID field is 1 octet, and its value is equal to the one in the multicast key / STAKey announcement packet sent by AE.

— The USKID field is 1 octet in which bit 0 indicates the message authentication key used to calculate the Message Authentication Code field. This field is equal to the USKID field in the multicast key / STAKey announcement packet sent by AE.

— The ADDID field is 12 octets in length, and its value is equal to the one in the multicast key / STAKey announcement packet.

— The Key Announcement Identifier field is 16 octets in length, and its value is equal to the one in the multicast key / STAKey announcement packet sent by AE.

— The Message Authentication Code field is 20 octets in length. Its value is generated by ASUE through the HMAC_SHA256 algorithm. The ASUE calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

ASUE sends AE the multicast key / STAKey response packet.

After the AE receives the multicast key response packet sent by ASUE, it does as follows:

a) Uses the message authentication key identified by the USKID field to calculate the check value, and



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

then compares it with the value of the Message Authentication Code field. If they are equal, goes to b). Otherwise, discards the packet.

b) Compares the FLAG field, the MSKID/STAKeyID field, the USKID field, the ADDID field and the Key Announcement Identifier field with those in the multicast key/STAKey announcement packet, respectively. If every field is equal to the corresponding field in the announcement packet, then the multicast key/STAKey announcement process is successful. Otherwise,AE discards this packet.

c) Installs the new key when announcing successfully.
If this procedure is the multicast key announcing procedure, the AE will send multicast data frame encrypted with the new key. When rekeying enabled, the old key should be deleted.
If this procedure is for the STAKey initiator, the AE will relay the STA-to-STA data frame from initiator to the peer transparently without encrypting and decrypting them.

During the multicast rekeying procedure, AE uses the old multicast key to encrypt the multicast data frame and send it. AE will not use the last announced multicast key until all of the STAs associated with the AP have been successfully announced with the new key.

#### 8.1.4.5 STAKey establishment procedure

When STA1 sends data to STA2 in the same BSS, STA1 can require to establish STAKey with STA2 to encrypt the unicast data from STA1 to STA2. The encryption/decryption algorithm is the multicast cipher suite announced by AP. AP will do not encrypt/decrypt the data sent from STA1 to STA2.

This procedure is mostly composed of the multicast key / STAKey announcement procedure and is showed in Figure 43ag.

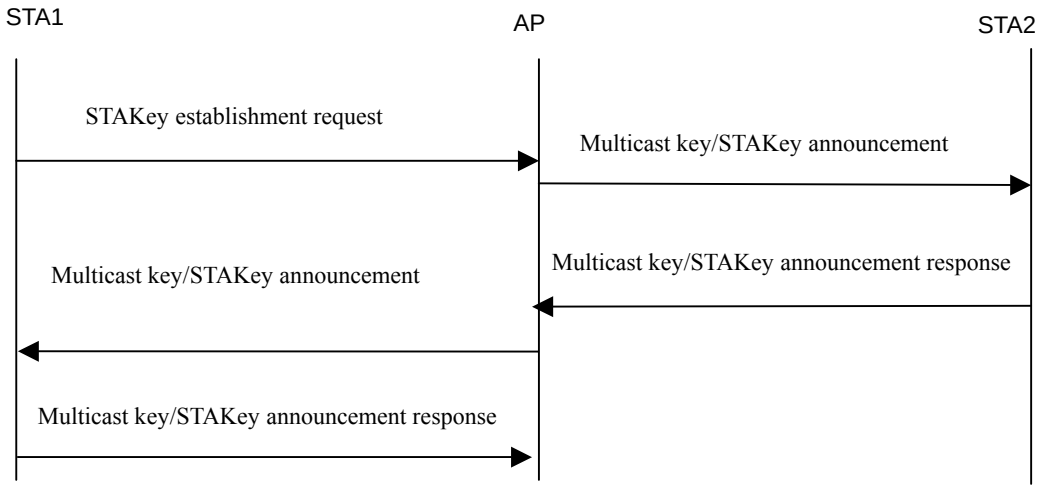



**Figure 43ag—The flow chart of the STAKey establishment**

STA1 firstly sends AP the STAKey establishment request. Then AP uses the random number generation algorithm to generate the STAKey master key and transfers it to STA1 and STA2 through the multicast key / STAKey announcement procedure.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

If AP has announced the STAKey to STA2 successfully, but it is unsuccessful to STA1, then AP should notify STA2 to delete the corresponding STAKey through the multicast key / STAKey announcement procedure, and AP should set the STAKey deletion flag to 1.The deletion flag is in the FLAG field of the multicast key / STAKey announcement packet.

If STA1 delete some STAKey, it should set the STAKey deletion flag in the FLAG field of the STAKey establishment request packet to 1.

The STAKey establishment request packet is illustrated in Figure 43ah.

| | FLAG | STAKeyID | USKID | ADDID | Replay Counter | Message Authentication Code |
|---|---|---|---|---|---|---|
| **Octets:** | 1 | 1 | 1 | 12 | 16 | 20 |

**Figure 43ah—The format of the Data field in the STAKey establishment request packet**

Here:

— The FLAG field is 1 octet in length, and is defined as before. Here, bits 5 and 6 are significant, and bit 5 (STAKey negotiation) is set to 1. When establishing the STAKey, bit 6(STAKey deletion ) is set to 0; and when deleting the STAKey, bit 6 is set to 1.

— The STAKeyID field is 1 octet in length, and is set to the key index value chosen by the initiator.

— The USKID field is 1 octet in length, in which bit 0 indicates the message authentication key used to calculate the Message Authentication Code field. The value of the USKID field should be equal to the one in the multicast key announcement packet sent by AP.

— The ADDID field is 12 octets in length, and its value is the initiator's MAC address || the peer's MAC address.

— The Replay Counter field is 16 octets in length, with the initial value 1. During each rekeying announcement, its value increases by 1.

— The Message Authentication Code field is 20 octets in length. Its value is generated by STA through the HMAC_SHA256 algorithm. STA calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

### 8.1.4.6 Pre-authentication

If the destination AP supports the pre-authentication, STA can execute the pre-authentication with the destination AP through the currently associated AP. STA requires to start the pre-authentication by sending a pre-authentication start packet to the destination AP. When the currently associated AP receives the pre-authentication start packet, if the Replay Counter and the Message Authentication Code are valid, then it forwards the packet to the destination AP. When the destination AP receives this packet, it can start the WAI certificate authentication procedure by sending the authentication activation packet to the corresponding STA identified by the ADDID field.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The pre-authentication start packet is illustrated in Figure 43ai .

| | FLAG | USKID | ADDID | Replay Counter | Message Authentication Code |
|---|---|---|---|---|---|
| **Octets:** | 1 | 1 | 2 | 16 | 20 |

**Figure 43ai—The format of the Data field in the pre-authentication start packet**

Here:

— The FLAG field is 1 octet in length, and is defined as before.

— The USKID field is 1 octet in length, where bit 0 indicates the message authentication key used in calculating the Message Authentication Code field. The value of this field is equal to the one in the USKID field of the multicast key announcement packet sent by AP.

— The ADDID field is 12 octets in length, and its value is the initiator STA's MAC address || the destination AP's MAC address.

— The Replay Counter field is 16 octets in length, with the initial value 1. Each time the STA sends the pre-authentication start packet the value of this field increases by 1. This Replay Counter field is same as that in the STAKey establishment request packet.

— The Message Authentication Code field is 20 octets in length. Its value is generated by STA through the HMAC_SHA256 algorithm. STA calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

After the STA finishes the negotiation and the unicast session key installation with the currently associated AP, the STA can start the pre-authentication procedure with other APs. If the destination AP declares to support the pre-authentication in the WAPI Capability Information field of the WAPI Information Element, the STA can start the pre-authentication procedure with this AP. Only if the certificate authentication and key management method is adopted, the pre-authentication procedure can be started.

The ASUE of STA activates the authentication procedure by sending a WAI "pre-authentication start" packet to the associated AP. In this packet, the DA is set to the destination AP's BSSID and the RA is set to the currently associated AP's BSSID. The destination AP should use BSSID as AE's MAC address. The currently associated AP should forward the pre-authentication packet sent by STA to DS, and send the pre-authentication packet sent by DS to STA.

The destination AP that receives the "pre-authentication start" packet through DS should start the authentication procedure with the corresponding STA. DS should forward the authentication message to the associated AP.

If the WAI certificate authentication procedure succeeds, the result of the pre-authentication is BKSA. If STA has been associated to the pre-authenticated AP, STA can use BKSA to start the unicast key negotiation procedure and multicast key announcement procedure. If the cached status of the pre-authentication result between AP and STA is not synchronized, the unicast key negotiation procedure will fail. In this situation, the value of dot11WAPIStatsWAIUnicastHandshakeFailures MIB will increase by 1.



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Whether the destination AP is in the signal range of STA or not, if only the AP supports pre-authentication, the ASUE of the STA can execute pre-authentication procedure with the AP.

Even if the pre-authentication procedure has already succeeded, the STA may require to execute the WAI certificate authentication and key negotiation procedure after associated to the destination AP. Because of the limitation of the resources or timeout, AP can possibly delete the pre-authentication result, BKSA.

NOTE: All WAI packets (including all packets in the certificate authentication, the unicast key negotiation, the multicast key announcement and the STAKey establishment procedures) are transferred by 0x88B4 protocol. If the format of a packet is incorrect, the MIB value dot11WAPIStatsWAIFormatErrors should be increased by 1; if the result of the signature verification is incorrect, the MIB value dot11WAPIStatsWAISignatureErrors should be increased by 1; if the result of the message authentication code check is incorrect, the MIB value dot11WAPIStatsWAIHMACErrors should be increased by 1.

#### 8.1.4.7 Cached BKSAs and WAPI key management

A STA can cache BKSAs between STA and AP. The BKSAs cannot be changed when being cached.

If a STA in an ESS has a valid BKSA with an AP to be (re) associated, it includes the BKID of the BKSA in the WAPI Information Element of the (Re) Association Request. After receives a (Re) Association Request with one or more BKIDs, an AP checks whether it has cached a BKSA for the BKID and whether the BKSA is still valid. If so, it starts the unicast session key negotiation procedure and the multicast key announcement procedure; otherwise it starts a full authentication and key negotiation procedure after association.

If a STA roams to an AP which has not completed the pre-authentication, and the STA does not have a BKSA for that AP, the STA should start a full WAI certificate authentication and key negotiation procedure.

#### 8.1.4.8 Rekeying

AP can start the updating of the base key, the unicast key and the multicast key updating, and STA can start the updating of the base key and the unicast key. The identified key index in rekeying process must be the currently invalid key index, and the old key must be deleted after the new key is activated.

Once the base key is updated, the unicast key must be updated as well. The process from the base key update beginning to the unicast key update completing must be completed within the time set by the MIB value dot11WAPIConfigSATimeout, otherwise the two STAs will be delinkverified.

During the unicast rekeying procedure, there is no WAPI parameter set confirmation or unicast ciper algorithm selector, and the STA will still use the unicast cipher algorithm negotiated in the initial unicast cipher negotiation procedure.

#### 8.1.4.9 Timeout processing

After STA and AP send the message, if they cannot receive any response within a timeout, then increase the MIB value dot11WAPIStatsWAITimeoutCounters by 1.
If the message is a certificate authentication packet, then re-send the message



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

"dot11WAPIConfigCertificateUpdateCount" times.
If the message is a unicast key negotiation packet, then re-send the message "dot11WAPIConfigUnicastUpdateCount" times.
If the message is a multicast key / STAKey announcement packet, then re-send the message "dot11WAPIConfigMulticastUpdateCount" times.
The default timeout value of the certificate authentication request packet is set to 10 seconds, that of the access authentication request packet is 31 seconds, and others are 1 second.

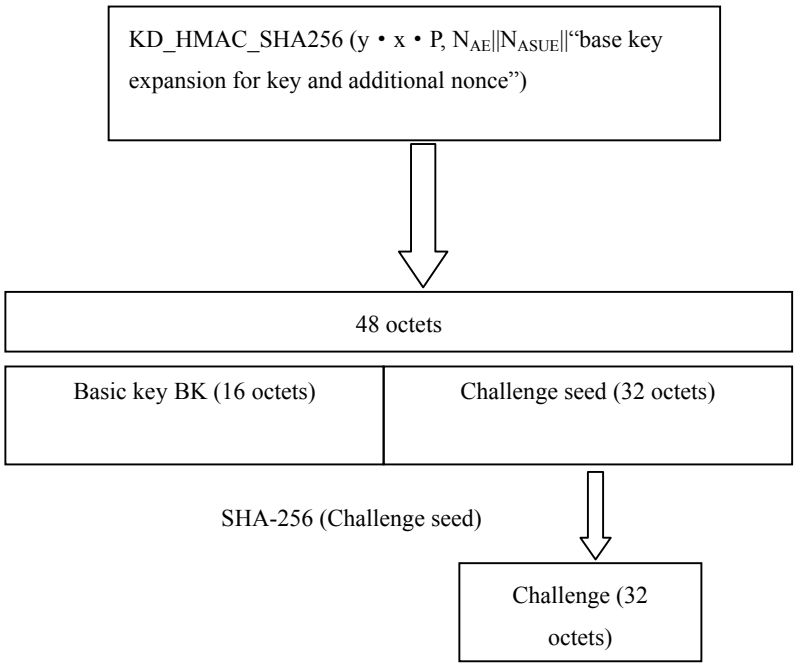
During the certificate authentication procedure, the unicast key negotiation procedure and multicast key announcement procedure, if there is still no response after re-sending the message certain times, then STA and AP will terminate the linkverification.
If the procedure is a certificate authentication procedure, then increases the MIB value dot11WAPIStatsWAICertificateHandshakeFailures by 1.
If the procedure is a unicast key negotiation procedure, then increases MIB value dot11WAPIStatsWAIUnicastHandshakeFailures by 1.
If the procedure is a multicast key announcement procedure, then increases the MIB value dot11WAPIStatsWAIMulticastHandshakeFailures by 1.

During the STAKey announcement procedure, if there is still no response after re-sending the message certain times, then increases the MIB value dot11WAPIStatsWAIMulticastHandshakeFailures by 1.

### 8.1.4.10 Key derivation architecture

In the WAI authentication and key management systems, the KD-HMAC-SHA256 is used to derive every key and challenge from the corresponding master key.

#### 8.1.4.10.1 BK key derivation architecture






--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

**Figure 43aj—BK key derivation architecture**

### 8.1.4.10.2 Unicast key derivation architecture

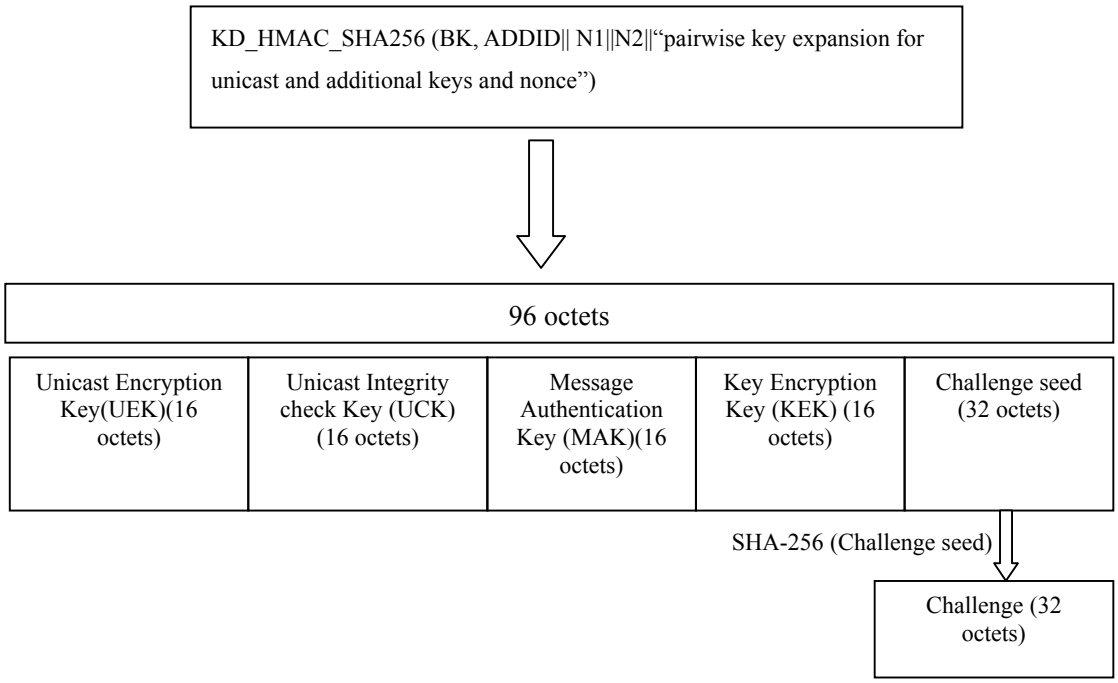



**Figure 43ak—Unicast key derivation architecture**

### 8.1.4.10.3 Multicast / STAKey derivation architecture

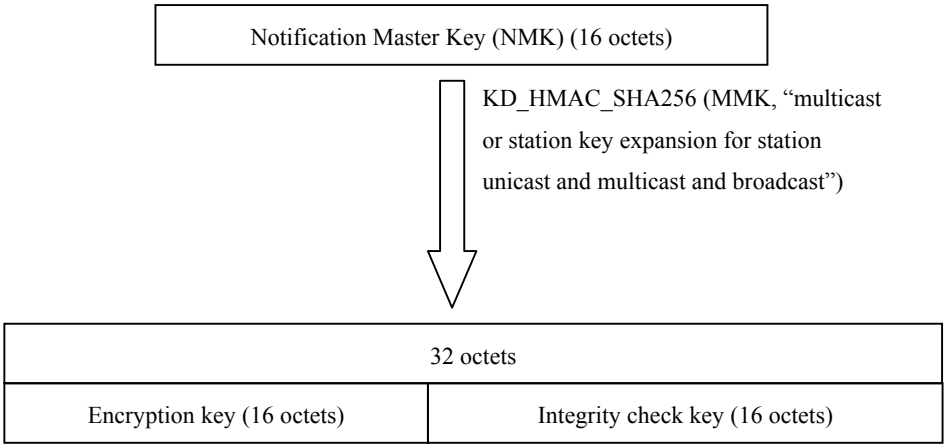



**Figure 43al—Multicast / STAKey derivation architecture**

### 8.1.4.10.4 Preshared key derivation architecture



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

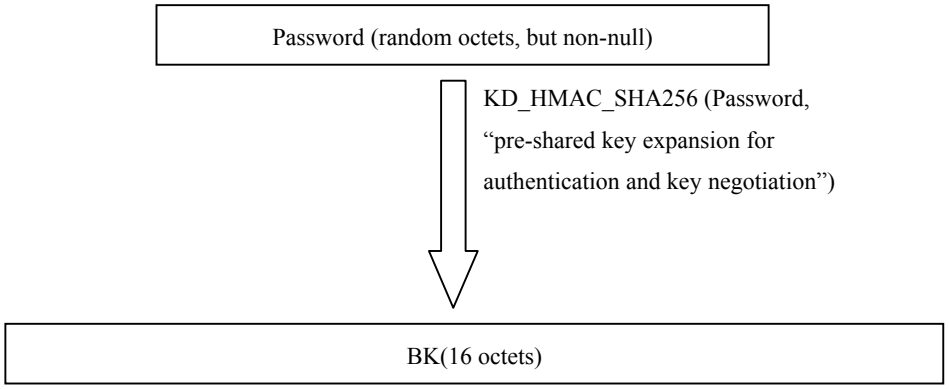



**Figure 43am—Preshared key derivation architecture**

### 8.1.4.11 WAI protocol packet's fragmentation and defragmentation

When STA or AP has a WAI protocol packet to send, it needs to determine which local port to send the packet, and then query this port to get its MTU (Maximum Transmission Unit). Comparing the packet length with the MTU, if the packet length is larger than the MTU, the sender has to fragmentate the packet. Each fragment has its own WAI protocol header. All fragments except the last one should have the same length, and their length should be a maximum value smaller than the MTU. Here we suggest this value is multiple of 8 octets.

When these fragments reach the destination receiver, it is necessary to reassemble them. The information contained in the WAI protocol fragment header is enough for the receiver to correctly reassemble these fragments into a packet.

The WAI protocol layer uses three fields in the protocol packet header to fragmentate and defragmentate.
For each WAI protocol packet from a sender, its WAI packet serial number field includes a unique value (the serial number of the first WAI protocol packet is 1, afterwards the serial number of the WAI protocol packet increases by 1), that value is copied to every fragment during the WAI protocol packet fragmentation. If a packet is a retransmitted one, its packet serial number keeps unchanged.
The fragment serial number field denotes the sequence number of the fragment (the serial number of the first fragment is 0, afterwards the serial number of the fragment increases by 1).
One bit in the fragment flag field is used to indicate "more fragments". In all fragments except the last one of the protocol packet, this bit should be set to 1; the last fragment of the protocol packet, this bit should be set to 0.
Moreover, when a protocol packet is fragmented, the length field of each fragment denotes the length of the fragment, i.e. the total length of the WAI protocol header and the fragment data (in octet).

Fragmentation and defragmentation should be completed in the WAI protocol layer, and the MAC layer should operate in the strict order mode in these processes. Whether the sender is STA or AP, it should send the packets according to the packet serial number from small to large in the protocol layer. When the fragmentation appears, the fragments of the same WAI protocol packet should be sent according to fragment serial number from small to large. During the receiving, if the receiver protocol layer has received all fragments of the same WAI protocol packet, it will reassemble them to recover the original protocol packet and process. If the received WAI protocol fragments have wrong sequence, discard all the fragments of the currently received WAI protocol packet. If the



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

received WAI protocol packet has wrong sequence, discard the last former WAI protocol packet.

The timeout re-transmission mechanism of each protocol packet in the WAI protocol (the certificate authentication and key negotiation)has been already defined, therefore it is not necessary to define each fragment timeout re-transmission mechanism, i.e., in a WAI protocol packet transmission procedure, even if there is only one fragment lost, the whole protocol packet should be re-transmitted.

#### 8.1.4.12 Port control and data transmission

The WAI controlled port status is *off* at the beginning. When the unicast key negotiation is finished, STA uses the MLME-SETWPIKEYS.request primitive to set the unicast key. When the multicast key announcement is finished, STA uses the MLME-SETWPIKEYS.request primitive to set the multicast key. At the same time, STA uses the MLME-SETPROTECTION.request primitive to start the WPI encrypting /decrypting function for unicast data and multicast data, and STA sets the WAI controlled port status to *on*.

During the updating of BKSA, USKSA and MSKSA, the WAI controlled port status is kept *on*.

Whether WPI is enabled or not, the WAI protocol packet will be transmitted in the plaintext mode.

### 8.2 WPI privacy infrastructure

WPI encrypts MPDUs (except the MPDUs of the WAI protocol packet) with the block cipher of 128-bit size and 128-bit key in the MAC sub-layer. Operation mode and encapsulation structure of the block cipher are illustrated as follow.

#### 8.2.1 Opertate mode

In WPI, the integrity check algorithm works in the CBC-MAC mode, and the data privacy uses the symmetric encryption algorithm operating in the OFB mode. The CBC-MAC and OFB mode is defined in ISO/IEC 10116: 1997 (2nd edition) Information technology - Security techniques - Modes of operation for an n-bit block cipher algorithm.

These two modes are illustrated as follow:



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

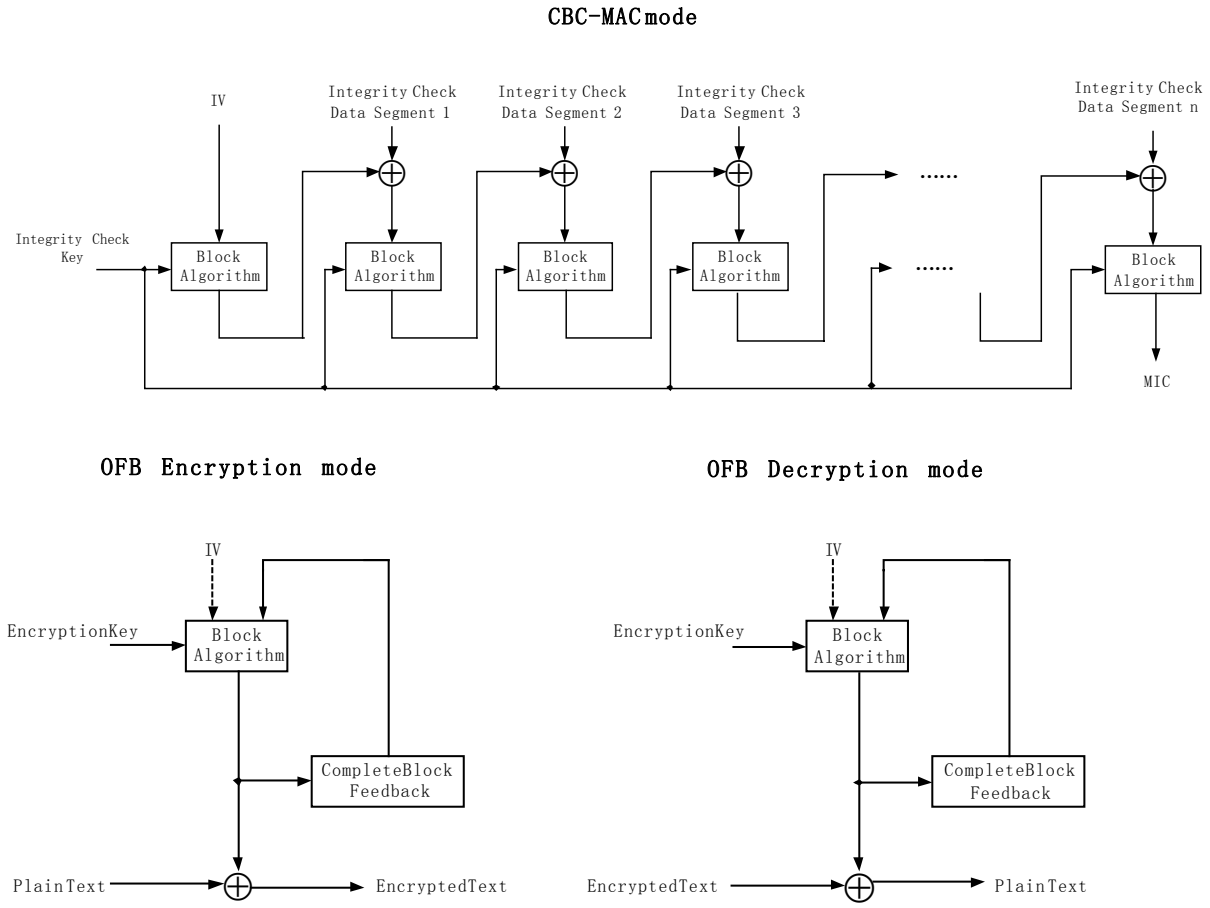



**Figure 43an—Operate modes**

### 8.2.2 Key

**Key management based on the certificate**

During the certificate authentication procedure, ASUE and AE firstly use the ECDH exchange and the KD_HMAC_SHA256 algorithm to negotiate a 16-octet BK. Then during the unicast key negotiation procedure, ASUE and AE respectively exchange a random number and use the KD_HMAC_SHA256 algorithm and the BK to generate 96 octets. The first 64 octets are the unicast session key (the first 16 octets are unicast encryption key, the second 16 octets are the unicast integrity check key, the third 16 octets are KCK, and the fourth 16 octets are KEK), the last 32 octets are the challenge seed for the next unicast session key negotiation procedure. Then use SHA-256 function, with the seed to obtain a 32-octet challenge for the next unicast key negotiation procedure, and save it. At last, during the multicast key/STAKey announcement procedure, ASUE and AE will respectively use the KD_HMAC_SHA256 algorithm to extend the 16-octet announcement master key to generate a 32-octet session key (the first 16 octets are the encryption key, and the last 16 octets are the integrity check key).

**Key management based on the preshared key**

In the preshared key mode, ASUE and AE will use the shared key as the seed to generate the BK. During the unicast key negotiation procedure, ASUE and AE respectively exchange a random number and use the KD_HMAC_SHA256 algorithm and the BK to generate 96 octets. The first 64 octets are the unicast session key (the first 16 octets are unicast encryption key, the second 16 octets are the unicast integrity check key, the third



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16 octets are KCK, and the fourth 16 octets are KEK), the last 32 octets are the challenge seed for the next unicast session key negotiation procedure. Then use the SHA-256 function, , with the seed to obtain a 32-octet challenge for the next unicast key negotiation procedure, and save it. At last, during the multicast key/STAKey announcement procedure, ASUE and AE will respectively use the KD_HMAC_SHA256 algorithm to extend the 16-octet announcement master key to generate a 32-octet session key (the first 16 octets are the encryption key, and the last 16 octets are the integrity check key).

NOTE: In BSS, the key management based on the certificate or the key management based on the preshared key can be adopted. In the IBSS mode, the key management based on the certificate or the key management based on the preshared key can be used, and the initiator STA and the peer STA are respectively acted as AE and ASUE, and each of them is required to be the initiator to complete a unicast key negotiation procedure and a multicast key announcement procedure.The unicast data key negotiated in the process initiated by STA with a larger MAC address in choosed as the data transmission key. In the WDS mode, only the key management based on the preshared key can be adopted, and its procedure is the same as that in IBBS mode. Currently, the WDS privacy communication supporting is optional

The MAC address is 6 octets. For the purposes of comparison, the MAC address is encoded as 6 octets, taken to represent an unsigned binary number. The first octet of the MAC address shall be used as the most significant octet.

When the user inputs the shared master key(password), it needs to support both the Hex mode and the ASCII Character mode.

### 8.2.3 Encapsulation and decapsulation

The encapsulation structure of WPI is as follow.

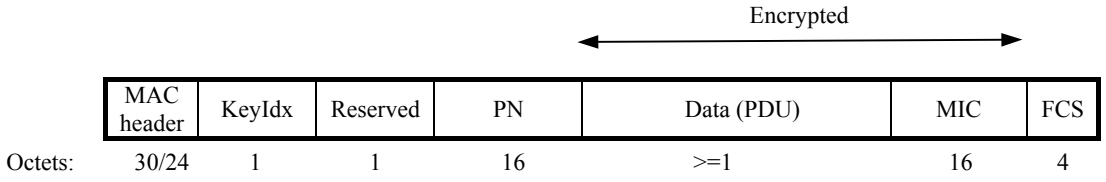



**Figure 43ao—WPI's MPDU encapsulation structure**

Here,

— The MAC Header field is 30 octets in length if the address **A4** exists, otherwise it is 24 octets in length.

— The KeyIdx field is one octet in length, and BSS indicates the USKID, MSKID, or STAKeyID..

— The Reserved field is one octet in length, with the default value 0.

—The PN field is 16 octets in length, and it indicates the serial number of the data packet.

—The PDU (data) field indicates the MPDU data, and its maximum length is 2278(2312-18-16).

—The MIC field is 16 octets in length, and its value is calculated over the integrity check data using the integrity check key in the CBC-MAC mode, the IV required for the calculation is the value of the PN field in the packet.

—The FCS field is 4 octets in length.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The integrity check data is defined as below:

| | FC | A1 | A2 | SC | A3 | A4 | KeyIdx | Reserved | L | Data (PDU) |
|---|---|---|---|---|---|---|---|---|---|---|
| **Octets:** | 2 | 6 | 6 | 2 | 6 | 6 | 1 | 1 | 2 | >0 |

**Figure 43ap—Integrity check data**

— The MAC Header field: the Frame Control field **FC** (Bits 4, 5, 6, 11, 12 and 13 are 0; bit 14 is1) with a length of 2 octets; address **A1** with the length of 6 octets; address **A2** with the length of 6 octets; address **A3** with the length of 6 octets; the Serial Control field **SC** (bits 4—15 are 0) with the length of 2 octets; address **A4** with the length of 6 octets. Altogether, they are 28 octets in length.

NOTE: if address **A4** does not exist in the MAC header, the corresponding 6 octets are set to 0.

— The KeyIdx field, with the length of one octet.

— The Reserved field, with the length of one octet, with the default value 0.

— The length **L** of the MPDU data, with the length of 2 octets.

— The MPDU data, with the length of more than 0 octet.

In the WPI, when using CBC-MAC to calculate MIC, if the integrity check data length is not multiple of 16 octets, then expend the integrity check data to the multiple of 16 octets by adding as less as possible at the end. The receiver will use the same method to verify.

During the data transmission, the WPI's MPDU encapsulation procedure is as follow, where "||" is the concatenation operation:

a) Using the integrity check key and the data packet serial number PN, calculate over the integrity check data to get the integrity check code MIC in CBC-MAC mode;
b) Using the encryption key and the data packet serial number PN, encrypt MPDU data || MIC to obtain the ciphertext in OFB mode;
c) Encapsulate and send it.

During the data reception, the WPI's MPDU decapsulation procedure is as follow:

a) Verify whether PN is valid. If it is invalid, then discard the data, and increase the MIB value dot11WAPIStatsWPIReplayCounters by 1;
b) Using the decryption key and PN, decrypt MPDU data || MIC ciphertext to recover the plaintext in OFB mode. If there is no valid decryption key, then discard the data and increase the MIB value dot11WAPIStatsWPIDecryptableErrors by 1;
c) Using the integrity check key and PN, locally calculate over the integrity check data to obtain the integrity check code MIC in CBC-MAC mode. If the calculated MIC is not equal to the received MIC, then discard the data and increase the MIB value dot11WAPIStatsWPIMICErrors by 1;
d) Decapsulate and reassemble the MPDU plaintext.

### 8.2.4 Rules for using data packet serial number PN

**In the unicast session**

After each unicast key update,



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ASUE initializes the value of PN as 0x5C365C365C365C 365C365C365C365C36. Before ASUE sends each unicast data frame, it should firstly increase the value of PN by 2.
AE initializes the value of PN as 0x5C365C365C365C365C365C365C365C37. Before AE sends each unicast data frame, it should firstly increase the value of PN by 2.

When ASUE receives the unicast data frame from AE, it should verify whether the value of PN corresponding to the USKID increases strictly monotonically and whether it is an odd number. If either not, then the packet is discarded.

When AE receives the unicast data frame from ASUE, it should verify whether the value of PN corresponding to the USKID increases strictly monotonically and whether it is an even number. If either not, then the packet is discarded.

AE can update the unicast key by the policy based on time or data packet amount. Besides, the overflow problem of PN value needs to be solved by the unicast key update initiated by AE.

**In the multicast session**
After each multicast key update, AE initializes the value of PN as 0x5C365C365C365C365C365C365C365C36. Before AE sends each multicast data frame, it should firstly increase the value of PN by 1.

When ASUE receives the multicast data frame, it should verify whether the value of PN corresponding to the MSKID increases strictly monotonically. If not, then discard the packet.

AE can update the unicast key by the policy based on time or data packet amount. Besides, the overflow problem of PN value needs to be solved by the unicast key update initiated by AE.

**In the STA-to-STA session**
After each STA-to-STA key establishment, the initiator of the STAKey negotiation initializes the value of PN as 0x5C365C365C365C365C365C365C 365C36. Before the initiator sends each unicast data frame to the peer part, it should firstly increase the PN value by 1.

When the peer STA receives the unicast data encrypted with the STAKey, it should verify whether the value of PN corresponding to STAKeyID increases strictly monotonically. If not, discard that packet.

The initiator of STAKey can update the STAKey by the policy based on time or the data packet amount. Besides, the overflow problem of the PN value needs to be solved by the STA-to-STA key update.

## 8.3 WAPI Authentication and key management state machine

### 8.3.1 WAPI ASUE Authentication and key management state machine

The ASUE shall reinitialize the ASUE state machine whenever its system is initialized. An ASUE enters the AUTHENTICATION state on an event from the MAC that requests another STA to be authenticated. If the MIC



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

or any of the WAI packets fails, the ASUE silently discards the packet. Figure 43aq and Figure 43ar depict the ASUE state machine.

The management entity will send an authentication request event when it requires an AE authenticated. This can be before or after the STA associates to the AP. In an IBSS environment, the event will be generated when a Probe Response frame is received.

Unconditional transfer (UCT) means the event triggers an immediate transition.

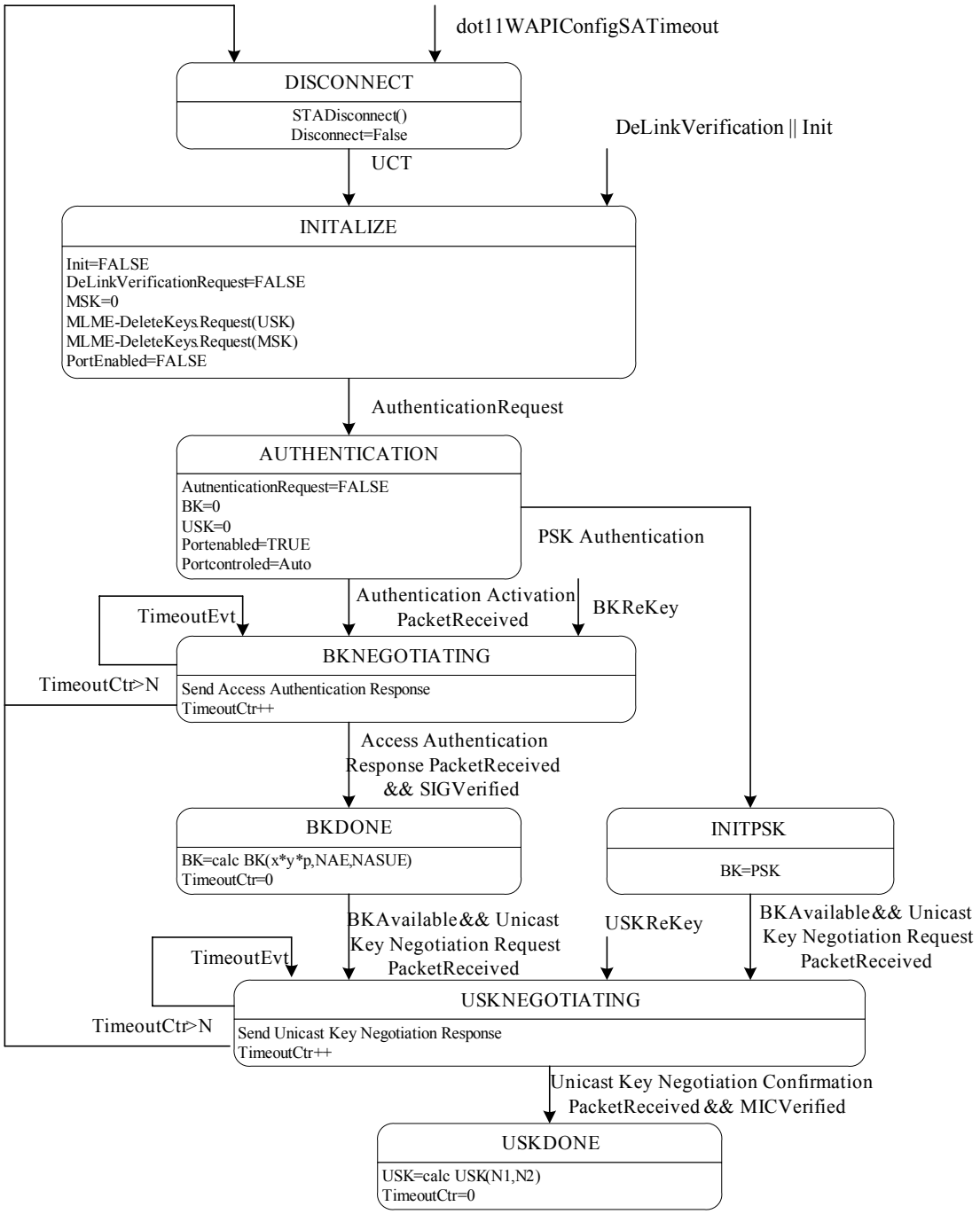



**Figure 43aq—ASUE state machine, Part 1**



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

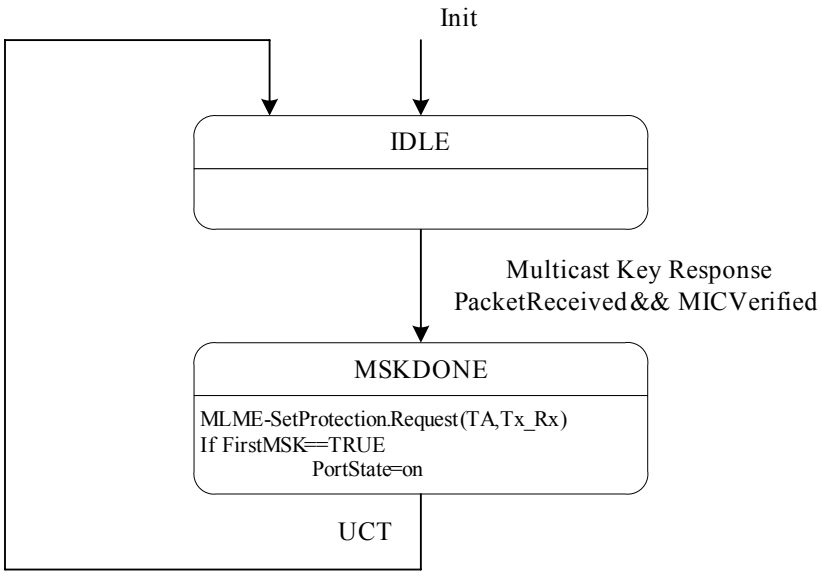



**Figure 43ar—ASUE state machine, Part 2**

### 8.3.1.1 ASUE state machine states

The following list summarizes the states of the ASUE state machine:

— **DISCONNECT:** A STA's ASUE entered this state when WAI procedure is timeout. The ASUE executes StaDisconnect and enters the INITIALIZE state.

— **INITIALIZE:** A STA's ASUE enters this state from the DISCONNECT state, when it receives Disassociation or DeLinkverification messages or when the STA initializes, causing the STA's ASUE to initialize the key state variables.

— **AUTHENTICATION**: This state is entered when an AuthenticationRequest is sent from the management entity to authenticate a BSSID.

— **INITPSK:** This state is entered when a PSK is configured.

— **BKNEGOTIATING:** This state is entered when the Authentication Activation packet is received or if BKReKey is set TRUE.

— **BKDONE:** This state is entered when the Access Authentication Response packet is received and signature is verified.

— **USKNEGOTIATING:** This state is entered from BKDONE to start the USK Negotiation when BKAvailable is set to TRUE and Unicast Key Negotiation Request packet is received or if USKReKey is set to TRUE.

— **USKDONE:** This state is entered when the Unicast Key Negotiation Confirmation packet is received and MIC is verified.

— **MSKDONE:** This state is entered when the Multicast Key Announcement packet is received from the AE and MIC is verified.

— **IDLE:** This state is entered when no Multicast Key Announcement is occurring.



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

#### 8.3.1.2 ASUE state machine variables

The following list summarizes the variables used by the ASUE state machine:

— *AuthenticationRequest* – This variable is set to TRUE if the STA's ISO/IEC 8802.11 management entity requires an association to be authenticated. This can be set when the STA associates or at other time.

— *DeLinkVerificationRequest* – This variable is set to TRUE if a Disassociation or DeLinkVerification message is received.

— *Disconnect* – This variable is set to TRUE when the STA should initiate a DeLinkVerification.

— *PacketReceived* – This variable is set to TRUE when a WAI packet is received.

— *Init* – This variable is used to initialize per-STA state machine.

— *TimeoutEvt* – This variable is set to TRUE if the WAI packet sent out fails to obtain a response from the AE. The variable may be set by management action or set by the operation of a timeout while in the USKNEGOTIATING and BKNEGOTIATING states.

— *TimeoutCtr* **–** This variable maintains the count of WAI packet receive timeouts. It is incremented each time a timeout occurs on a WAI packet receive event and is initialized to 0.

— *HMACVerified* – This variable is set to TRUE if the MIC on the received WAI packet is verified correct. Any WAI packets with an invalid HMAC will be dropped and ignored.

— *SIGVerified* – This variable is set to TRUE if the signature on the received WAI packet is verified correct. Any WAI packets with an invalid signature will be dropped and ignored.

— *USK* – This variable is the current USK.

— *MSK* – This variable is the current MSK.

— *BK* – This variable is the buffer holding the current BK.

— *BKReKey* – This variable is set to TRUE when a BK rekeying is required.

— *USKReKey* – This variable is set to TRUE when a USK rekeying is required.

—*Certificate_Verificaton*– This variable is set to TRUE when the certificate is required to be verified.

—PortControlled – This variable is the method by which the port is controlled. The value is Auto or ForceUnauthenticated.

—PortEnabled – This variable is set to TRUE if the port control mechanism is applied.

—PortState – This variable is set to on if the data frame is permitted to pass through the port.

—Certificate Authentication– This variable is set to TRUE if the certificate authentication and key management is chosen.

—PSK Authentication– This variable is set to TRUE if the PSK authentication and key management is chosen.

#### 8.3.1.3 ASUE state machine procedures

The following list summarizes the procedures used by the ASUE state machine:

— **STADisconnect**() – Execution of this procedure DeLinkVerifies the STA.

### 8.3.2 WAPI AE Authentication and key management state machine

There is one state diagram for the AE. In an ESS, the AE will always be on the AP; and in an IBSS environment,



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

the AE will be on every STA.

The state diagram shown in parts in Figure 43as through Figure 43awconsists of the following states:

a) The AUTHENTICATION, INITPSK, BKSTART, BKNEGOTIATING, BKNEGOTIATING2, BKNEGOTIATING3, BKDONE, USKSTART, USKNEGOTIATING, USKDONE, DISCONNECT, DISCONNECTED, and INITIALIZE states. These states handle the initialization, certificate authentication, unicast key negotiation, tear-down, and general clean-up. These states are per associated STA.
b) The IDLE, MSKNEGOTIATING, KEYERROR, MSKDONE states. These states handle the transfer of the MSK to the associated client. These states are per associated STA.
c) The MSK_INIT, SETKEYS, and SETKEYSDONE states. These states change the MSK when required, trigger all the USK multicast key state machines, and update the ISO/IEC 8802.11 MAC in the AE's AP when all STAs have the updated MSK. These states are global to the AE.

When the MSK is to be updated the variable MSKReKey is set. The SETKEYS state updates the MSK and triggers all the USK multicast key state machines that currently exist—one per associated STA. Each USK multicast key state machine sends the MSK to its STA. When all the STAs have received the MSK (or failed to receive the key), the SETKEYSDONE state is executed which updates the APs encryption/integrity engine with the new key.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

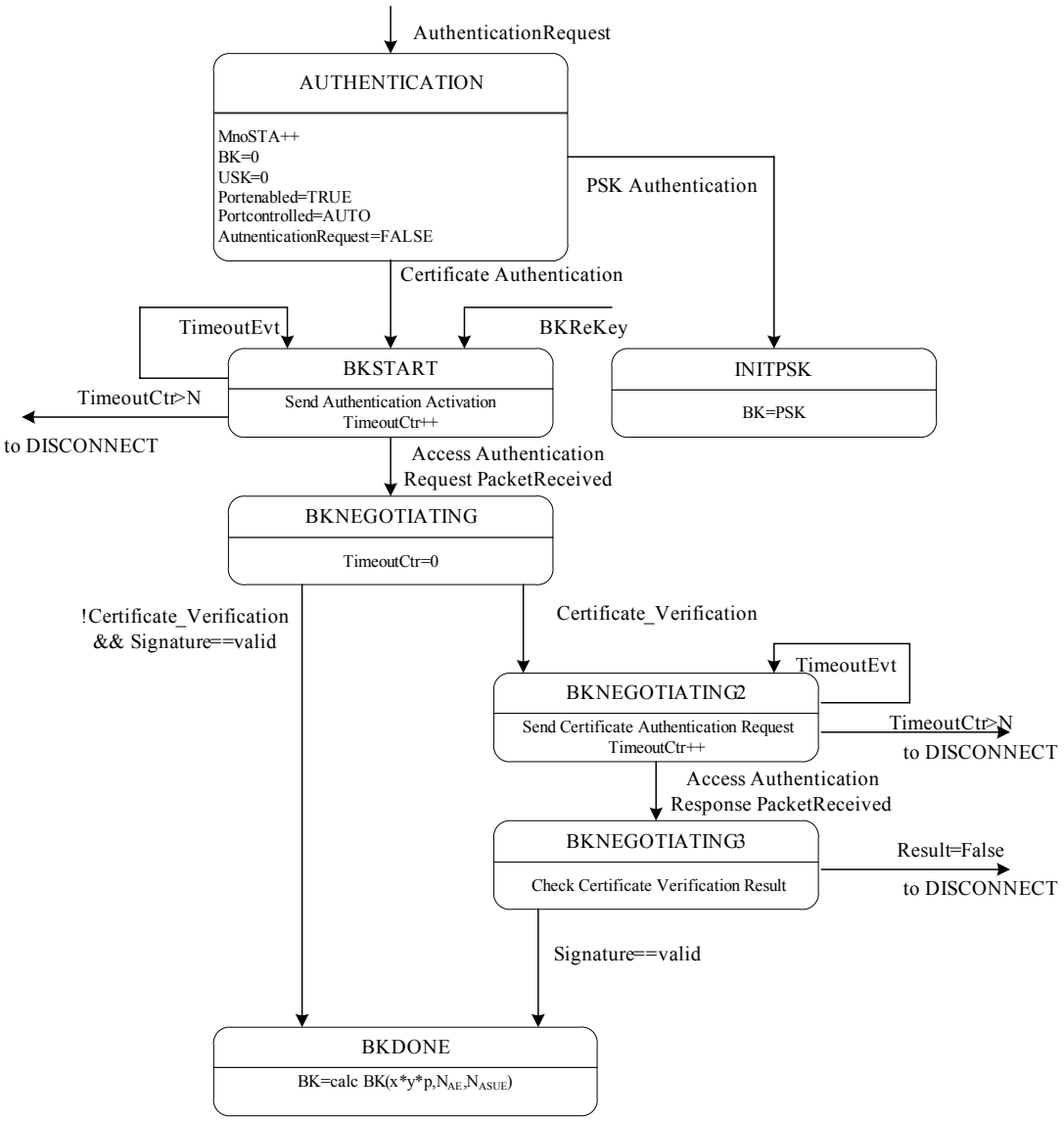



**Figure 43as—AE state machine, Part 1**



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

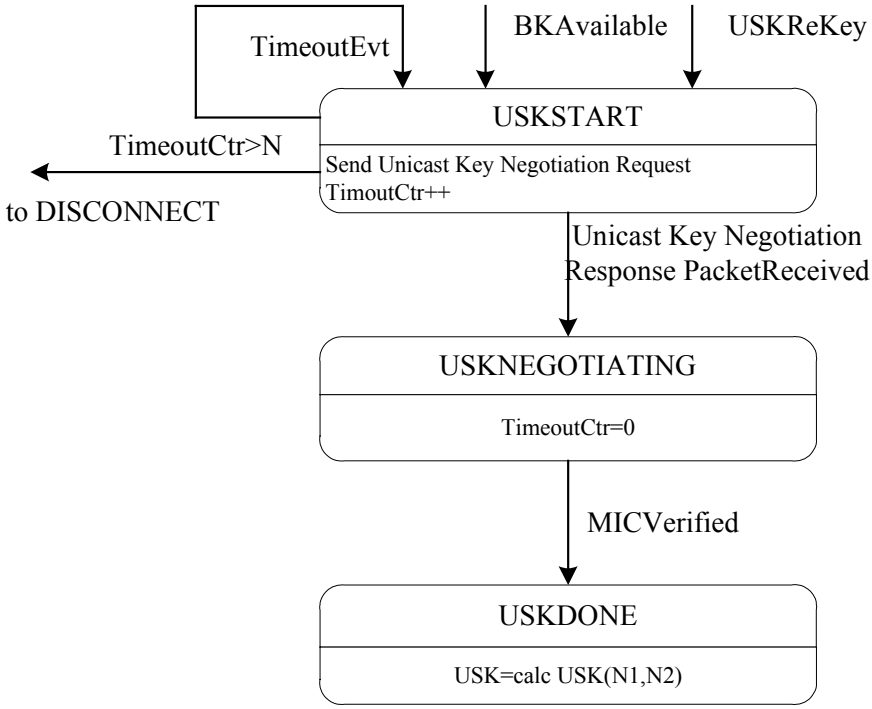



**Figure 43at—AE state machine, Part 2**

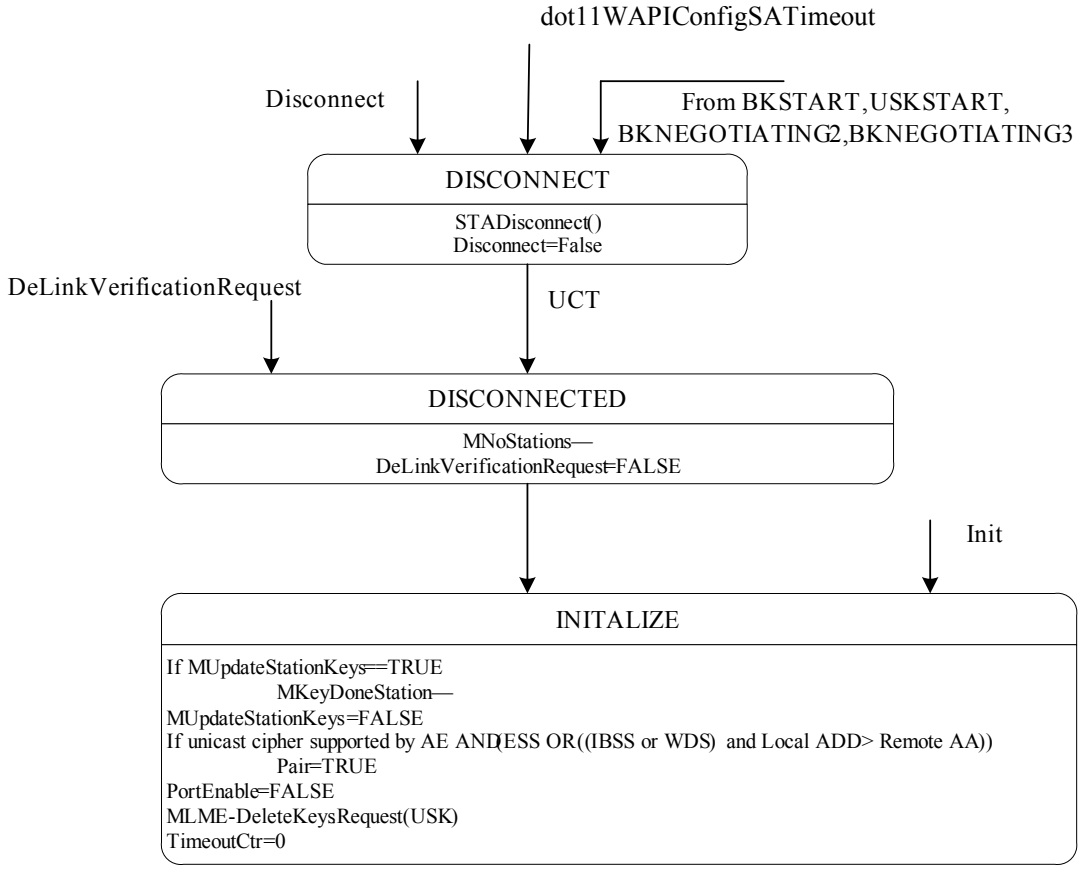



**Figure 43au—AE state machine, Part 3**



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

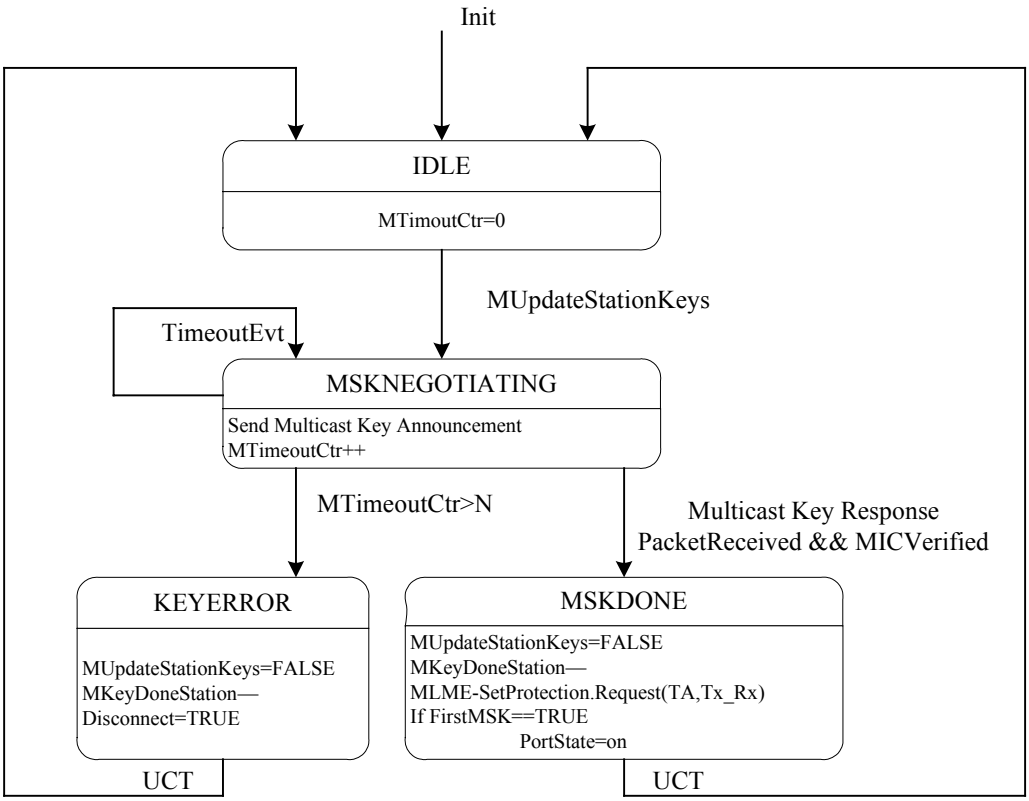



**Figure 43av—AE state machine, Part 4**

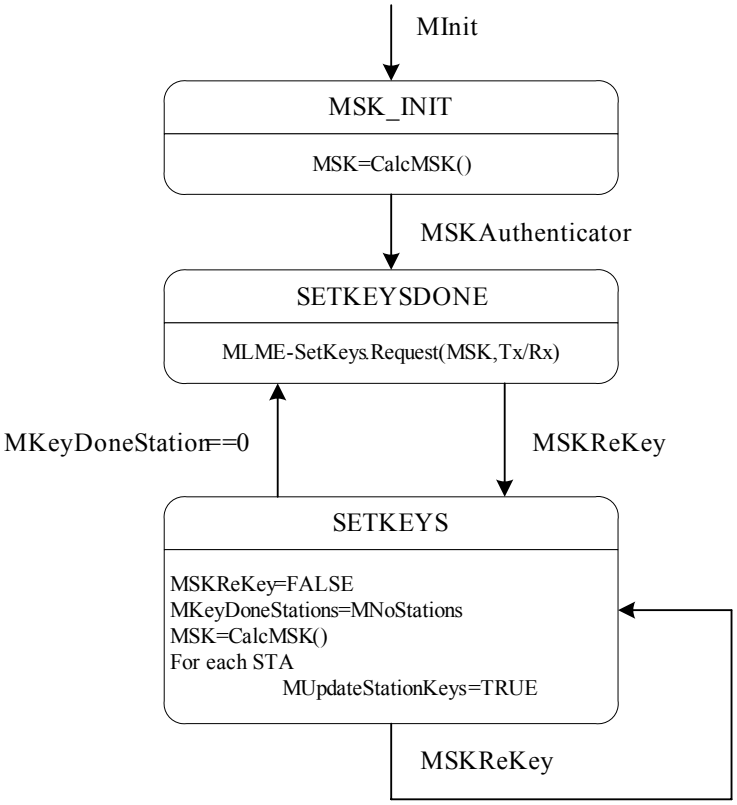



**Figure43aw—AE state machine, Part 5**



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

#### 8.3.2.1 AE state machine states

##### 8.3.2.1.1 AE state machine: Certificate Authentication and USK Negotiation (per STA)

The following list summarizes the states the AE state machine uses to support the Certificate Authentication and the USK Negotiation:

— **AUTHENTICATION**: This state is entered when an AuthenticationRequest is sent from the management entity to authenticate a BSSID.

— **BKSTART:** This state is entered from AUTHENTICATION to start the BK Negotiation or if BKReKey is set TRUE.

— **INITPSK:** This state is entered when a PSK is configured.

— **BKNEGOTIATING:** This state is entered when the Access Authentication Request packet is received.

— **BKNEGOTIATING2:** This state is entered when the certificate is required to be verified.

— **BKNEGOTIATING3:** This state is entered when the Certificate Authentication Response packet is received.

— **BKDONE:** This state is entered when the certificate is valid and the signature for Access Authentication Request packet is verified.

— **USKSTART:** This state is entered from BKDONE to start the USK Negotiation when BKAvailable is set TRUE or if USKReKey is set TRUE.

— **USKNEGOTIATING:** This state is entered when the Unicast Key Negotiation Response packet is received.

— **USKDONE:** This state is entered when the HMAC for the Unicast Key Negotiation Response packet is verified.

— **DISCONNECT**: This state is entered if Disconnect is set TRUE or WAI procedure is timeout. It sends a DeLinkVerificaton message to the STA and enters the INITIALIZE state.

— **DISCONNECTED**: This state is entered when Disassociation or DeLinkVerification messages are received.

— **INITIALIZE**: This state is entered from the DISCONNECTED state, when a DeLinkVerification request event occurs, or when the station initializes. The state initializes the key state variables.

##### 8.3.2.1.2 AE state machine: Multicast Key Announcement (per STA)

The following list summarizes the states the AE state machine uses to support the Multicast Key Announcement:

— **IDLE:** This state is entered when no Multicast Key Announcement is occurring.

— **KEYERROR:** This state is entered if the Multicast Key Response packet for the Multicast Key Announcement is not received.

— **MSKDONE:** This state is entered when the Multicast Key Response packet for the Multicast Key Announcement is received from the ASUE.

— **MSKNEGOTIATING:** This state is entered when the MSK is to be sent to the ASUE.



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

#### 8.3.2.1.3 Authenticator state machine: Multicast Key Announcement (global)

The following list summarizes the states the AE state machine uses to coordinate a multicast key update of all STAs:

— **MSK_INIT:** This state is entered on system initialization.
— **SETKEYS:** This state is entered if the MSK is to be updated on all ASUEs.
— **SETKEYSDONE:** This state is entered if the MSK has been updated on all ASUEs.

### 8.3.2.2 AE state machine variables

The following list summarizes the variables used by the Authenticator state machine:

— *AuthenticationRequest* – This variable is set to TRUE if the STA's ISO/IEC 8802.11 management entity requires an association to be authenticated. This can be set when the STA associates or at other time.
— *DeLinkVerificationRequest* – This variable is set to TRUE if a Disassociation or DeLinkVerification message is received.
— *Disconnect* – This variable is set to TRUE when the STA should initiate a deLinkVerification.
— *PacketReceived* – This variable is set to TRUE when a WAI packet is received.
— *MTimeoutCtr* – This variable maintains the count of a specific WAI packet receive timeouts for the Multicast Key Announcement. It is incremented each time a timeout occurs on a specific WAI packet receive event and is initialized to 0.
— *MInit* – This variable is used to initialize the multicast key state machine.
— *Init* – This variable is used to initialize per-STA state machine.
— *TimeoutEvt* – This variable is set to TRUE if the WAI packet sent out fails to obtain a response from the ASUE. The variable may be set by management action or set by the operation of a timeout while in the BKSTART, USKSTART, BKNEGOTIATING2 and MSKNEGOTIATING states.
— *TimeoutCtr* **–** This variable maintains the count of WAI packet receive timeouts. It is incremented each time a timeout occurs on a WAI packet receive event and is initialized to 0.
— *HMACVerified* – This variable is set to TRUE if the MIC on the received WAI packet is verified and is correct. Any WAI packets with an invalid HMAC will be dropped and ignored.
— *MSKAuthenticator* – This variable is set to TRUE if the AE is on an AP or it is the designated AE for an IBSS.
— *MKeyDoneStations* – Count of number of STAs left to have their MSK updated. This is a global variable.
— *MSKRekey* – This variable is set to TRUE when a Multicast Key Announcement is required. This is a global variable.
— *BKReKey* – This variable is set to TRUE when a BK rekeying is required.
— *USKReKey* – This variable is set to TRUE when a USK rekeying is required.
— *MUpdateStationKeys* – This variable is set to TRUE when a new MSK is available to be sent to ASUEs.
— *MNoStations* – This variable counts the number of AEs so it is known how many ASUEs need to be sent the MSK. This is a global variable.
— *USK* – This variable is the current USK.
— *MSK* – This variable is the current MSK.
— *BK* – This variable is the buffer holding the current BK.
—*Certificate_Verificaton*– This variable is set to TRUE when the certificate is required to be verified.



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

—PortControlled – This variable is the methods by which the port is controlled. The value is Auto or ForceUnauthenticated.
—Portenabled – This variable is set to TRUE if the port control mechanism is applied.
—PortState – This variable is set to on if the data frame is permitted to pass through the port.
—Certificate Authentication– This variable is set to TRUE if the certificate authentication and key management is chosen.
—PSK Authentication– This variable is set to TRUE if the PSK authentication and key management is chosen.

#### 8.3.2.3 AE state machine procedures

The following list summarizes the procedures used by the AE state machine
— **STADisconnect**() – Execution of this procedure DeLinkVerifies the STA.
— **CalcMSK**() – Generates the MSK.

***End of changes to Clause 8.***

## 10. Layer management

### 10.3 MLME SAP interface

#### 10.3.2 Scan

#### 10.3.2.2 MLME-SCAN.confirm

#### 10.3.2.2.2 Semantics of the service primitive

***Insert the following elements at the end of the untitled table listing the elements of BSS Description in 10.3.2.2.2:***

| Name | Type | Valid range | Description |
|---|---|---|---|
| WAPI | WAPI Information Element | As defined in frame Format | A description of the cipher suites and AKM suites supported in the BSS. |

#### 10.3.4 ~~Authenticate~~ Linkverification

***Change the text in 10.3.4 as follows:***
This mechanism supports the process of establishing a ~~authentication~~ Linkverification relationship with a peer MAC entity.

#### 10.3.4.1 MLME-~~AUTHENTICATE~~ LinkVerify.request



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

### 10.3.4.1.1 Function

***Change the text in 10.3.4.1.1 as follows:***

This primitive requests ~~authentication~~ Linkverification with a specified peer MAC entity.

### 10.3.4.1.2 Semantics of the service primitive

***Change the text in 10.3.4.1.2 as follows:***

The primitive parameters are as follows:

MLME- ~~AUTHENTICATE~~ LinkVerify .request (
PeerSTAAddress,
~~AuthenticationType,~~
~~AuthenticateFailureTimeout~~
LinkVerifyFailureTimeout
)

| Name | Type | Valid range | Description |
|---|---|---|---|
| PeerSTAAddress | MACAddress | Any valid Individual MAC address | Specifies the address of the peer MAC entity with which to perform the Linkverification process |
| ~~AuthenticationType~~ | ~~AuthenticationType~~ | ~~OPEN_SYSTEM, SHARED_KEY~~ | ~~Specifies the type of authentication algorithm to use during the authentication process.~~ |
| ~~Authentication~~LinkVerifyFailure-Timeout | Integer | ≥ 1 | Specifies a time limit (in TU) after which the Linkverification procedure will be terminated |

### 10.3.4.1.3 When generated

***Change the text in 10.3.4.1.3 as follows:***

This primitive is generated by the SME for a STA to establish ~~authentication~~ Linkverification with a specified peer MAC entity in order to permit Class 2 frames to be exchanged between the two STAs. During the ~~authentication~~ Linkverification procedure, the SME may generate additional MLME~~—AUTHENTICATE~~



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

LinkVerify.request primitives.

#### 10.3.4.1.4 Effect of request

***Change the text in 10.3.4.1.4 as follows:***

This primitive initiates a~~n authentication~~ Linkverification procedure. The MLME subsequently issues an MLME~~- AUTHENTICATE~~ Linkverify.confirm that reflects the results.

### 10.3.4.2 MLME~~- AUTHENTICATE~~ Linkverify.confirm

#### 10.3.4.2.1 Function

***Change the text in 10.3.4.2.1 as follows:***

This primitive reports the results of a~~n authentication~~ Linkverification attempt with a specified peer MAC entity.

#### 10.3.4.2.2 Semantics of the service primitive

***Change the text in 10.3.4.2.2 as follows:***

The primitive parameters are as follows:

MLME- ~~AUTHENTICATE~~ LinkVerify.confirm (
PeerSTAAddress,
~~AuthenticationType~~,
ResultCode
)

| Name | Type | Valid range | Description |
|---|---|---|---|
| PeerSTAAddress | MACAddress | Any valid individual MAC address | Specifies the address of the peer MAC entity with which the Linkverification process was attempted. This value must match the peerSTAAddress parameter specified in the corresponding MLME-~~-AUTHENTICATE~~ LinkVerify.request. |
| ~~AuthenticationType~~ | ~~Enumeration~~ | ~~OPEN_SYSTEM, SHARED_KEY~~ | ~~Specifies the type of authentication algorithm that was used during the authentication process. This value must match the authenticationType parameter specified in the corresponding~~ |



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

| Name | Type | Valid range | Description |
|---|---|---|---|
| | | | ~~MLME-AUTHENTICATE.request.~~ |
| ResultCode | Enumeration | SUCCESS<br>INVALID_PARAMETERS<br>TIMEOUT<br>TOO_MANY_<br>SIMULTANEOUS_<br>REQUESTS<br>REFUSED | Indicates the result of the MLME-~~AUTHENTICATE~~ LinkVerify.request. |

#### 10.3.4.2.3 When generated

***Change the text in 10.3.4.2.3 as follows:***

This primitive is generated by the MLME as a result of an MLME-~~-AUTHENTICATE~~ Linkverify.request to perform the Linkverification with a specified peer MAC entity.

#### 10.3.4.2.4 Effect of receipt

***Change the text in 10.3.4.2.4 as follows:***

The SME is notified of the results of the~~ authentication~~ Linkverification procedure.

### 10.3.4.3 MLME-~~AUTHENTICATE~~ LINKVERIFY.indication

#### 10.3.4.3.1 Function

***Change the text in 10.3.4.3.1 as follows:***

This primitive reports the establishment of a~~n authentication~~ Linkverification relationship with a specified peer MAC entity.

#### 10.3.4.3.2 Semantics of the service primitive

***Change the text in 10.3.4.3.2 as follows:***

The primitive parameters are as follows:

MLME-~~-AUTHENTICATE~~ LINKVERIFY.indication (
PeerSTAAddress
~~AuthenticationType~~
)



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

| Name | Type | Valid range | Description |
|---|---|---|---|
| PeerSTAAddress | MACAddress | Any valid individual MAC address | Specifies the address of the peer MAC address with which the ~~authentication~~ Linkverification relationship was established |
| ~~AuthenticationType~~ | ~~Enumeration~~ | ~~OPEN_SYSTEM, SHARED_KEY~~ | ~~Specifies the type of authentication algorithm that was used during the authentication process.~~ |

#### 10.3.4.3.3 When generated

***Change the text in 10.3.4.3.3as follows:***

This primitive is generated by MLME as a result of the establishment of a~~n authentication~~ Linkverification relationship with a specified peer MAC entity that resulted from a~~n authentication~~ Linkverification procedure that is initiated by that specified peer MAC entity.

#### 10.3.4.3.4 Effect of receipt

***Change the text in 10.3.4.3.4 as follows:***

The SME is notified of the establishment of the ~~authentication~~ Linkverification relationship.

### 10.3.5 ~~Deauthenticate~~ DELINKVERIFY

***Change the text in 10.3.5 as follows:***

This mechanism supports the process of invalidating a ~~authentication~~ Linkverification relationship with a peer MAC entity.

#### 10.3.5.1 MLME- ~~DEAUTHENTICATE~~ DELINKVERIFY.request

#### 10.3.5.1.1 Function

***Change the text in 10.3.5.1.1 as follows:***

This primitive requests that the ~~authentication~~ Linkverification relationship with a specified peer MAC entity be invalidated.

#### 10.3.5.1.2 Semantics of the service primitive

***Change the text in 10.3.5.1.2 as follows:***

The primitive parameters are as follows:



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

MLME- ~~**DEAUTHENTICATE**~~ **DELINKVERIFY**.request (
PeerSTAAddress,
~~ReasonCode~~
)

| Name | Type | Valid range | Description |
|---|---|---|---|
| PeerSTAAddress | MACAddress | Any valid individual MAC address | Specifies the address of the peer MAC entity with which to invalidate the relationship of the ~~authentication~~ Linkverification |
| ~~ReasonCode~~ | ~~As defined in frame format~~ | ~~As defined in frame format~~ | ~~Specifies the reason for initiating the invalidation procedure of the authentication~~ |

### 10.3.5.1.3 When generated

***Change the text in 10.3.5.1.3 as follows:***

This primitive is generated by the SME for a STA to invalidate the ~~authentication~~ Linkverification with a specified peer MAC entity in order to prevent the exchange of Class 2 frames between the two STAs. During the invalidation procedure, the SME may generate additional MLME-~~DEAUTHENTICATE~~ DELINKVERIFY.request primitives.

### 10.3.5.1.4 Effect of receipt

***Change the text in 10.3.5.1.4 as follows:***

This primitive initiates the invalidation procedure of the ~~authentication~~ Linkverification. The MLME subsequently issues an MLME-~~DEAUTHENTICATE~~ DELINKVERIFY .confirm that reflects the results.

### 10.3.5.2 MLME-~~DEAUTHENTICATE~~ DELINKVERIFY.confirm

### 10.3.5.2.1 Function

***Change the text in 10.3.5.2.1 as follows:***

This primitive reports the result of a ~~Deauthentication Delinkverification~~ attempt with a specified peer MAC entity.

### 10.3.5.2.2 Semantics of the service primitive

***Change the text in 10.3.5.2.2as follows:***



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The primitive parameters are as follows:

MLME-~~ DEAUTHENTICATE~~ DELINKVERIFY.confirm (
PeerSTAAddress,
ResultCode
)

| Name | Type | Valid range | Description |
|---|---|---|---|
| PeerSTAAddress | MACAddress | Any valid individual MAC address | Specifies the address of the peer MAC entity with which the invalidation process of the ~~authentication~~ linkverification was attempted |
| ResultCode | Enumeration | SUCCESS<br>INVALID_PARAMETERS<br>TOO_MENY_<br>SIMULTANEOUS_<br>REQUESTS | Indicates the result of the MLME-~~DEAUTHENTICATE~~ DELINKVERIFY.request |

### 10.3.5.2.3 When generated

***Change the text in 10.3.5.2.3 as follows:***

This primitive is generated by the MLME as the result of an MLME-~~ DEAUTHENTICATE~~ DELINKVERIFY.request to invalidate the a~~uthentication~~ linkverification relationship with a specified peer MAC entity.

### 10.3.5.2.4 Effect of receipt

***Change the text in 10.3.5.2.4 as follows:***

The SME is notified of the results of the ~~authentication~~ linkverification procedure.

### 10.3.5.3 MLME-~~ DEAUTHENTICATE~~ DELINKVERIFY.indication

### 10.3.5.3.1 Function

***Change the text in 10.3.5.3.1 as follows:***

This primitive reports the invalidation of a ~~authentication~~ Linkverification relationship with a specified peer MAC entity.

### 10.3.5.3.2 Semantics of the service primitive

***Change the text in 10.3.5.3.2 as follows:***



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The primitive parameters are as follows:

MLME- ~~DEAUTHENTICATE~~ DELINKVERIFY.indication (
PeerSTAAddress,
ReasonCode
)

| Name | Type | Valid range | Description |
|---|---|---|---|
| PeerSTAAddress | MACAddress | Any valid individual MAC address | Specifies the address of the peer MAC address with which the ~~authentication~~ Linkverification relationship is invalidated |
| ReasonCode | As defined in frame format | As defined in frame format | Specifies the reason the ~~deauthentication~~ delinkverification procedure is initiated |

### 10.3.5.3.3 When generated

***Change the text in 10.3.5.3.3 as follows:***

This primitive is generated by the MLME as a result of the invalidation of the ~~authentication~~ Linkverification relationship with specified peer MAC entity.

### 10.3.5.3.4 Effect of receipt

***Change the text in 10.3.5.3.4 as follows:***

The SME is notified of the invalidation of the specific ~~authentication~~ Linkverification relationship.

## 10.3.6 Associate

### 10.3.6.1 MLME-ASSOCIATE.request

### 10.3.6.1.2 Semantics of the service primitive

***Change the following primitive parameter list in 10.3.6.1.2:***

MLME-ASSOCIATE.request(
PeerSTAAddress,
AssociateFailureTimeout,
CapabilityInformation,
ListenInterval,
Supported Channels



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

WAPI
)

***Insert the following row at the end of the untitled table defining the primitive parameters in 10.3.6.1.2:***

| Name | Type | Valid range | Description |
|---|---|---|---|
| WAPI | WAPI Information element | As defined in frame format | Specifies the cipher suites and the AKM suites that BSS supports |

### 10.3.6.3 MLME-ASSOCIATE.indication

### 10.3.6.3.2 Semantics of the service primitive

***Change the following primitive parameter list in 10.3.6.3.2:***

MLME-ASSOCIATE.indication(
PeerSTAAddress,
WAPI
)

***Insert the following row at the end of the untitled table defining the primitive parameters in 10.3.6.3.2:***

| Name | Type | Valid range | Description |
|---|---|---|---|
| WAPI | WAPI Information Element | As defined in frame format | Specifies the cipher suites and the AKM suites that BSS supports. There is only one unicast cipher suite in the WAPI information element. |

### 10.3.7 Reassociate

### 10.3.7.1 MLME-REASSOCIATE.request

### 10.3.7.1.2 Semantics of the service primitive

***Change the following primitive parameter list in 10.3.7.1.2:***

MLME-REASSOCIATE.request(
NewAPAddress,
ReassociateFailureTimeout,
CapabilityInformation,



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ListenInterval,
Supported Channels,
WAPI
)

***Insert the following row at the end of the untitled table defining the primitive parameters in 10.3.7.1.2:***

| Name | Type | Valid range | Description |
|---|---|---|---|
| WAPI | WAPI Information Element | As defined in frame format | Specifies the cipher suites and the AKM suites that BSS supports |

### 10.3.7.3 MLME-REASSOCIATE.indication

#### 10.3.7.3.2 Semantics of the service primitive

***Change the following primitive parameter list in 10.3.7.3.2:***

MLME-REASSOCIATE.indication(
PeerSTAAddress,
WAPI
)

***Insert the following row at the end of the untitled table defining the primitive parameters in 10.3.7.3.2:***

| Name | Type | Valid range | Description |
|---|---|---|---|
| WAPI | WAPI Information Element | As defined in frame format | Specifies the cipher suites and the AKM suites that BSS supports. There is only one unicast cipher suite in the WAPI information element. |

***After 10.3.10.2.4, insert 10.3.17 through 10.3.21.1.4:***

### 10.3.17 MLME-SETWPIKEYS

This primitive sets the keys used by the STA and the peer STA authenticated when WPI is enabled.

#### 10.3.17.1 MLME-SETWPIKEYS.request

#### 10.3.17.1.1 Function



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

This primitive requests a STA's MAC entity set the keys used by the STA and another peer STA's MAC entity when WPI is enabled.

### 10.3.17.1.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME-SETWPIKEYS.request (
Keylist
)

| Name | Type | Description |
|---|---|---|
| Keylist | A set of SetKeyDescriptors | The list of keys to be used by the MAC. |

Each SetKeyDescriptor consists of the following elements:

| Name | Type | Description |
|---|---|---|
| Key | Octet string | Data encryption key and data integrity key |
| Length | Integer | The length of the key in octet |
| KeyIdx | Integer | The number of the key, with a value of 0 or 1 |
| KeyType | Integer | Multicast, Unicast,STAKey |
| PeerSTAAddress | MACAddress | Any valid individual MAC address. This parameter is valid only when the Key Type value is Unicast, when the Key Type value is Multicast and the STA is in IBSS, or when the Key Type value is STAKey. |
| AE/ASUE or Initiator/Peer | Boolean | Whether the key is configured by the AE(Initiator) or ASUE(Peer); true indicates AE or Initiator; false indicates ASUE or Peer. |
| GSN | Integer | The sequence number of the currently encrypted multicast package. It is valid only when KeyType is multicast key. |
| Cipher Suite Selector | 4 octets | As defined in the WAPI information element format The cipher suite required for this association. |

### 10.3.17.1.3 When generated

This primitive is generated by the SME at any time when the key negotiation is completed or the shared key is set.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

#### 10.3.17.1.4 Effect of receipt

This primitive enables MAC to perform the privacy of the data using the key. The MLME subsequently issues an MLME-SETWPIKEYS.confirm that reflects the results.

### 10.3.17.2 MLME-SETWPIKEYS.confirm

#### 10.3.17.2.1 Function

This primitive confirms that the action of the associated MLME-SETWPIKEYS.request primitive has been completed.

#### 10.3.17.2.2 Semantics of the service primitive

This primitive has no parameters.

#### 10.3.17.2.3 When generated

This primitive is generated by the MAC in response to receipt of an MLME-SETWPIKEYS.request primitive.This primitive is issued when the action requested has been completed.

#### 10.3.17.2.4 Effect of receipt

The SME is notified that the requested action of the MLME-SETWPIKEYS.request primitive is completed.

### 10.3.18 MLME-DELETEWPIKEYS

The following primitives delete the keys that a STA used in WPI.

### 10.3.18.1 MLME-DELETEWPIKEYS.request

#### 10.3.18.1.1 Function

This primitive requests a STA's MAC entity delete the keys that the STA used in WPI with the peer STA's MAC entity when WPI is enabled.

#### 10.3.18.1.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME-DELETEWPIKEYS.request (
Keylist
)



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

| Name | Type | Description |
|---|---|---|
| Keylist | A set of DeleteKeyDescriptors | The list of keys to be used by the MAC. |

Each DeleteKeyDescriptor consists of the following elements:

| Name | Type | Description |
|---|---|---|
| PeerSTAAddress | MACAddress | Any valid individual MAC address. This parameter is valid only when the Key Type value is Unicast, when the Key Type value is Multicast and the STA is in IBSS, or when the Key Type value is STAKey. |
| KeyIdx | Integer | The number of the key, with a value of 0 or 1 |
| KeyType | Integer | Multicast, Unicast, STAKey |

#### 10.3.18.1.3 When generated

This primitive is generated by the SME at any time when a STA wishes to delete the key.

#### 10.3.18.1.4 Effect of receipt

This primitive disables MAC to perform the privacy of the data using the key. The MLME subsequently issues an MLME- DELETEWPIKEYS.confirm that reflects the results.

### 10.3.18.2 MLME-DELETEWPIKEYS.confirm

#### 10.3.18.2.1 Function

This primitive confirms that the action of the associated MLME-DELETEWPIKEYS.request primitive has been completed.

#### 10.3.18.2.2 Semantics of the service primitive

This primitive has no parameters.

#### 10.3.18.2.3 When generated

This primitive is generated by the MAC in response to receipt of an MLME-DELETEWPIKEYS.request primitive. This primitive is issued when the action requested has been completed.



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

#### 10.3.18.2.4 Effect of receipt

The SME is notified that the requested action of the MLME-DELETEWPIKEYS.request primitive is completed.

### 10.3.19 MLME-STAKEYESTABLISHED

#### 10.3.19.1 MLME-STAKEYESTABLISHED.indication

#### 10.3.19.1.1 Function

This primitive notifies the SME that a STAKey is required.

#### 10.3.19.1.2 Semantics of the service primitive

This primitive has two parameters, the MAC addresses of the two STAs.
The primitive parameters are as follows:

MLME-STAKEYESTABLISHED.indication (
Address1,
Address2
)

| Name | Type | Valid range | Description |
|---|---|---|---|
| Address1 | MACAddress | Any valid individual MAC address | MAC address of initiating STA. |
| Address2 | MACAddress | Any valid individual MAC address | MAC address of peer STA. |

#### 10.3.19.1.3 When generated

This primitive is generated by the MAC when a STAKey is required.

#### 10.3.19.1.4 Effect of receipt

The SME is notified that a STAKey is required, is made aware of the two STA MAC addresses involved, and can then send STAKey Handshake messages to each STA.

### 10.3.20 SetProtection

#### 10.3.20.1 MLME-SETPROTECTION.request

#### 10.3.20.1.1 Function



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

This primitive indicates whether protection is required for frames sent to and received from the indicated MAC address.

### 10.3.20.1.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME-SETPROTECTION.request(
Protectlist
)

| Name | Type | Description |
|---|---|---|
| Protectlist | A set of protection elements | The list of how each key is being used currently. |

Each Protectlist consists of the following elements:

| Name | Type | Description |
|---|---|---|
| Address | MACAddress | Any valid individual MAC address. This parameter is valid only when the Key Type value is Unicast or STAKey or when the Key Type value is Multicast and is from an IBSS STA. |
| ProtectType | Enumeration (None, Rx, Tx, Rx_Tx) | The protection value for this MAC. |
| KeyType | Integer | Defines whether this key is a multicast key, unicast key, or STAKey. |

### 10.3.20.1.3 When generated

This primitive is generated by the SME when protection is required for frames sent to and received from the indicated MAC address.

### 10.3.20.1.4 Effect of receipt

Receipt of this primitive causes the MAC to set the protection and to protect data frames as indicated in the ProtectType element of the Protectlist parameter:

— None: Specifies that data frames neither from the MAC address nor to the MAC address shall be protected.
— Rx: Specifies that data frames from MAC address shall be protected.
— Tx: Specifies that data frames to MAC address shall be protected.
— Rx_Tx: Specifies that data frames to and from MAC address shall be protected.



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Once it is specified that a data frame is protected to or from a MAC address, this shall be reset by the MLME-SETPROTECTION.request primitive. The MLME-SETPROTECTION.request primitive deletes the state by specifying None.

#### 10.3.20.2 MLME-SETPROTECTION.confirm

##### 10.3.20.2.1 Function

This primitive indicates that the frame protection request is completed.

##### 10.3.20.2.2 Semantics of the service primitive

There are no parameters for this primitive.

##### 10.3.20.2.3 When generated

This primitive is generated by the MAC when the protection request is completed.

##### 10.3.20.2.4 Effect of receipt

The SME is notified that the protection request is completed.

#### 10.3.21 MLME-PROTECTEDFRAMEDROPPED

#### 10.3.21.1 MLME- PROTECTEDFRAMEDROPPED.indication

##### 10.3.21.1.1 Function

This primitive notifies the SME that a frame has been dropped because a temporal key was unavailable.

##### 10.3.21.1.2 Semantics of the service primitive

This primitive has two parameters, the MAC addresses of the two STAs.
The primitive parameters are as follows:

MLME- PROTECTEDFRAMEDROPPED.indication (
Address1,
Address2
)

| Name | Type | Valid range | Description |
|---|---|---|---|
| Address1 | MACAddress | Any valid individual MAC address | MAC address of SA. |
| Address2 | MACAddress | Any valid individual MAC address | MAC address of RA. |

##### 10.3.21.1.3 When generated



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

This primitive is generated by the MAC when a frame is dropped because no temporal key is available for the frame.

#### 10.3.21.1.4 Effect of receipt

The SME is notified that a frame was dropped. The SME can use this information in an IBSS to initiate a security association to the peer STA.

***End of changes to Clause 10.***

## 11. MLME

***Replace 11.3 and 11.4 in their entirety with the following text:***

### 11.3 Association and reassociation

This subclause describes the procedures used for ISO/IEC 8802.11 linkverification and delinkverification. The states used in this description are those defined in 5.5.

#### 11.3.1 Linkverification—originating STA

Upon receipt of an MLME-LINKVERIFY.request primitive, the originating STA shall linkverify with the indicated STA using the following procedure:

a) In an ESS, or optionally in an IBSS, the STA shall execute the linkverification mechanism.
b) If the linkverification is successful, the state variable of the indicated STA shall be set to State 2.
c) The STA shall issue an MLME-LINKVERIFY.confirm primitive to inform the SME of the result of the linkverification.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using MLME-DELETEWPIKEYS.request primitive before invoking MLME-LINKVERIFY.request primitive.

#### 11.3.2 Linkverification—destination STA

Upon receipt of a Linkverification frame with linkverification transaction sequence number equal to 1, the destination STA shall linkverify with the indicated STA using the following procedure:

a) The STA shall execute the linkverification mechanism.
b) The STA shall issue an MLME-LINKVERIFY.indication primitive to inform the SME of the linkverification.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using the MLME-DELETEWPIKEYS.request primitive upon receiving an MLME-LINKVERIFY.indication primitive.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If the STA is in an IBSS, if the SME decides to initiate a WAPI association, and if the SME does not know the security policy of the peer, it may issue a unicast Probe Request frame to the peer by invoking an MLME-SCAN.request to discover the peer's security policy.

### 11.3.3 Delinkverification—originating STA

Upon receipt of an MLME-DELINKVERIFY.request primitive, the originating STA shall delinkverify with the indicated STA using the following procedure:

a) If the state variable of the indicated STA is in State 2 or State 3, the STA shall send a Delinkverification frame to the indicated STA.
b) The state variable of the indicated STA shall be set to State 1.
c) The STA shall issue an MLME-DELINKVERIFY.confirm primitive to inform the SME of the completion of the delinkverification.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using an MLME-DELETEWPIKEYS.request primitive and by invoking MLME-SETPROTECTION. request(None) before invoking an MLME-DELINKVERIFY.request primitive.

### 11.3.4 Delinkverification—destination STA

Upon receipt of a Delinkverification frame, the destination STA shall delinkverify with the indicated STA using the following procedure:

a) The state variable of the indicated STA shall be set to State 1.
b) The STA shall issue an MLME-DELINKVERIFY.indication primitive to inform the SME of the delinkverification.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using the MLME-DELETEWPIKEYS.request primitive and by invoking MLME-SETPROTECTION.request (None) upon receiving an MLME-DELINKVERIFY.indication primitive.

## 11.4 Association, reassociation, and disassociation

This subclause defines how a STA associates and reassociates with an AP and how it disassociates from it. The states used in this description are those defined in 5.5.

### 11.4.1 STA association procedures

Upon receipt of an MLME-ASSOCIATE.request primitive, a STA shall associate with an AP via the following procedure:



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

a) The STA shall transmit an Association Request frame to the AP with which that STA is linkverified. If the MLME-ASSOCIATE.request primitive contained a WAPI information element with only one unicast cipher suite and only one authenticated key suite, this WAPI information element shall be included in the Association Request frame.
b) If an Association Response frame is received with a status value of "successful," the STA is now associated with the AP. The state variable shall be set to State 3, and the MLME shall issue an MLME-ASSOCIATE.confirm primitive indicating the successful completion of the operation.
c) If an Association Response frame is received with a status value other than "successful" or the AssociateFailureTimeout expires, the STA is not associated with the AP. The MLME shall issue an MLME-ASSOCIATE.confirm primitive indicating the failure of the operation.
d) The SME shall establish a WAPI association by calling MLME.SETPROTECTION.request primitive with ProtectType set to "Rx_Tx," or it shall do nothing if it does not wish to secure communication.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using MLME-DELETEWPIKEYS.request primitive before invoking MLME-ASSOCIATE.request primitive.

### 11.4.2 AP association procedures

When an Association Request frame is received from a STA, the AP shall associate with the STA using the following procedure:

a) If the STA is not linkverified, the AP shall transmit a Delinkverification frame to the STA and terminate the association procedure.
b) In a WAPI association, the AP shall check if the values received in the WAPI information element match the AP's security policy. If not, the association shall not be accepted.
c) The AP shall transmit an Association Response. If the status value is "successful," the association identifier assigned to the STA shall be included in the response.
d) When the Association Response with a status value of "successful" is acknowledged by the STA, the STA is considered to be associated with this AP. The state variable for the STA shall be set to State 3.
e) The MLME shall issue an MLME-ASSOCIATE.indication primitive to inform the SME of the association.
f) The SME shall establish a WAPI association by calling MLME.SETPROTECTION.request primitive with ProtectType set to "Rx_Tx," or it shall do nothing if it does not wish to secure communication.
g) The SME will inform the DS of the new association.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using MLME-DELETEWPIKEYS.request primitive upon receiving an MLME-ASSOCIATE.indication primitive.



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

### 11.4.3 STA reassociation procedures

Upon receipt of an MLME-REASSOCIATE.request primitive, a STA shall reassociate with an AP via the following procedure:

a) If the state variable is in State 1, the STA shall inform the SME of the failure of the reassociation by issuing an MLME-REASSOCIATE.confirm primitive.
b) The STA shall transmit a Reassociation Request frame to the new AP. If the MLME-REASSOCIATE.request primitive contained a WAPI information element with only one unicast cipher suite and only one authenticated key suite, this WAPI information element shall be included in the Reassociation Request frame.
c) If a Reassociation Response frame is received with a status value of "successful," the STA is now associated with the new AP. The state variable shall be set to State 3, and the MLME shall issue an MLME-REASSOCIATE.confirm primitive indicating the successful completion of the operation.
d) If a Reassociation Response frame is received with a status value other than "successful" or the AssociateFailureTimeout expires, the STA is not associated with the AP. The MLME shall issue an MLME-REASSOCIATE.confirm primitive indicating the failure of the operation.
e) The SME shall establish a WAPI association calling MLME.SETPROTECTION.request primitive with ProtectType set to "Rx_Tx," or it shall do nothing if it does not wish to secure communication.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using MLME-DELETEWPIKEYS.request primitive before invoking MLME-REASSOCIATE.request primitive.

### 11.4.4 AP reassociation procedures

Whenever a Reassociation Request frame is received from a STA, the AP uses the following procedure to support reassociation:

a) If the STA is not linkverified, the AP shall transmit a Delinkverification frame to the STA and terminate the reassociation procedure.
b) In a WAPI association, the AP shall check if the values received in the WAPI information element match the AP's security policy. If not, the association shall not be accepted.
c) The AP shall transmit a Reassociation Response frame. If the status value is "successful," the association identifier assigned to the STA shall be included in the response.
d) When the Reassociation Response frame with a status value of "successful" is acknowledged by the STA, the STA is considered to be associated with this AP. The state variable of the STA shall be set to State 3.
e) The MLME shall issue an MLME-REASSOCIATE.indication primitive to inform the SME of the association.
f) The SME shall establish a WAPI association calling MLME.SETPROTECTION.request primitive with ProtectType set to "Rx_Tx," or it shall do nothing if it does not wish to secure communication.
g) The SME will inform the DS of the new association.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

using MLME-DELETEWPIKEYS.request primitive upon receiving an MLME-REASSOCIATE.indication primitive.

### 11.4.5 STA disassociation procedures

Upon receipt of an MLME-DISASSOCIATE.request primitive, an associated STA shall disassociate from an AP using the following procedure:

a) The STA shall transmit a Disassociation frame to the AP with which that STA is associated.
b) The state variable of the AP shall be set to State 2 if and only if it was not State 1.
c) The MLME shall issue an MLME-DISASSOCIATE.confirm primitive indicating the successful completion of the operation.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using the MLME-DELETEWPIKEYS.request primitive and by invoking MLME-SETPROTECTION.request (None) before invoking an MLME-DISASSOCIATE.request primitive.

### 11.4.6 AP disassociation procedures

Upon receipt of a Disassociation frame from an associated STA, the AP shall disassociate the STA via the following procedure:

a) The state variable of the STA shall be set to State 2.
b) The MLME shall issue an MLME-DISASSOCIATE.indication primitive to inform the SME of the disassociation.
c) The SME will update the DS.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using an MLME-DELETEWPIKEYS.request primitive and by invoking MLME-SETPROTECTION. request (None) upon receiving an MLME-DISASSOCIATE.indication primitive.

***End of changes to Clause 11.***



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

# Annex A

(normative)

# Protocol Implementation Conformance Statements (PICS)

## A.4 PICS proforma—ISO/IEC 8802.11,2005 Edition

### A.4.4 MAC protocol

#### A.4.4.1 MAC protocol capabilities

***Change the text of PC1 and PC2 in A.4.4.1 as follows:***

| Item | Protocol capability | References | Status | Support |
|---|---|---|---|---|
| | Are the following MAC protocol capabilities supported? | | | |
| PC1 | ~~Authentication~~Linkverification service | 5.4.3.1, 5.4.3.2, 5.7.6, 5.7.7, ~~8.1~~, Annex C | M | Yes No |
| PC1.1 | ~~Authentication~~Linkverification service | 5.5 | M | Yes No |
| PC1.2 | Open System ~~Authentication~~ Linkverification | 5.7.6 | M | Yes No |
| ~~PC1.3~~ | ~~Shared Key authentication~~ | ~~8.1.2, 8.3~~ | ~~PC2:M~~ | ~~Yes No N/A~~ |
| ~~* PC2~~ | ~~Wired equivalent privacy (WEP) algorithm~~ | ~~5.4.3.3, 8.2, Annex C~~ | ~~O~~ | ~~Yes No~~ |
| ~~PC2.1~~ | ~~WEP Encryption procedure~~ | ~~8.2.3, 8.2.4, 8.2.5~~ | ~~PC2:M~~ | ~~Yes No N/A~~ |
| ~~PC2.2~~ | ~~WEP Decryption procedure~~ | ~~8.2.3, 8.2.4, 8.2.5~~ | ~~PC2:M~~ | ~~Yes No N/A~~ |



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

**Specifications for Enhanced Security**

| Item | Protocol capability | References | Status | Support |
|---|---|---|---|---|
| ~~PC2.3~~<br>PC2 | ~~Security services management~~<br>Authentication and Privacy | ~~8.3~~<br>7.2.2, 7.3.1.4, 5.4.3.3, 8.1, 11.3, 11.4, 8.2 | ~~M~~<br>O | ~~Yes No~~<br>Yes No |
| PC2.1 | WAPI Information Element (IE) | 7.3.2.25 | PC2:M | Yes No |
| PC2.1.1 | Multicast cipher suite | 7.3.2.25 | PC2.1:M | Yes No |
| PC2.1.2 | Unicast cipher suite list | 7.3.2.25 | PC2.1:M | Yes No |
| PC2.1.2.1 | WPI data confidentiality | 8.2 | PC2.1:M | Yes No |
| PC2.1.3 | Authentication key management (AKM) suite list | 7.3.2.25, | PC2.1:M | Yes No |
| PC2.1.3.1 | WAI Certificate Authentication and Key Management | 7.3.2.25 | PC2.1:M | Yes No |
| PC2.1.3.2 | WAI Preshared Key Authentication and Key Management | 7.3.2.25 | PC2.1:M | Yes No |
| PC2.1.3.3 | WAI Authentication and key management | 8.1 | PC2.1:M | Yes No |
| PC2.1.3.3.1 | Key derivation | 8.1.4.10 | PC2.1:M | Yes No |
| PC34.1.3.3.1.1 | Base key derivation | 8.1.4.10.1 | PC2.1:M | Yes No |
| PC2.1.3.3.1.2 | Unicast key derivation | 8.1.4.10.2 | PC2.1:M | Yes No |
| PC2.1.3.3.1.3 | Multicast key derivation | 8.1.4.10.3 | PC2.1:M | Yes No |
| PC2.1.3.3.2 | Unicast Key Negotiation | 8.1.4.3.2 | PC2.1:M | Yes No |
| PC2.1.3.3.3 | Multicast key / STAKey announcement | 8.1.4.4 | PC2.1:M | Yes No |
| PC2.1.4 | WAPI capabilities | 7.3.2.25, | PC2.1:M | Yes No |



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

| Item | Protocol capability | References | Status | Support |
|---|---|---|---|---|
| PC2.1.5 | WAPI preauthentication | 8.1.4.6 | PC2.1:O | Yes No |
| PC2.1.6 | WAPI security association management | 8.1.2 | PC2.1:M | Yes No |
| PC2.1.7 | BKSA caching | 8.1.4.7 | PC2.1:M | Yes No |
| PC2.1.8 | WAPI extended service set (ESS) | 8.1.4.6 | (PC2.1 and CF1):M | Yes No |
| PC2.1.8.1 | WAPI STAKey | 8.1.4.5 | PC2.1.8:O | Yes No |
| PC2.1.9 | WAPI independent basic service set (IBSS) | 8.1.2.1.2 | (PC2.1 and CF2):O | Yes No |
| PC2.1.10 | Rekeying | 8.1.4.8 | PC2.1.8:O | Yes No |
| PC2.1.11 | X.509 v3 certificate | 8.3.1.1 | PC2.1:M | Yes No |
| PC2.1.12 | GBW certificate | 8.3.1.2 | PC2.1:O | Yes No |



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

# Annex C

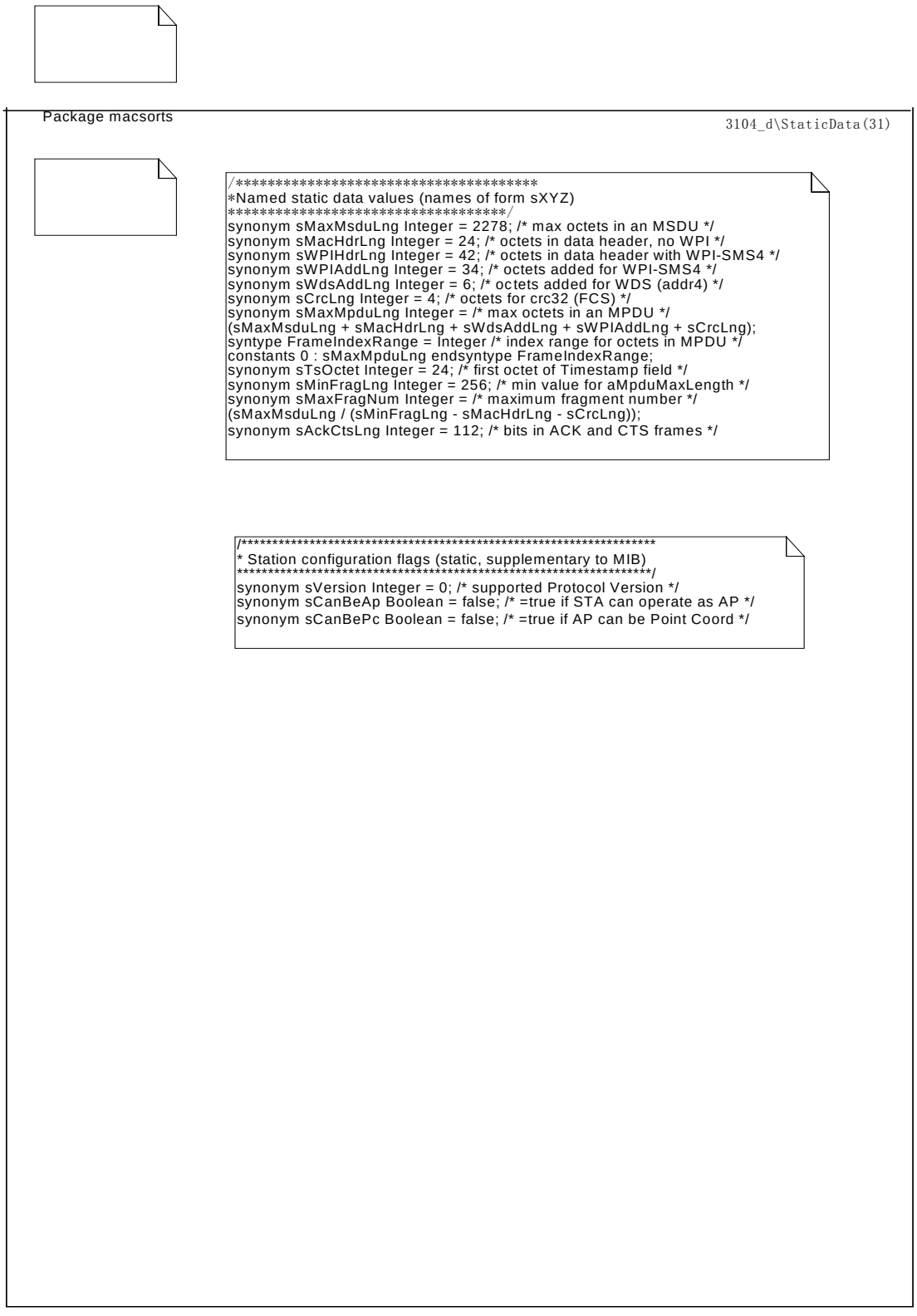
(normative)

# Formal description of MAC operation



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

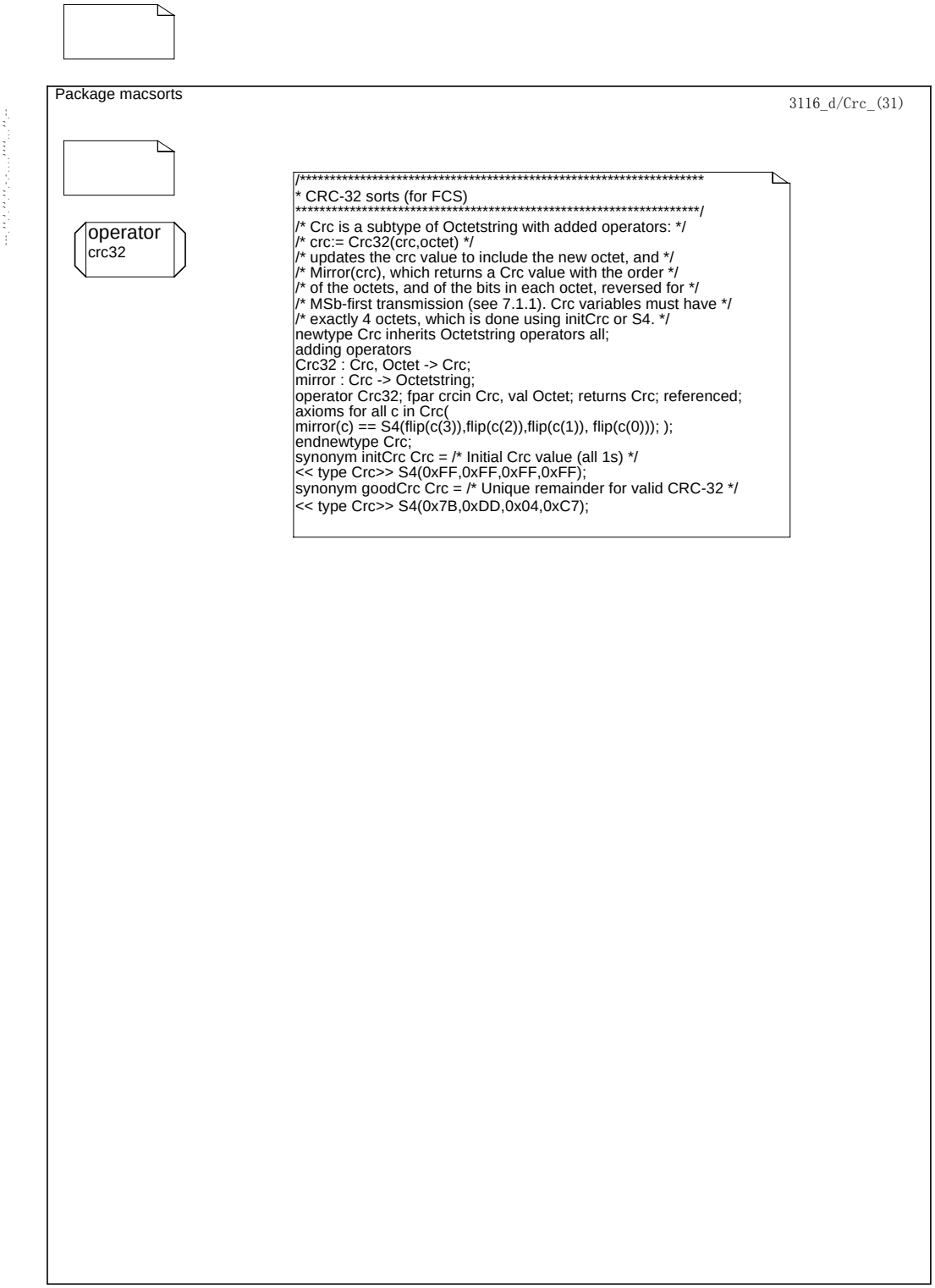
***Replace the diagram on Page 404 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

***Replace the diagram on Page 424 as follow:***

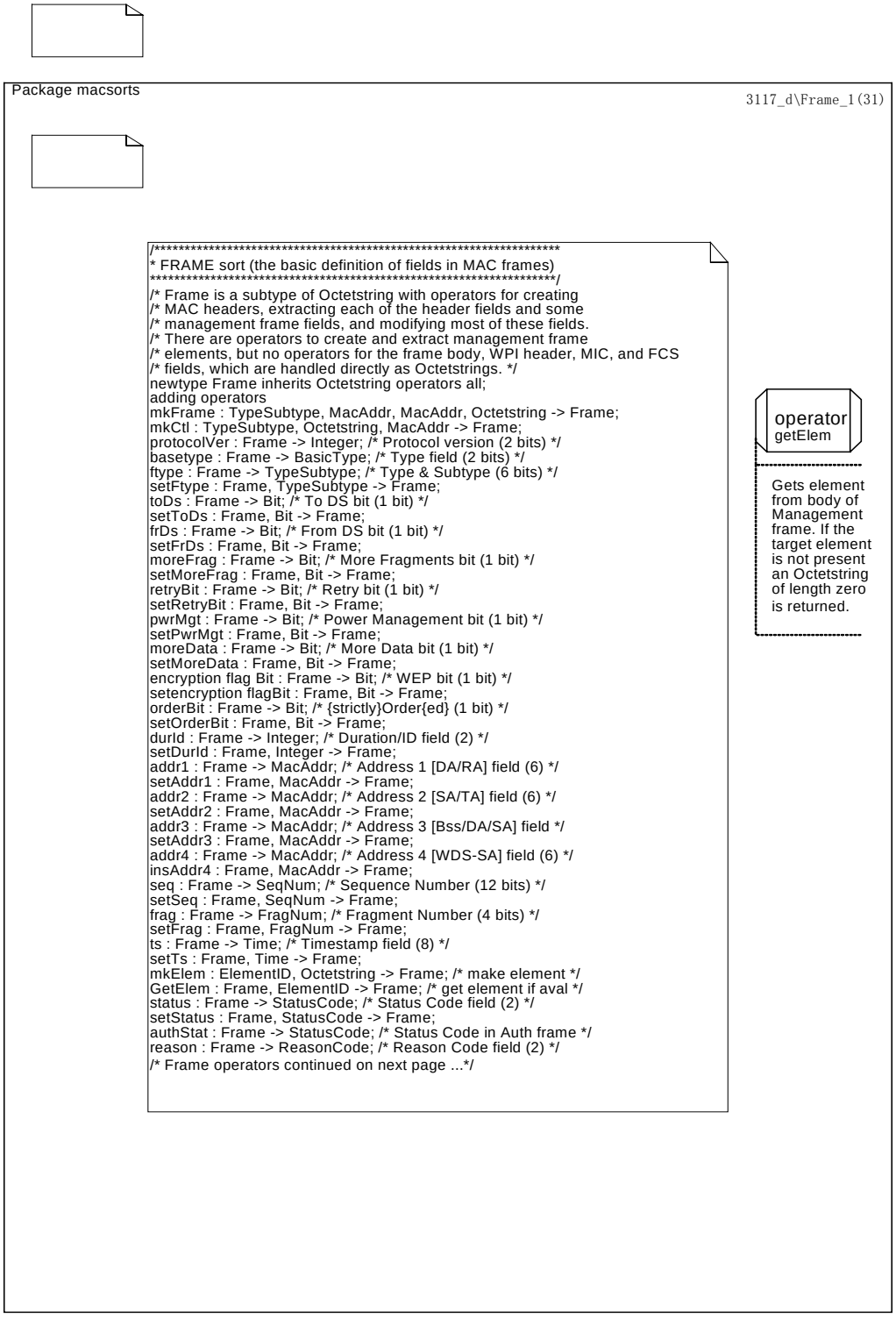





Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

***Delete the diagram on Page 426 .***

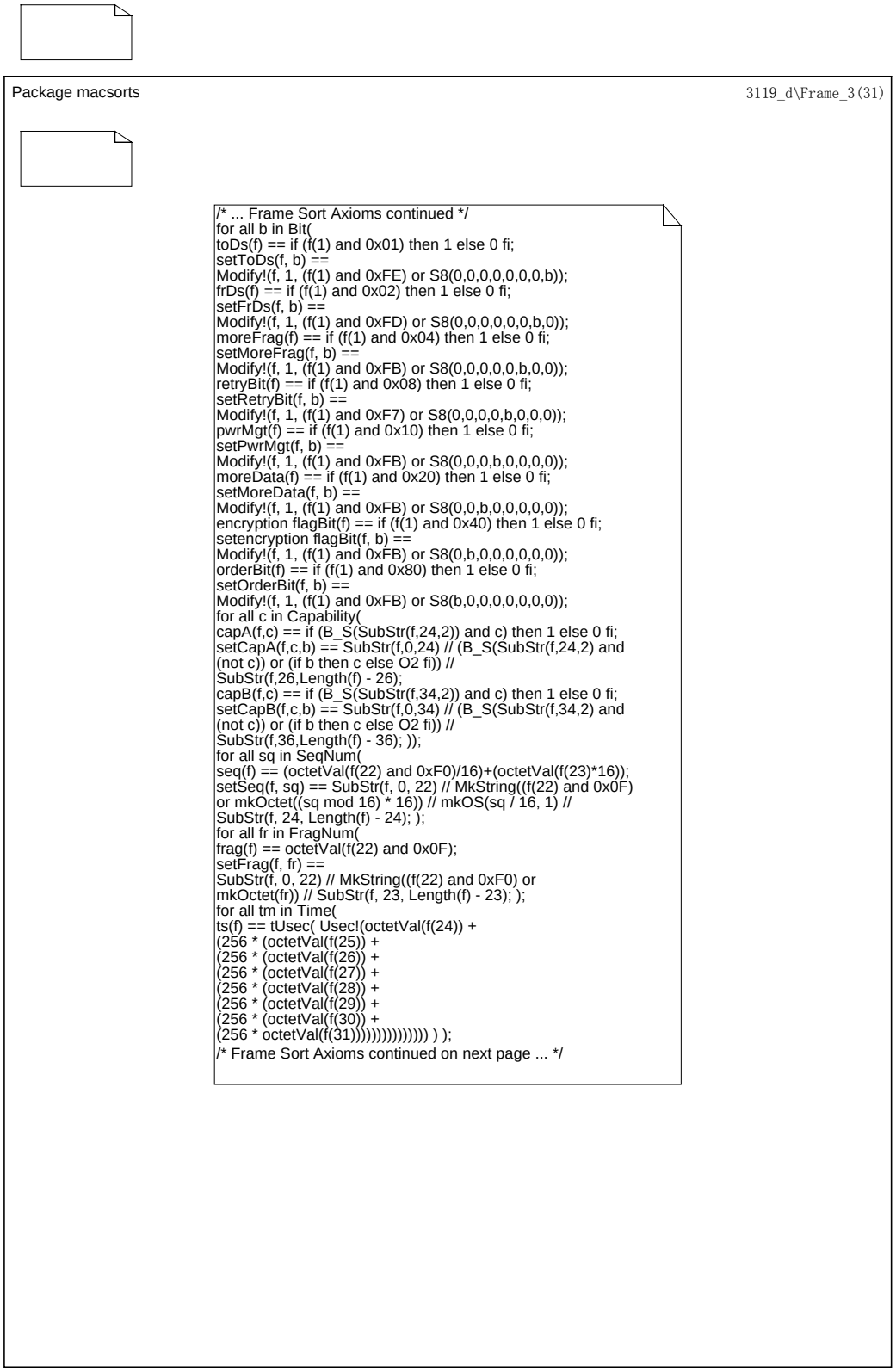
***Replace the diagram on Page 427 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

***Replace the diagram on Page 429 as follow:***

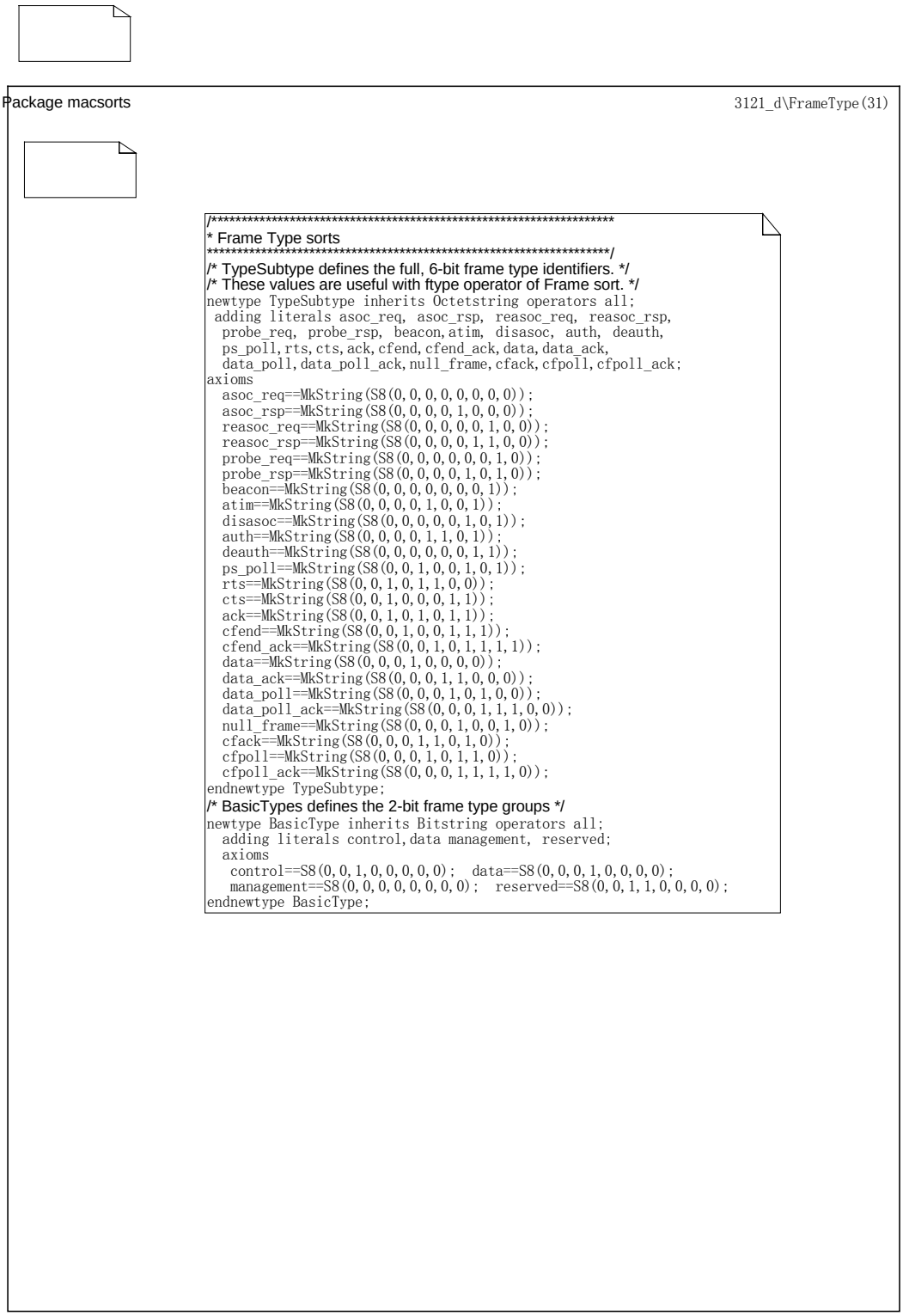





--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

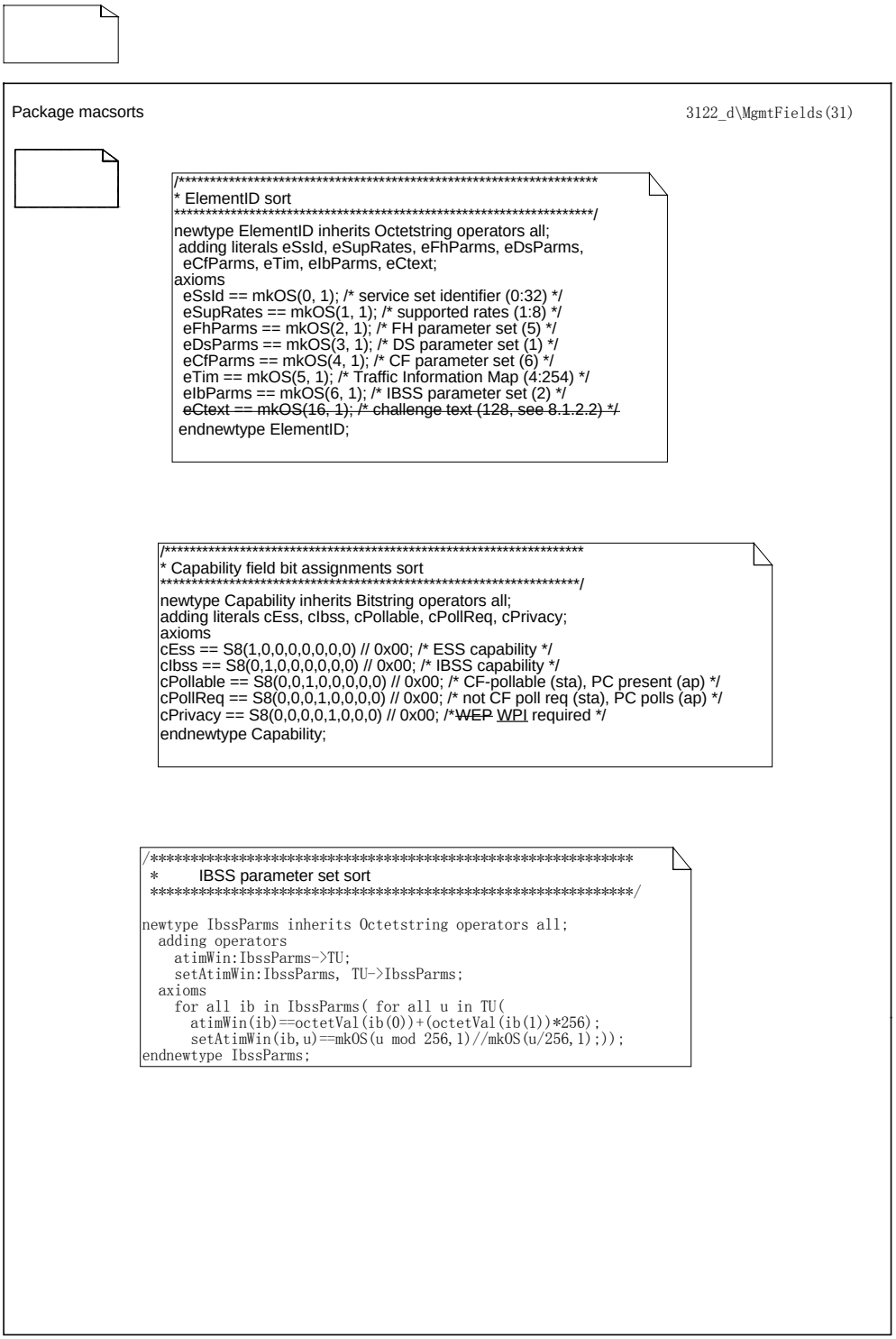
***Replace the diagram on Page 432 as follow:***






--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

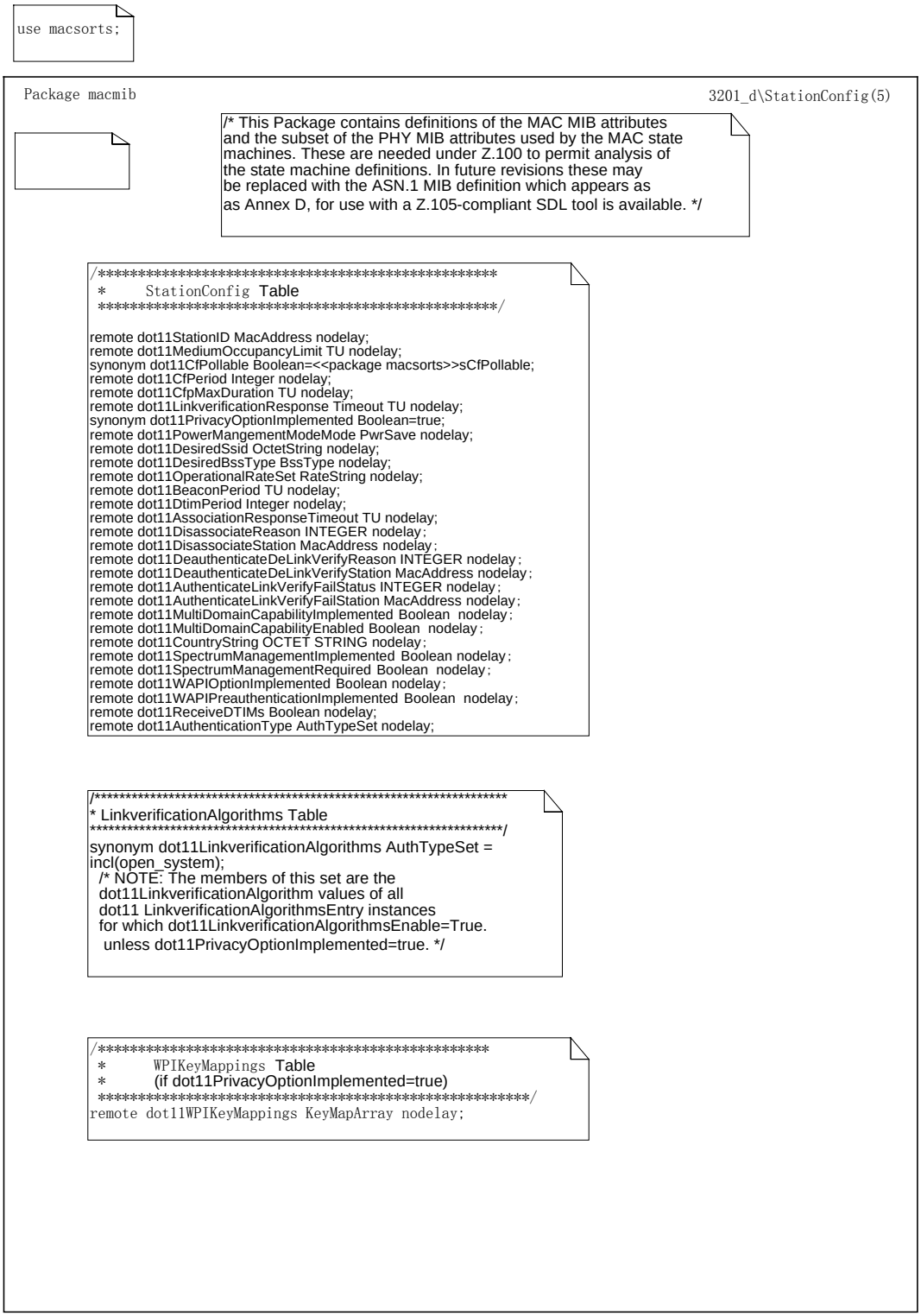
***Change the diagram on Page 433 as follow:***






--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

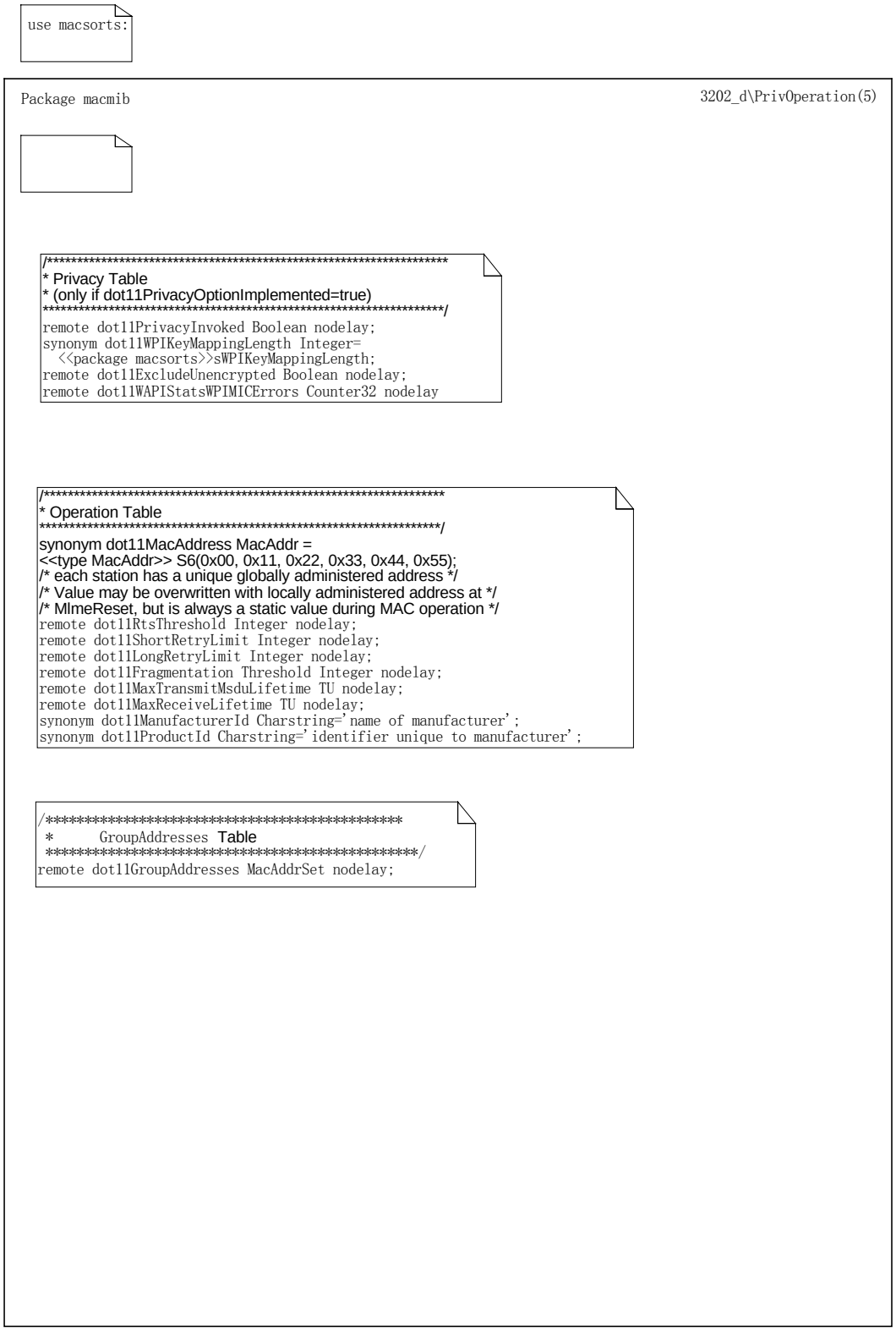
***Replace the diagram on Page 443 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

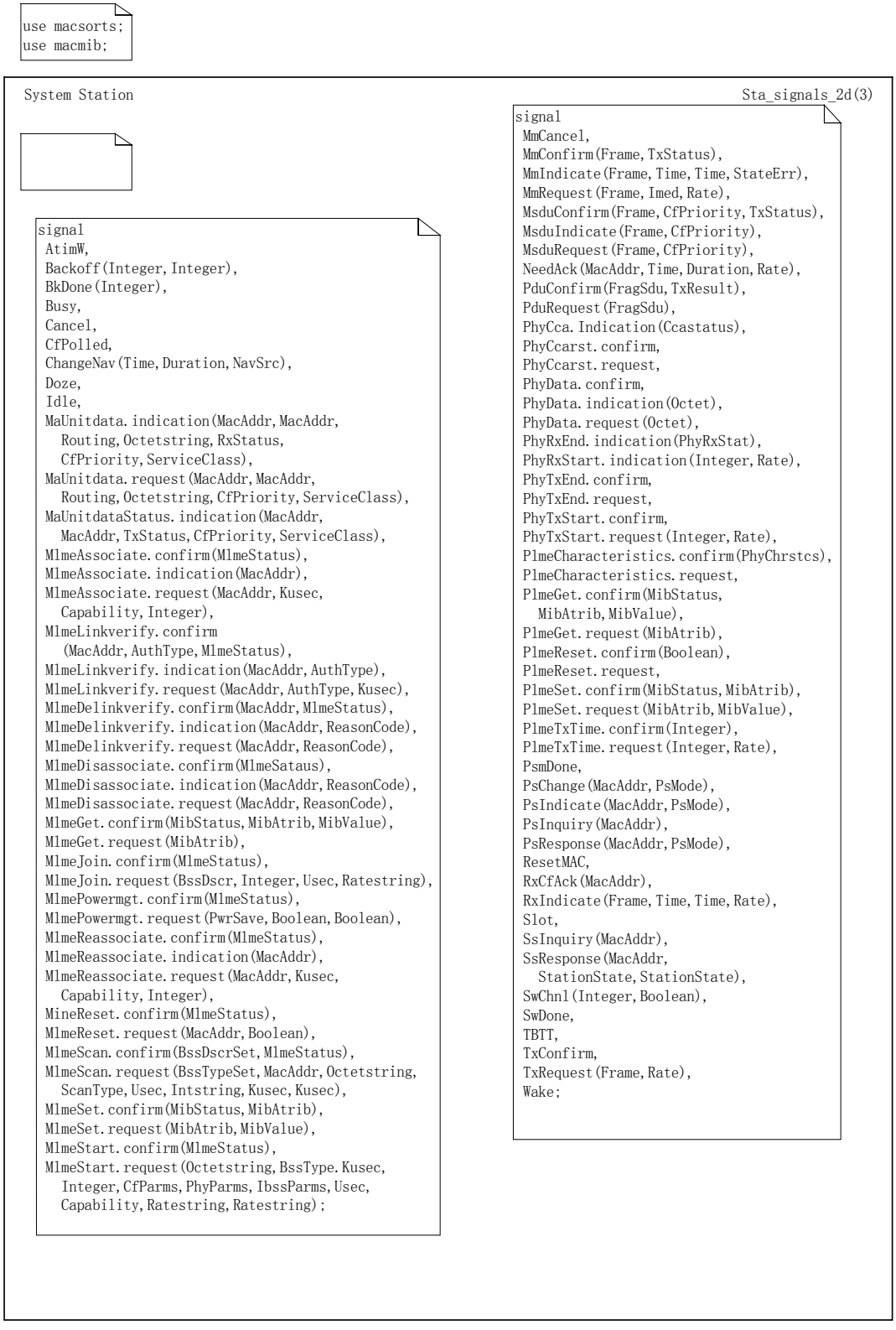
***Replace the diagram on Page 444 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

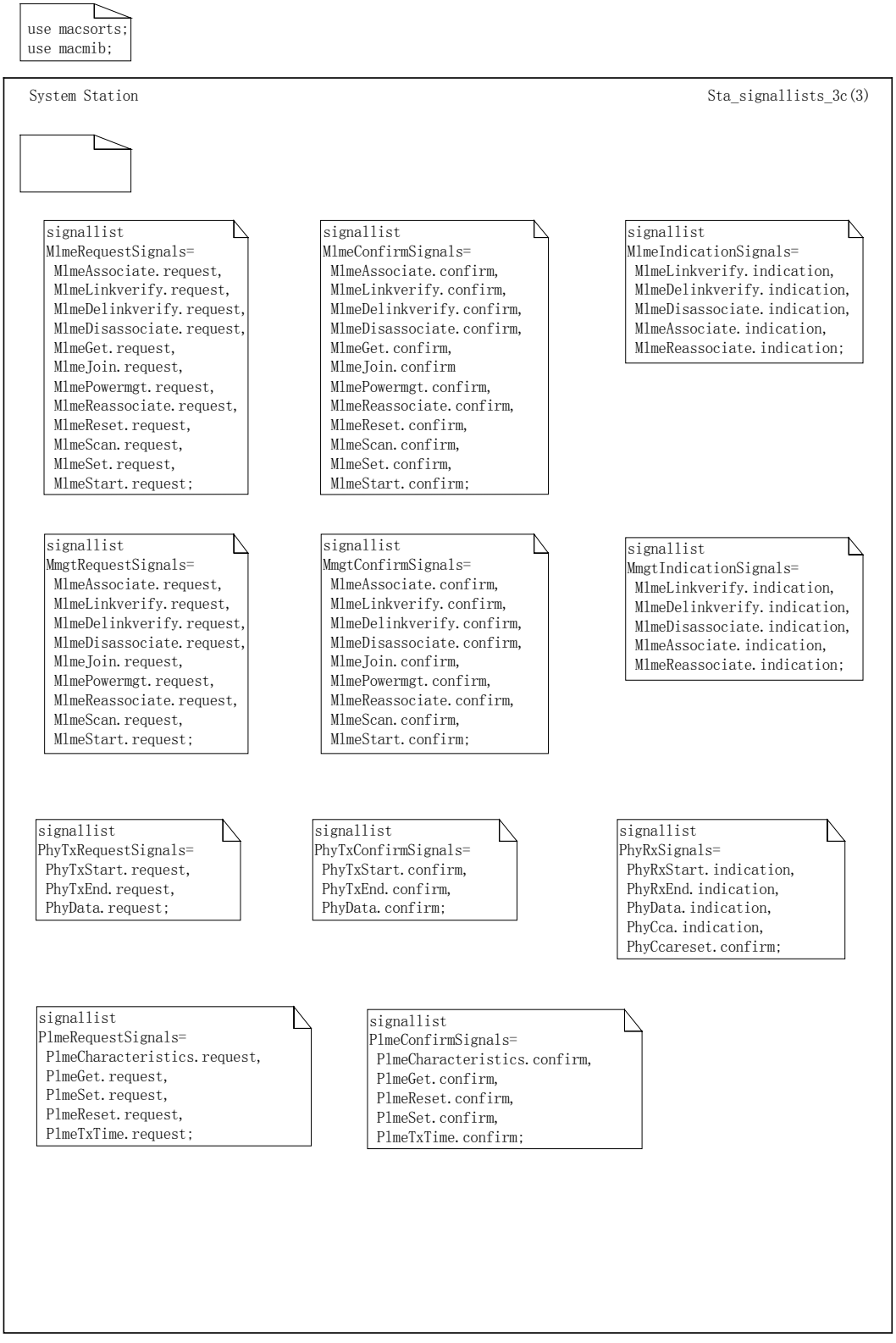
***Replace the diagram on Page 451 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

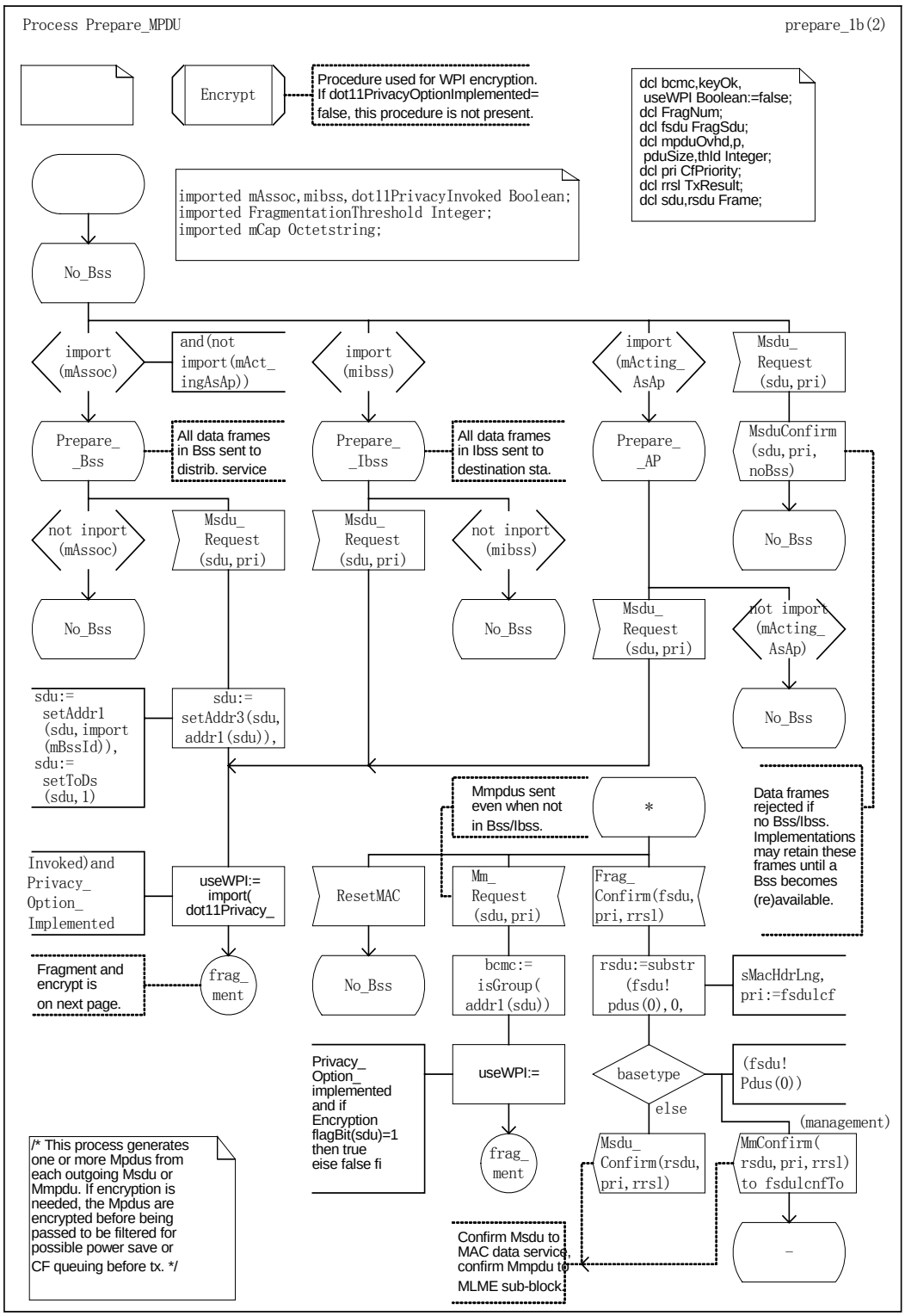
***Replace the diagram on Page 452 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

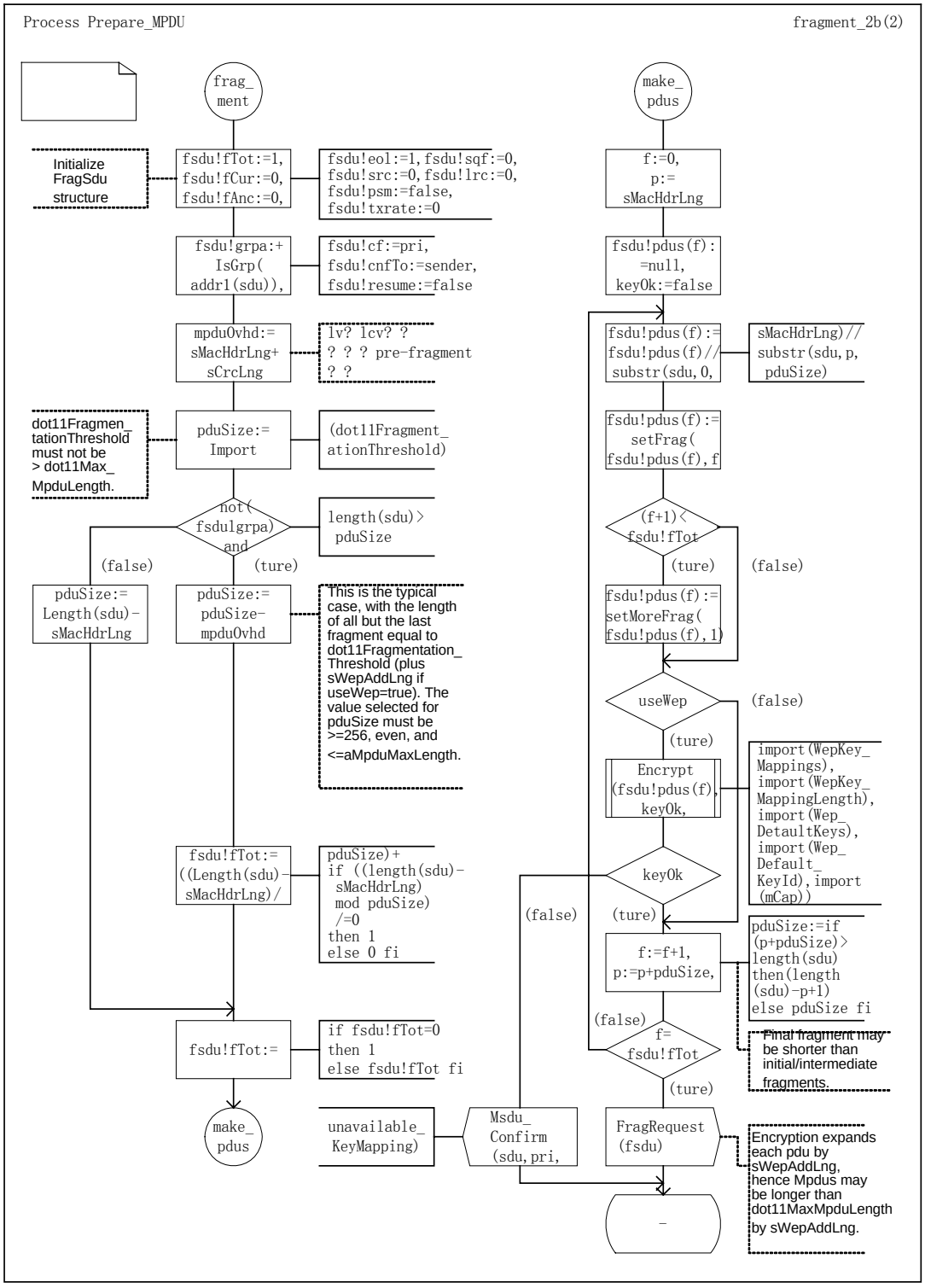
***Replace the diagram on Page 457 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

***Replace the diagram on Page 458 as follow:***

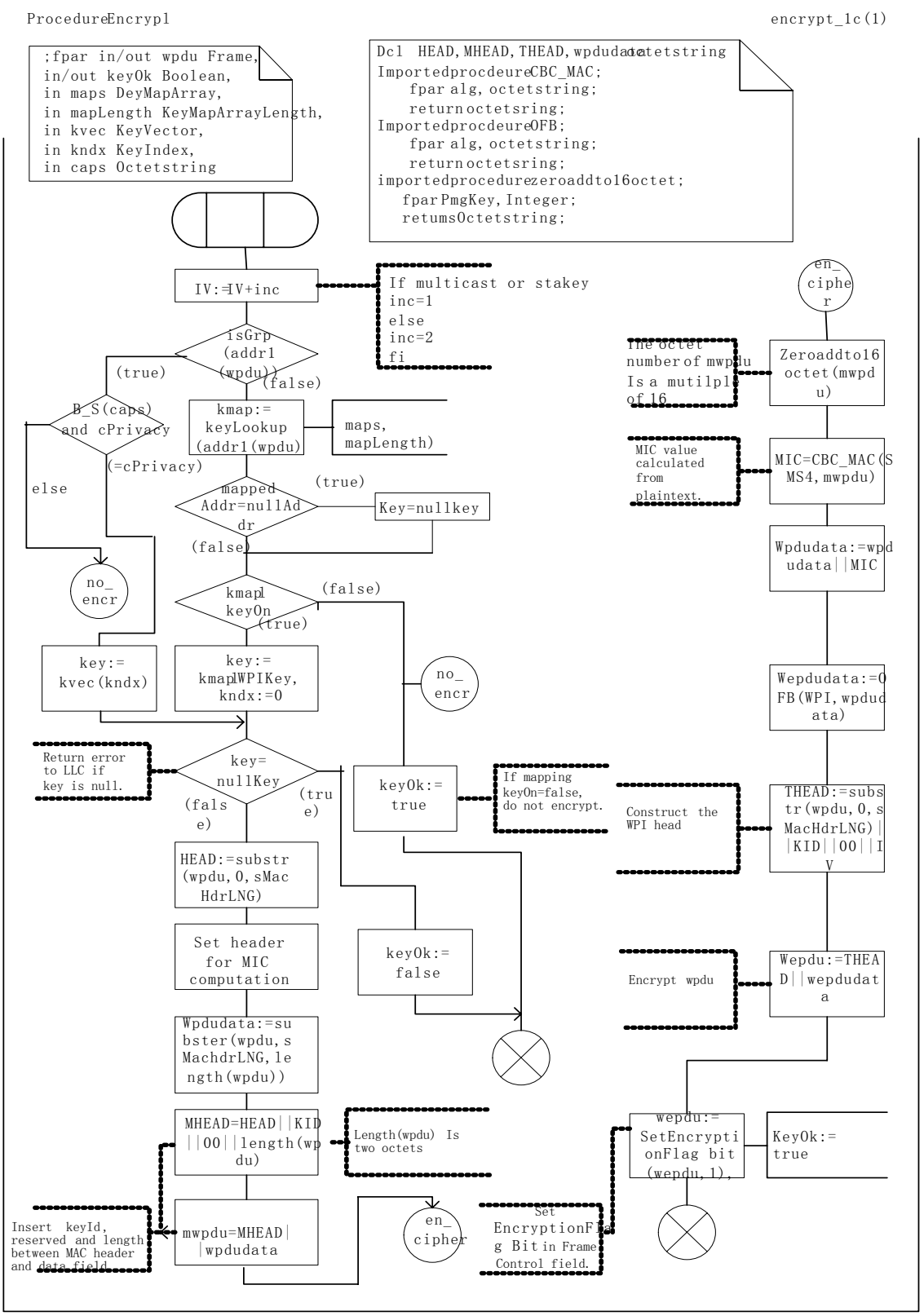





--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

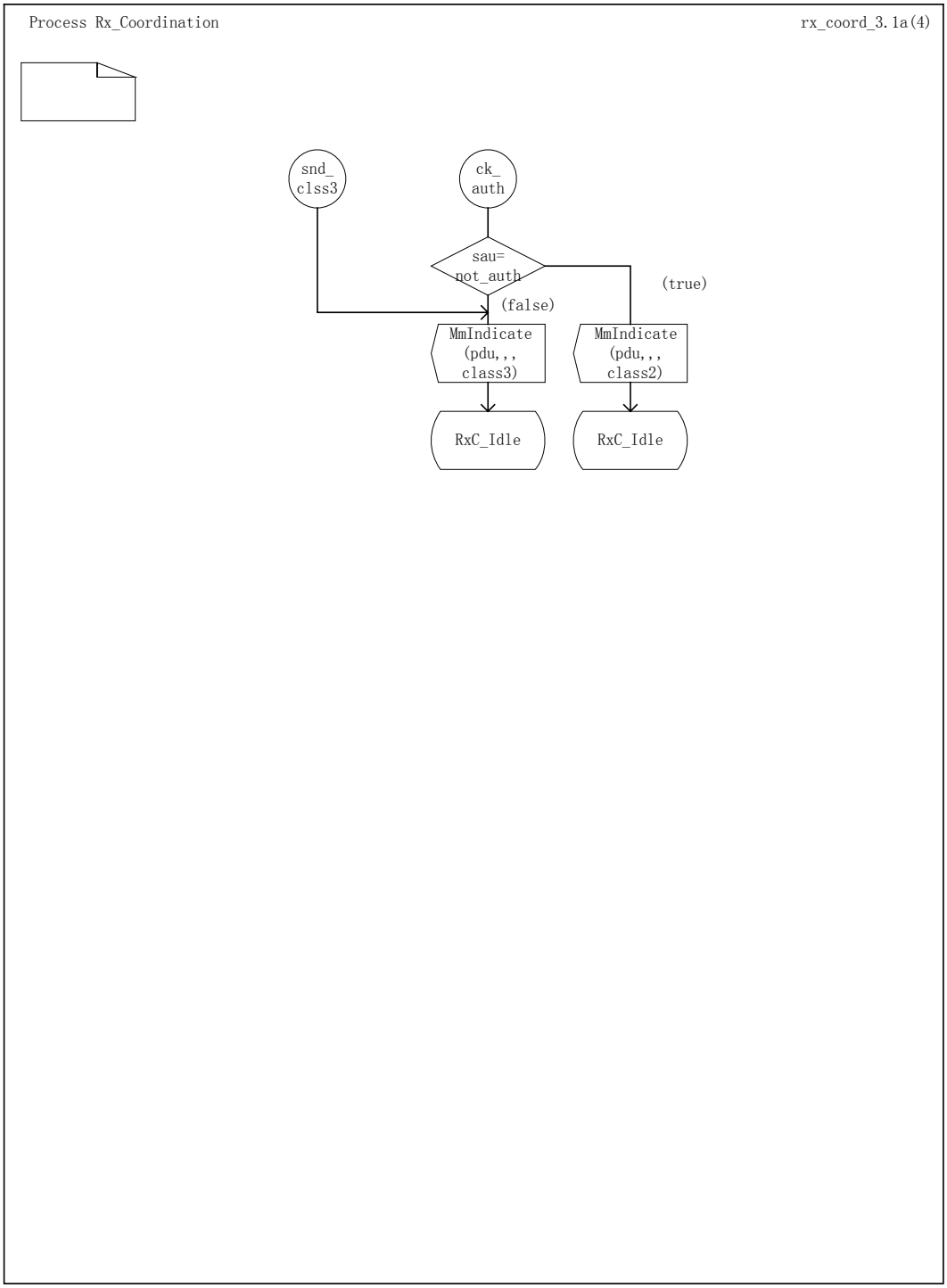
***Replace the diagram on Page 459 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

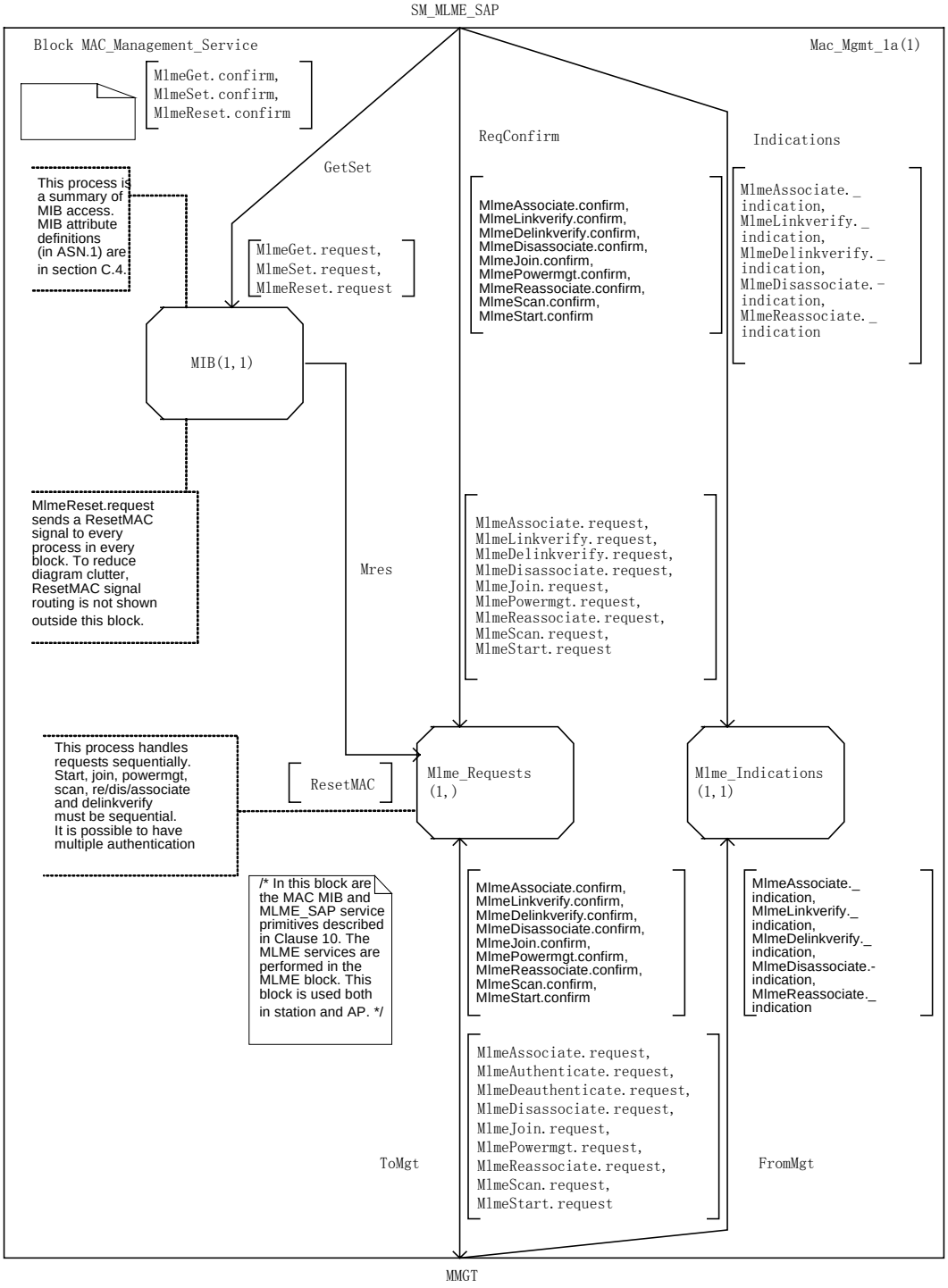
***Replace the diagram on Page 468 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

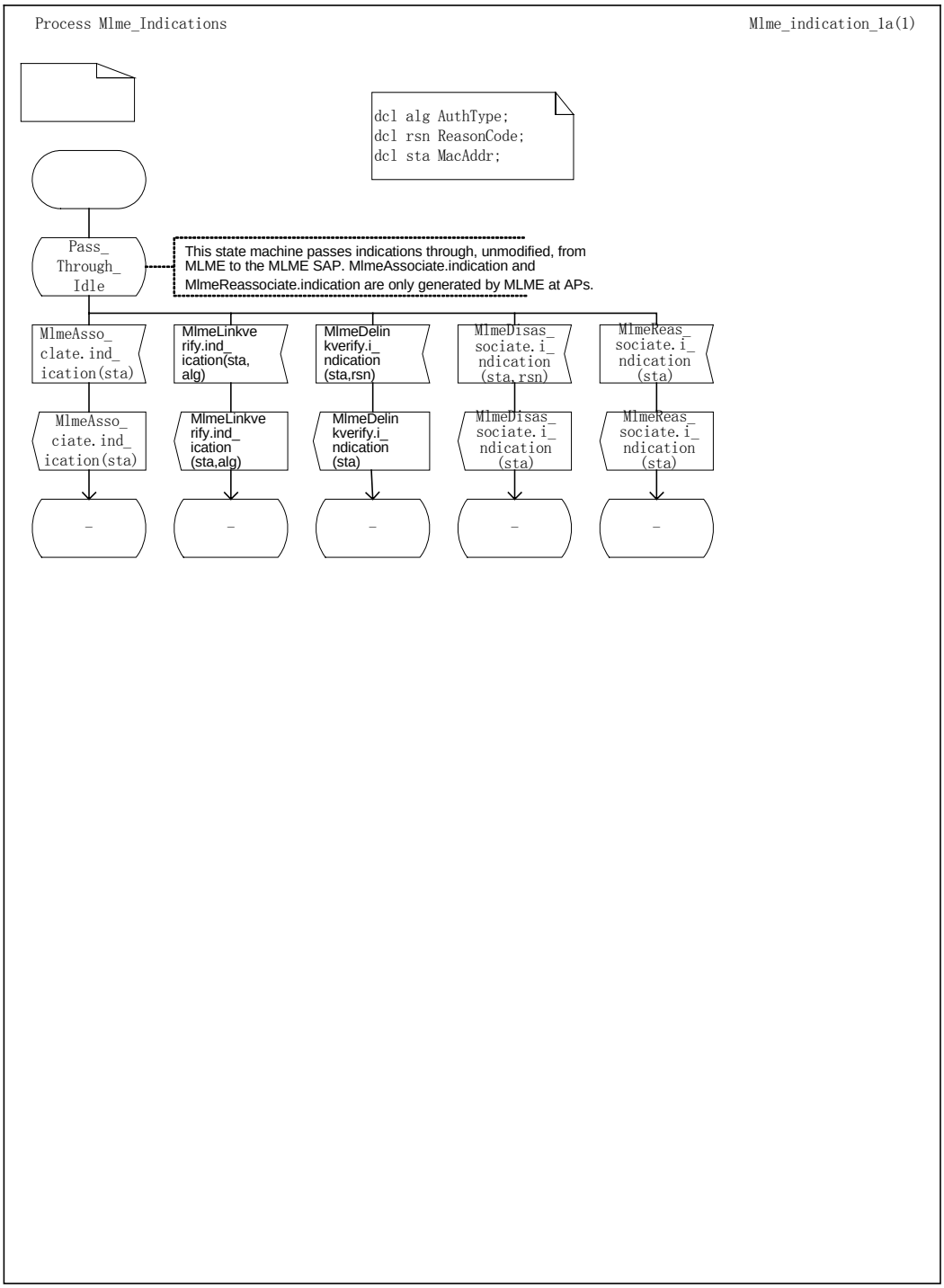
***Replace the diagram on Page 483 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

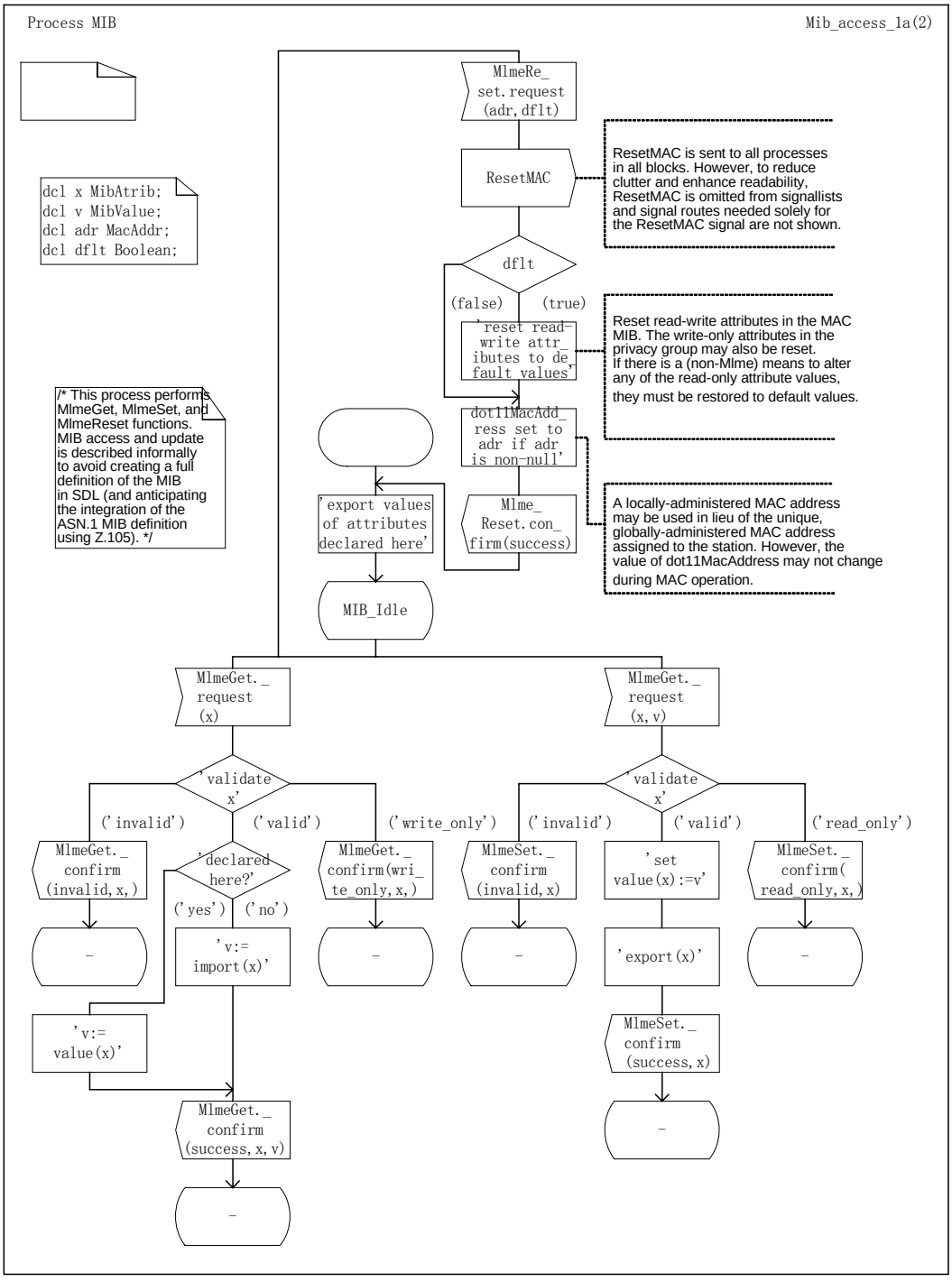
***Replace the diagram on Page 484 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

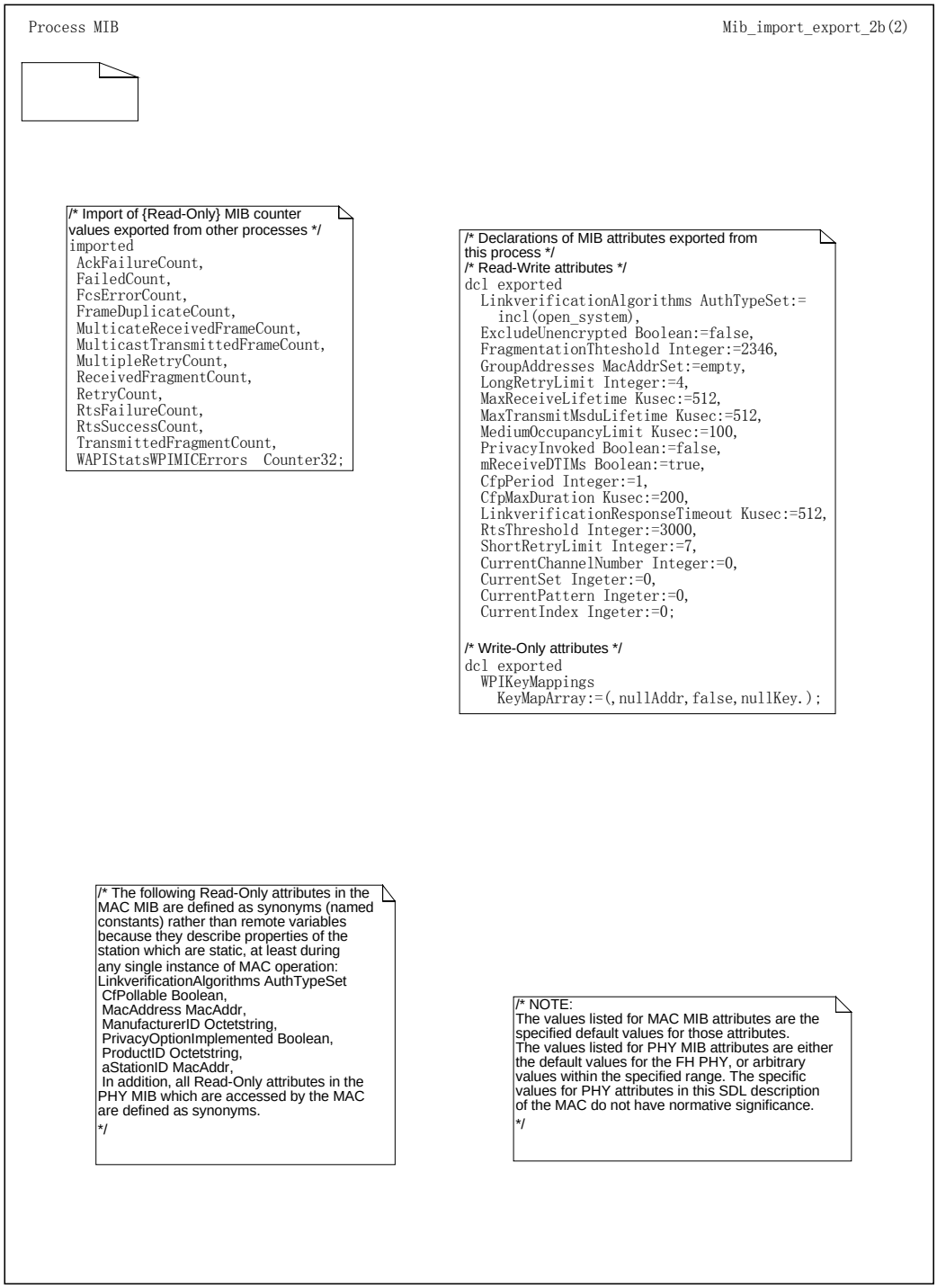
***Replace the diagram on Page 485 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

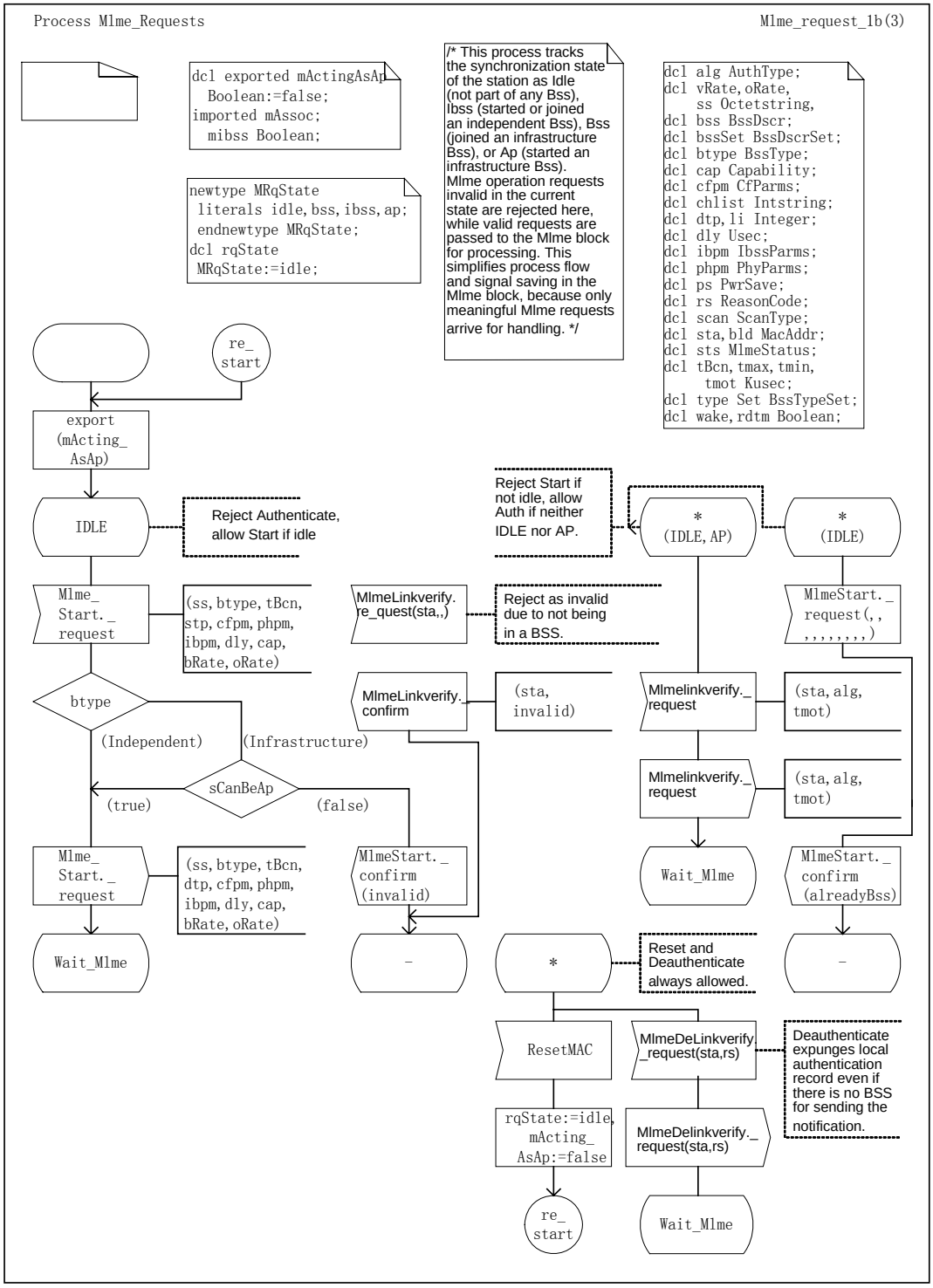
***Replace the diagram on Page 486 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

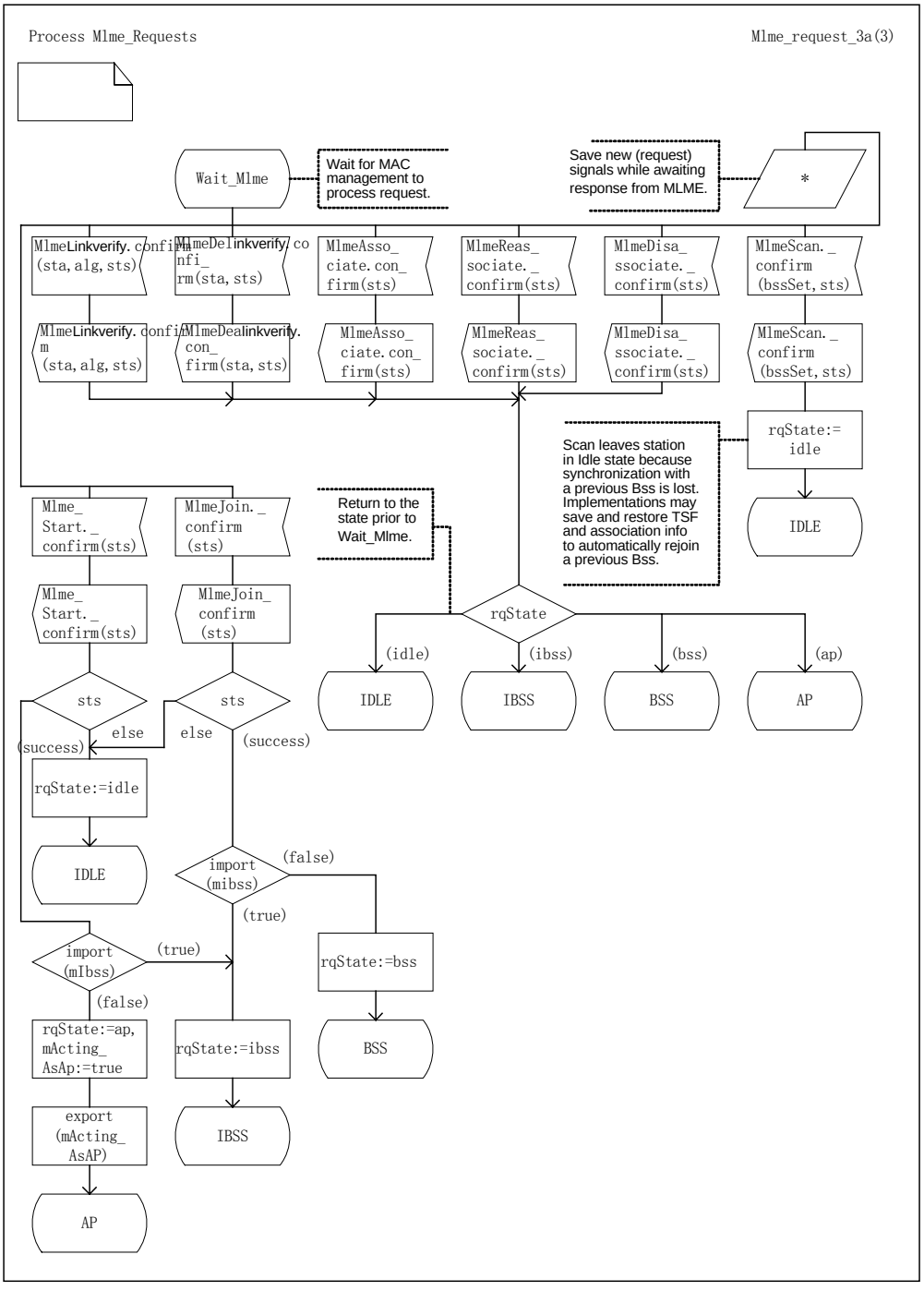
***Replace the diagram on Page 487 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

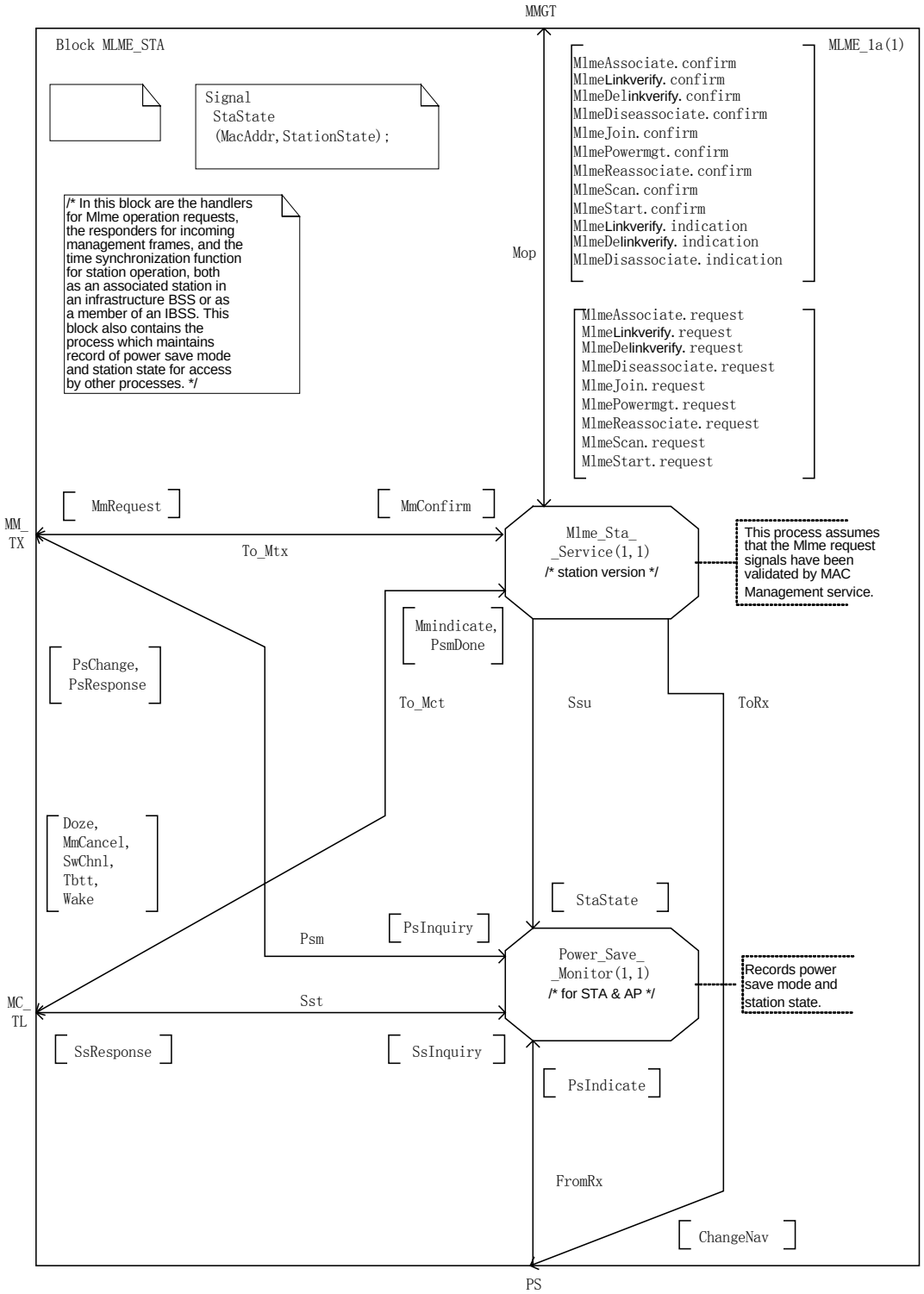
***Replace the diagram on Page 489 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

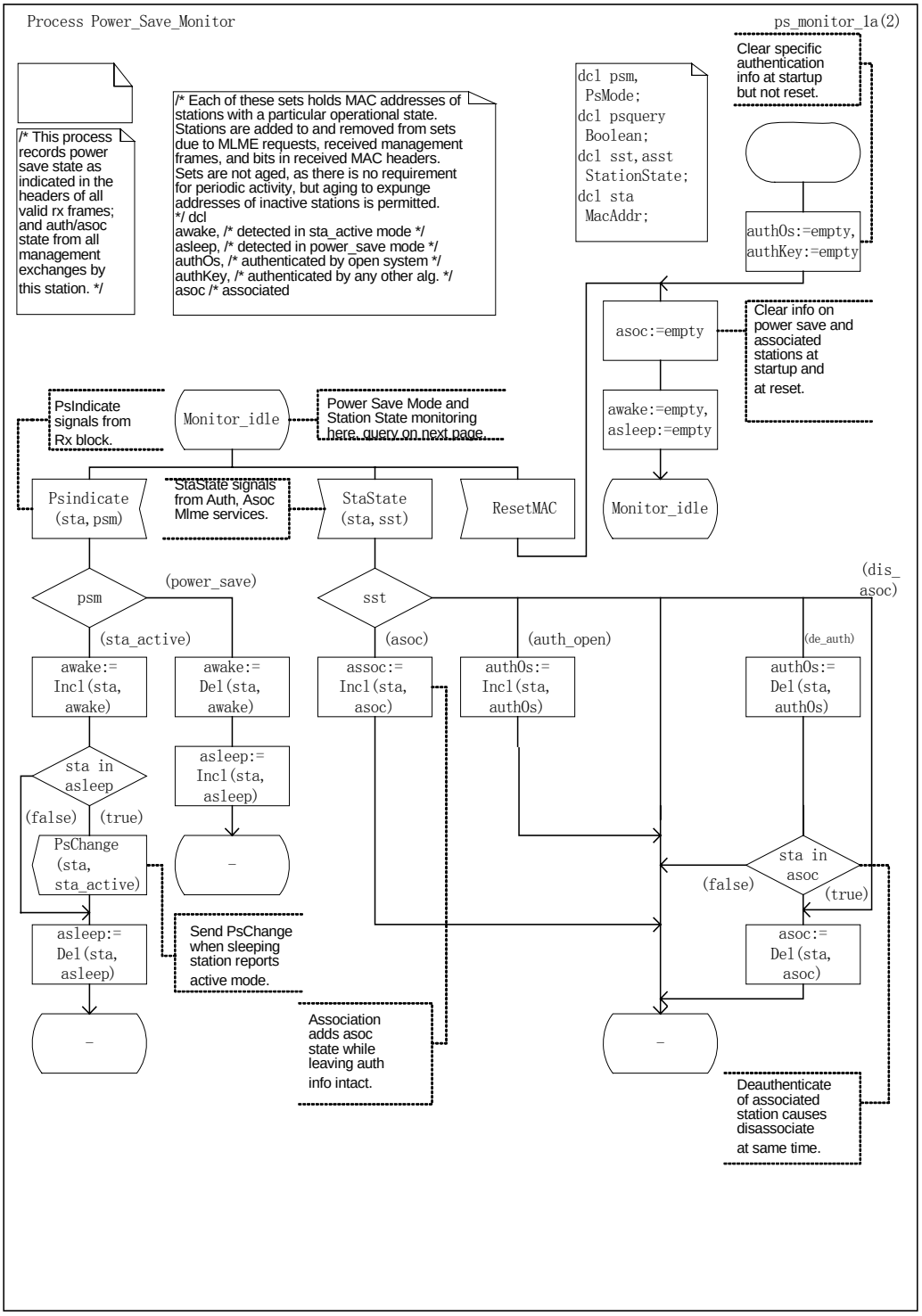
***Replace the diagram on Page 490 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

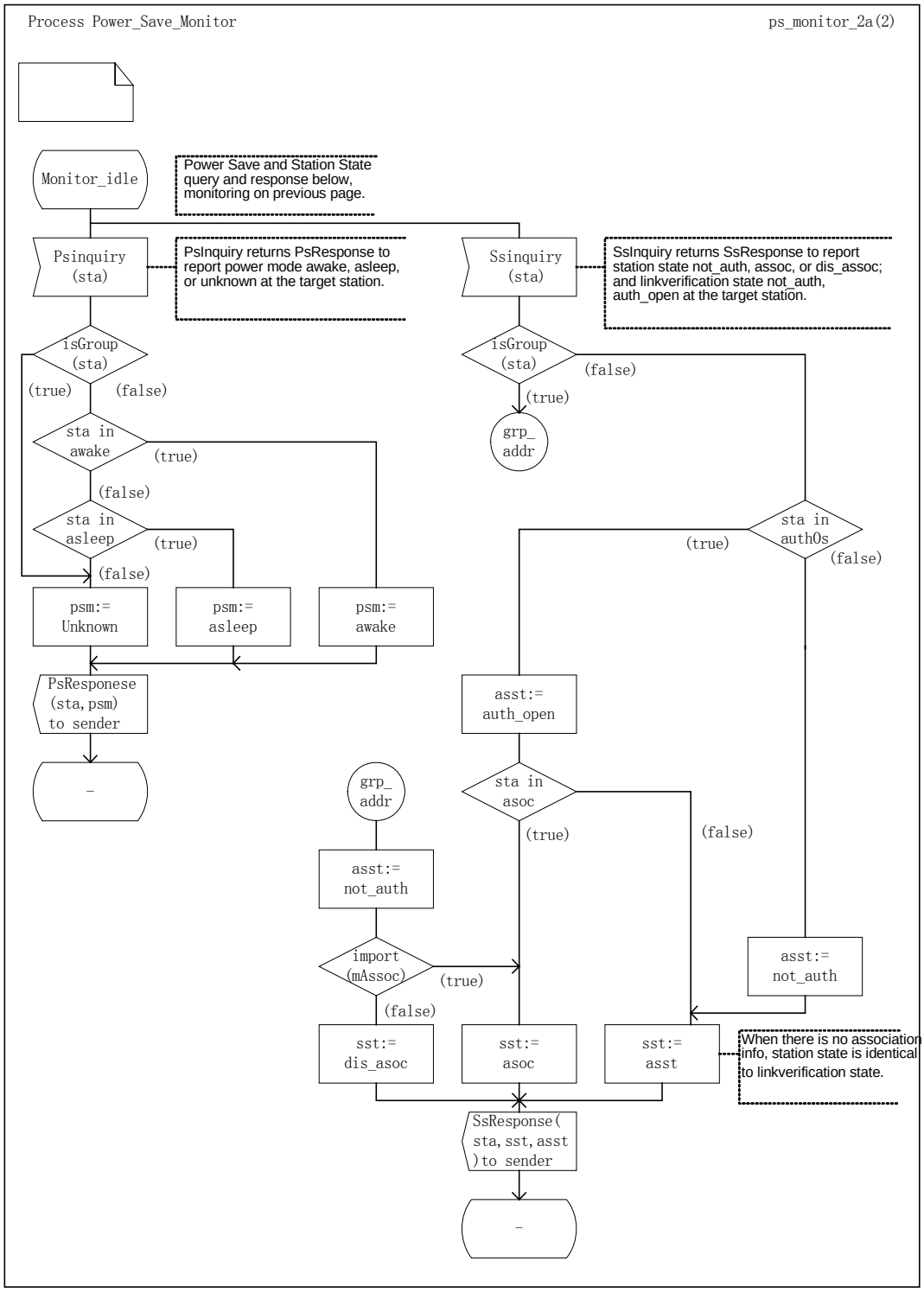
***Replace the diagram on Page 491 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

***Replace the diagram on Page 492 as follow:***

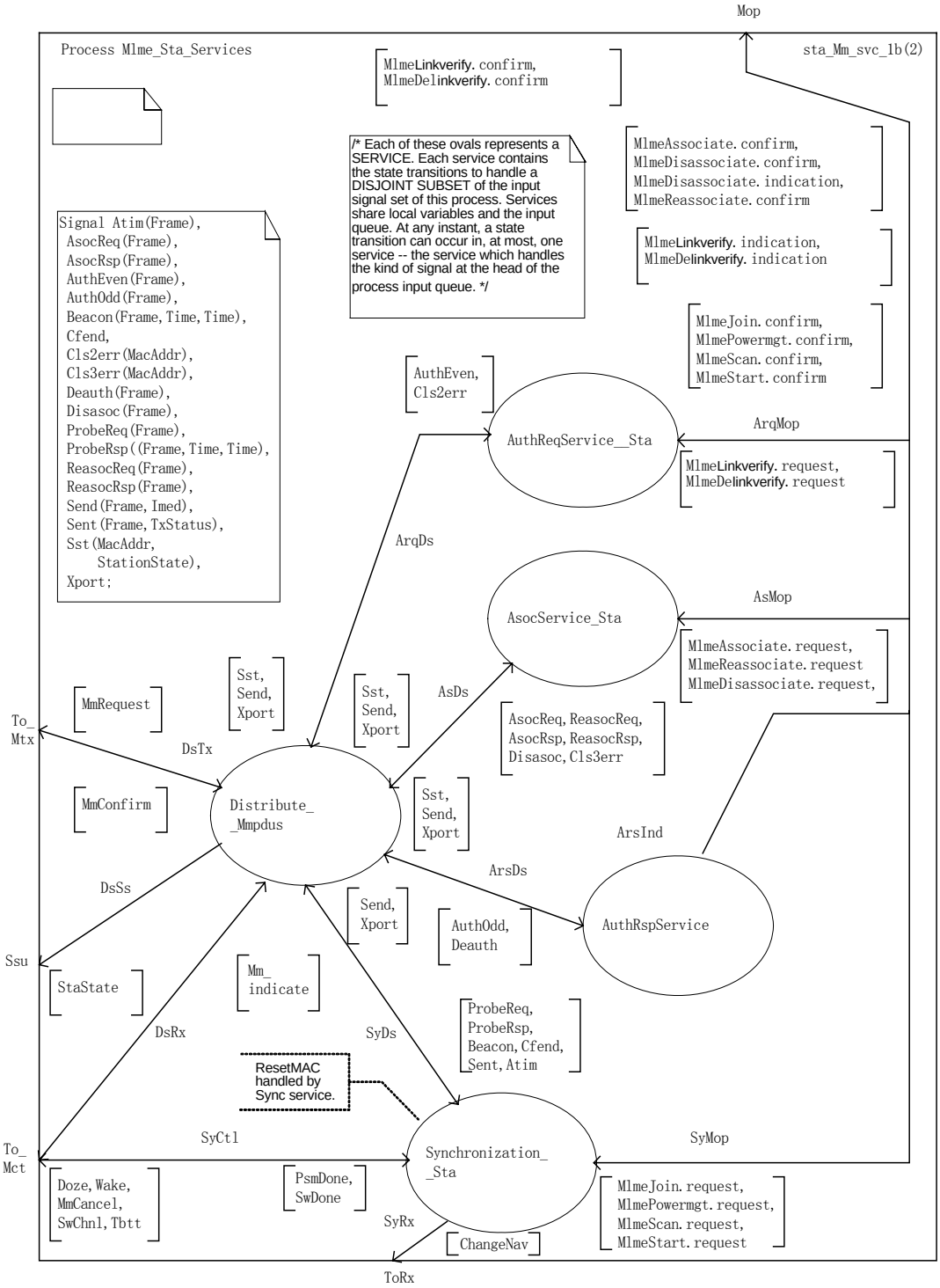





--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

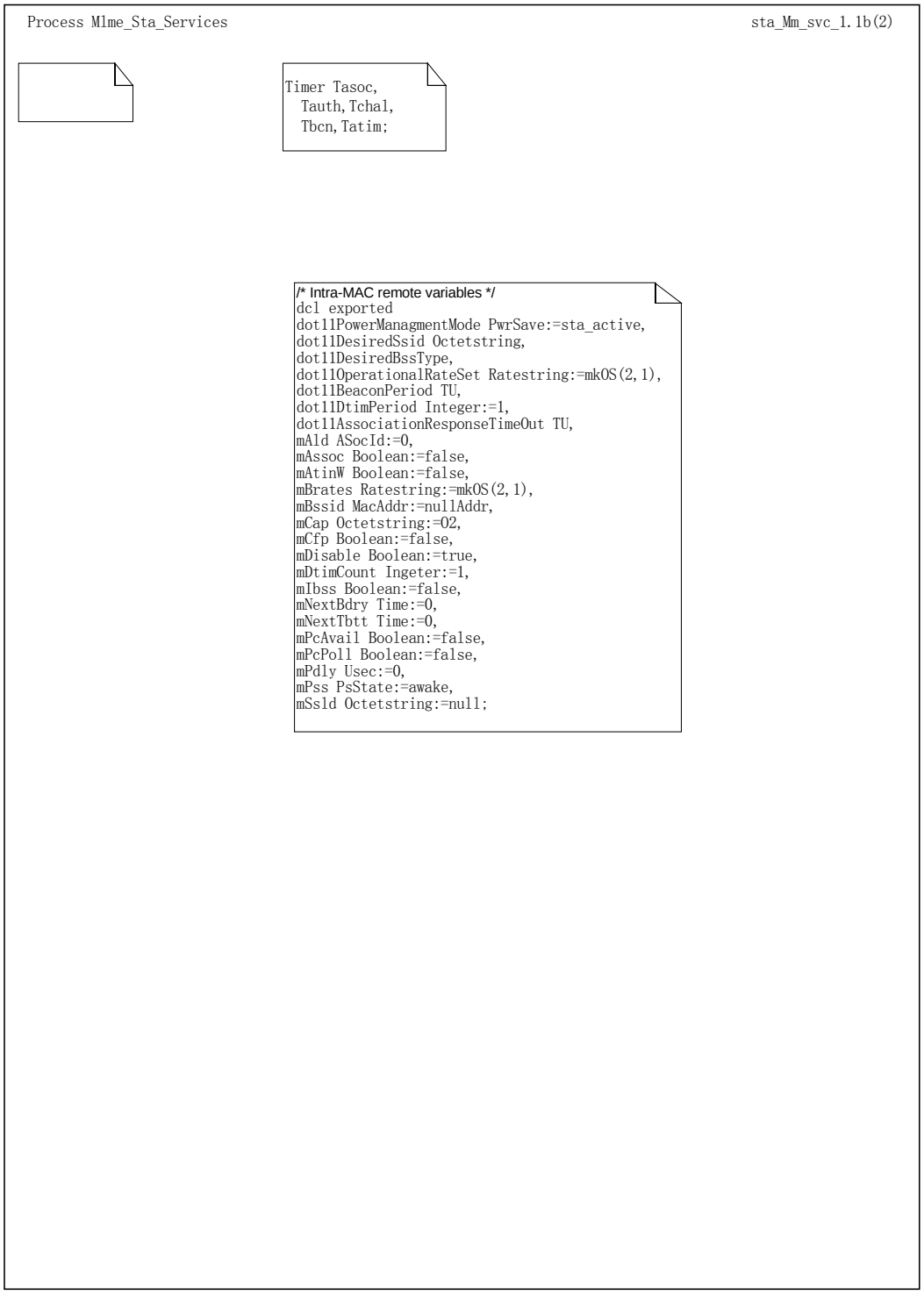
***Replace the diagram on Page 493 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

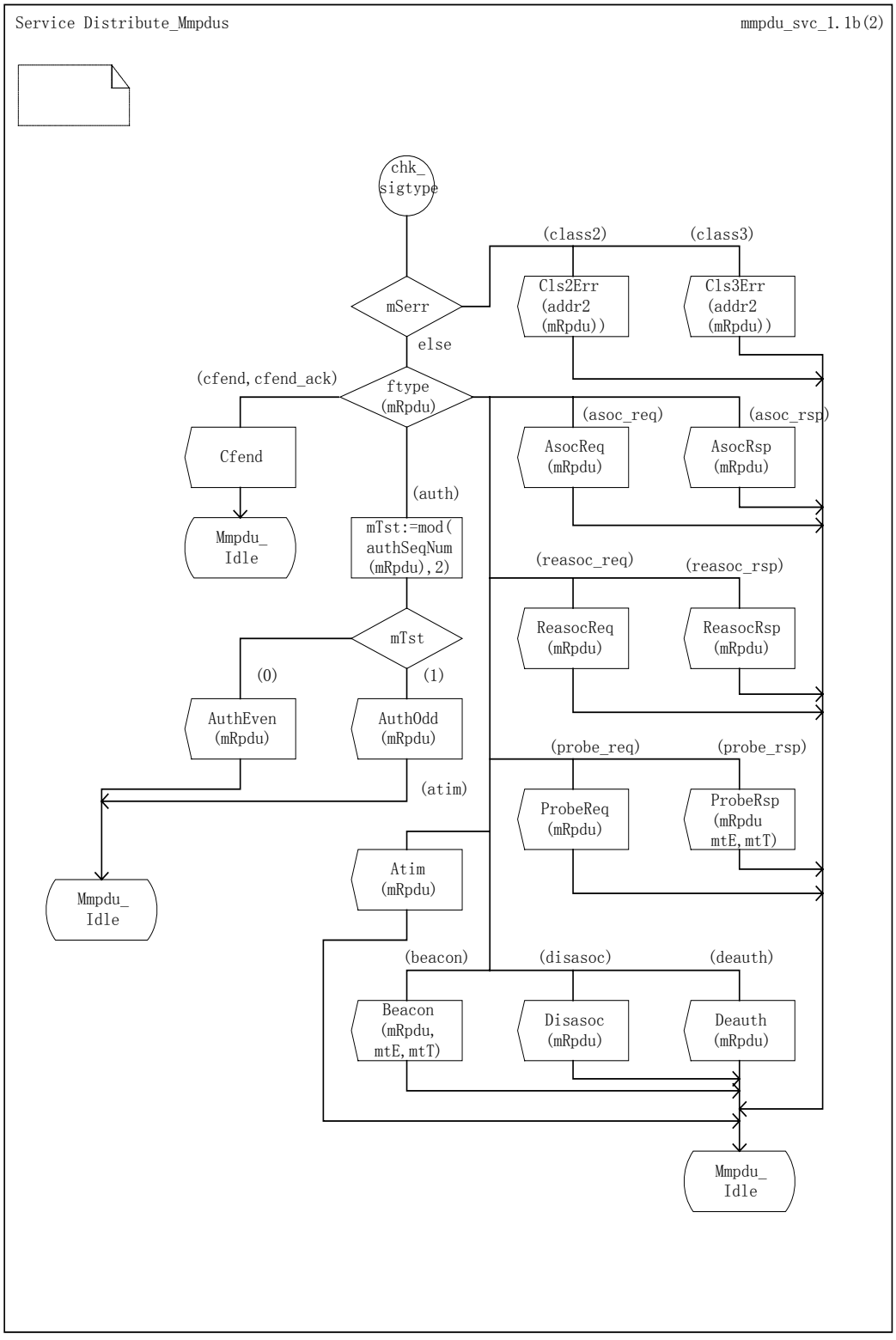
***Replace the diagram on Page 494 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

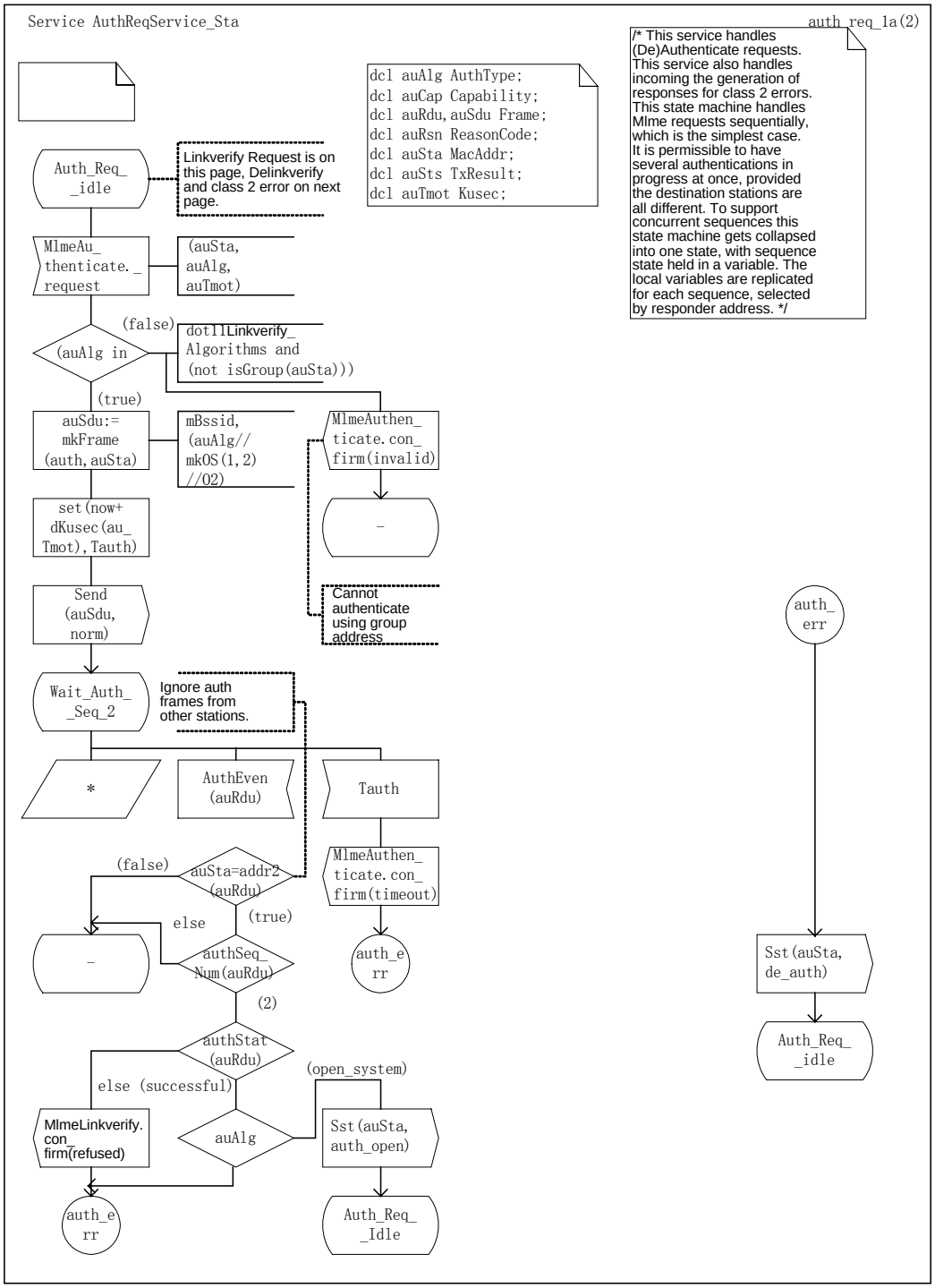
***Replace the diagram on Page 496 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

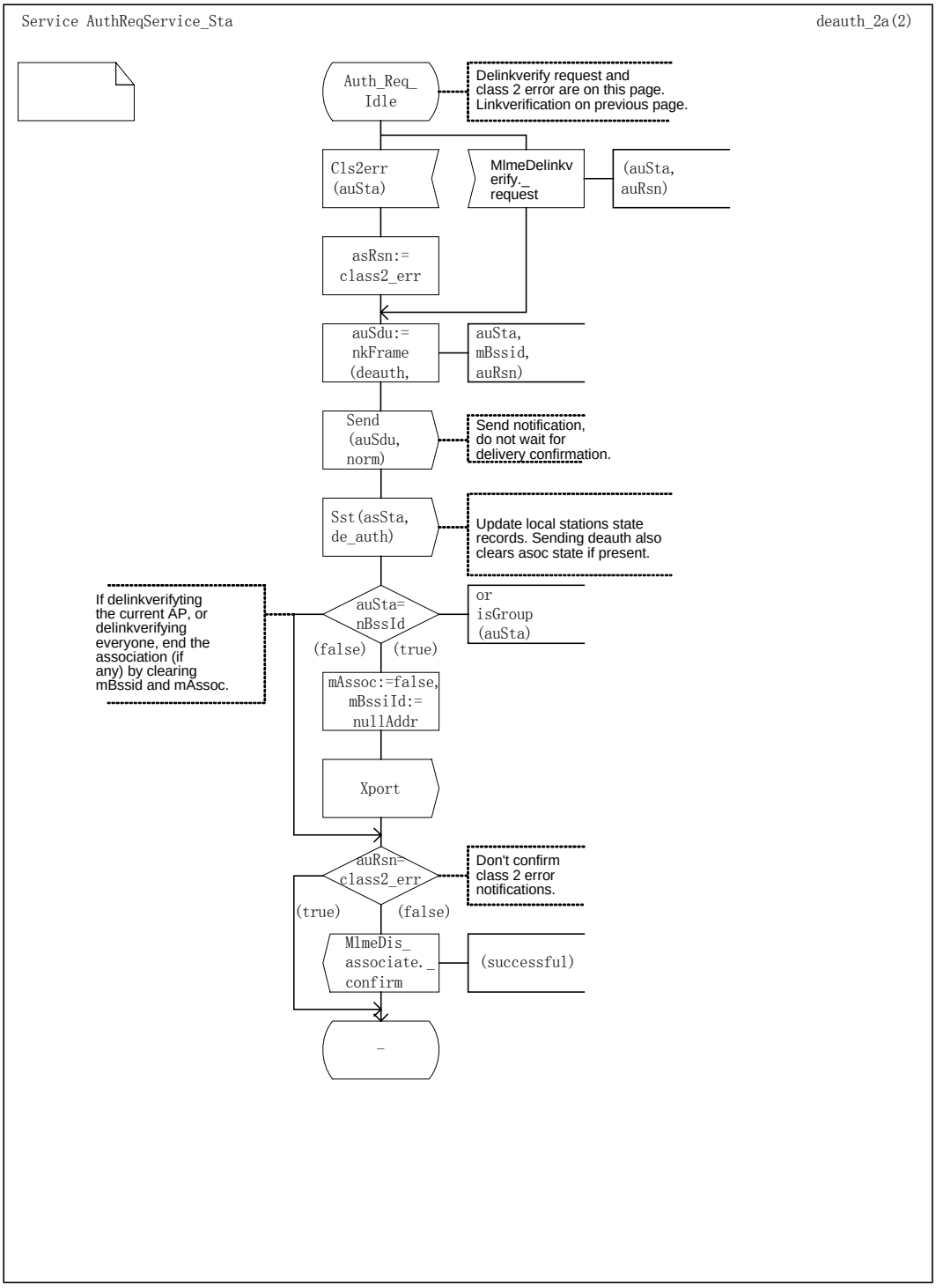
***Replace the diagram on Page 497 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

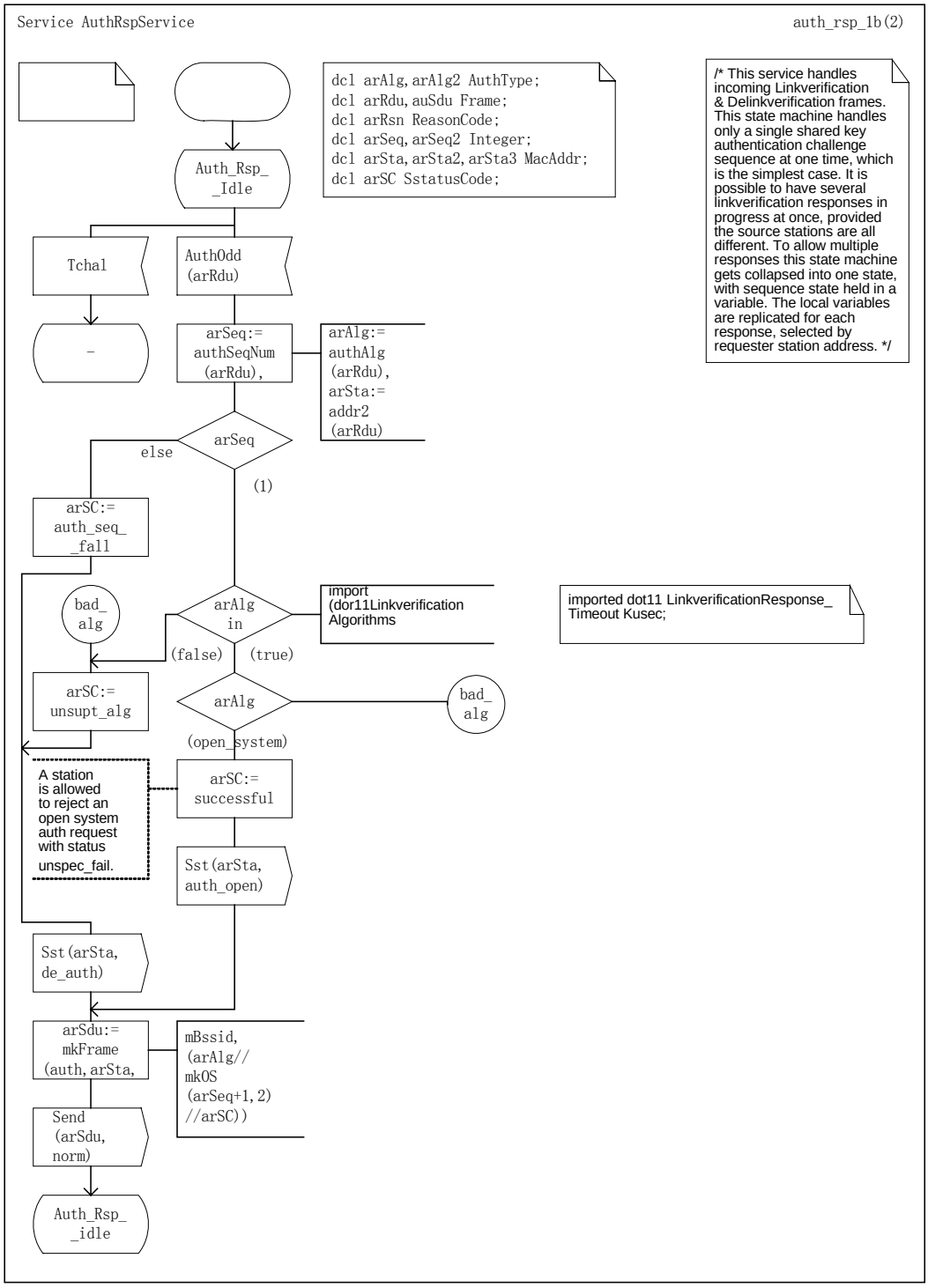
***Replace the diagram on Page 498 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

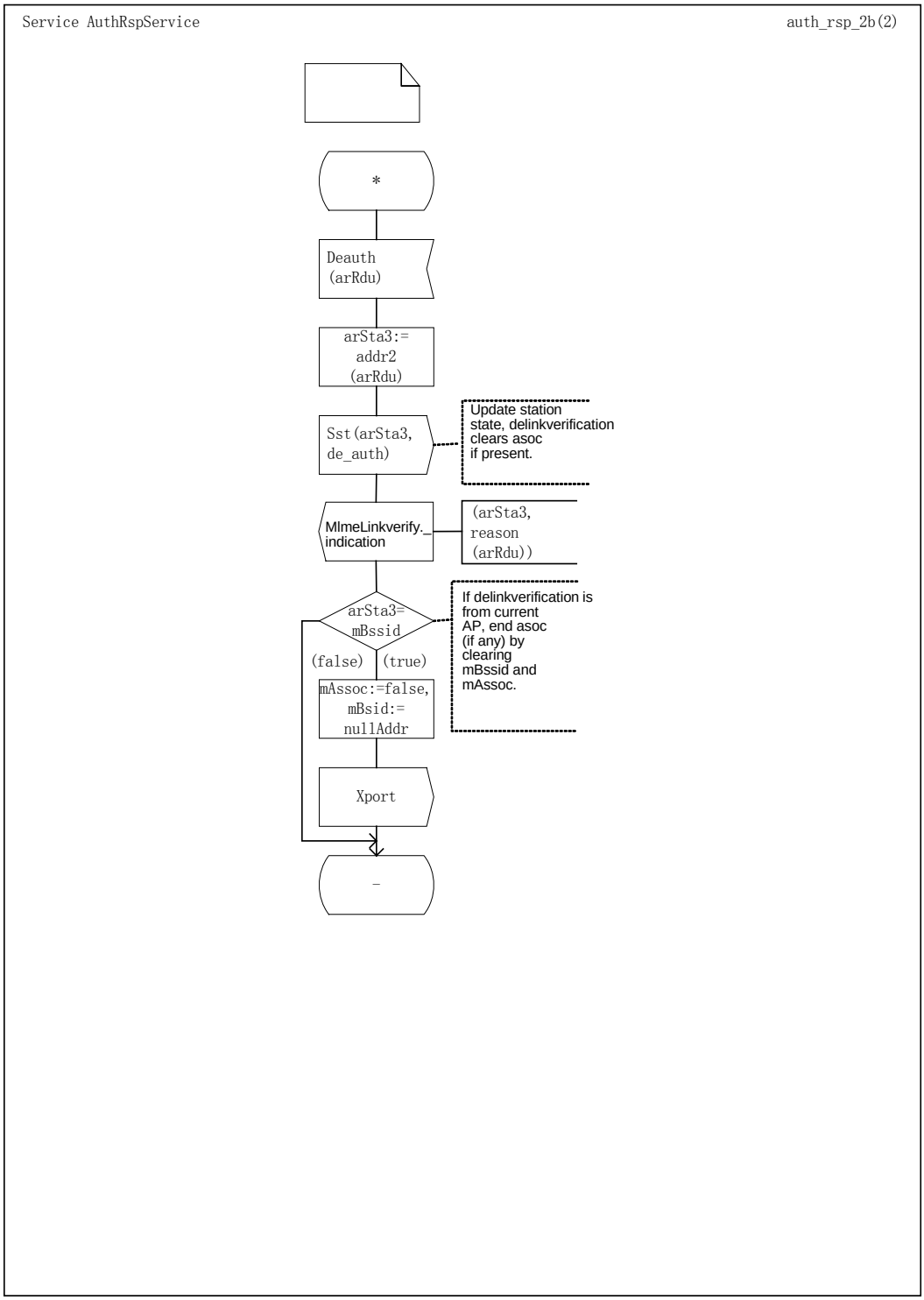
***Replace the diagram on Page 502 as follow:***






--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

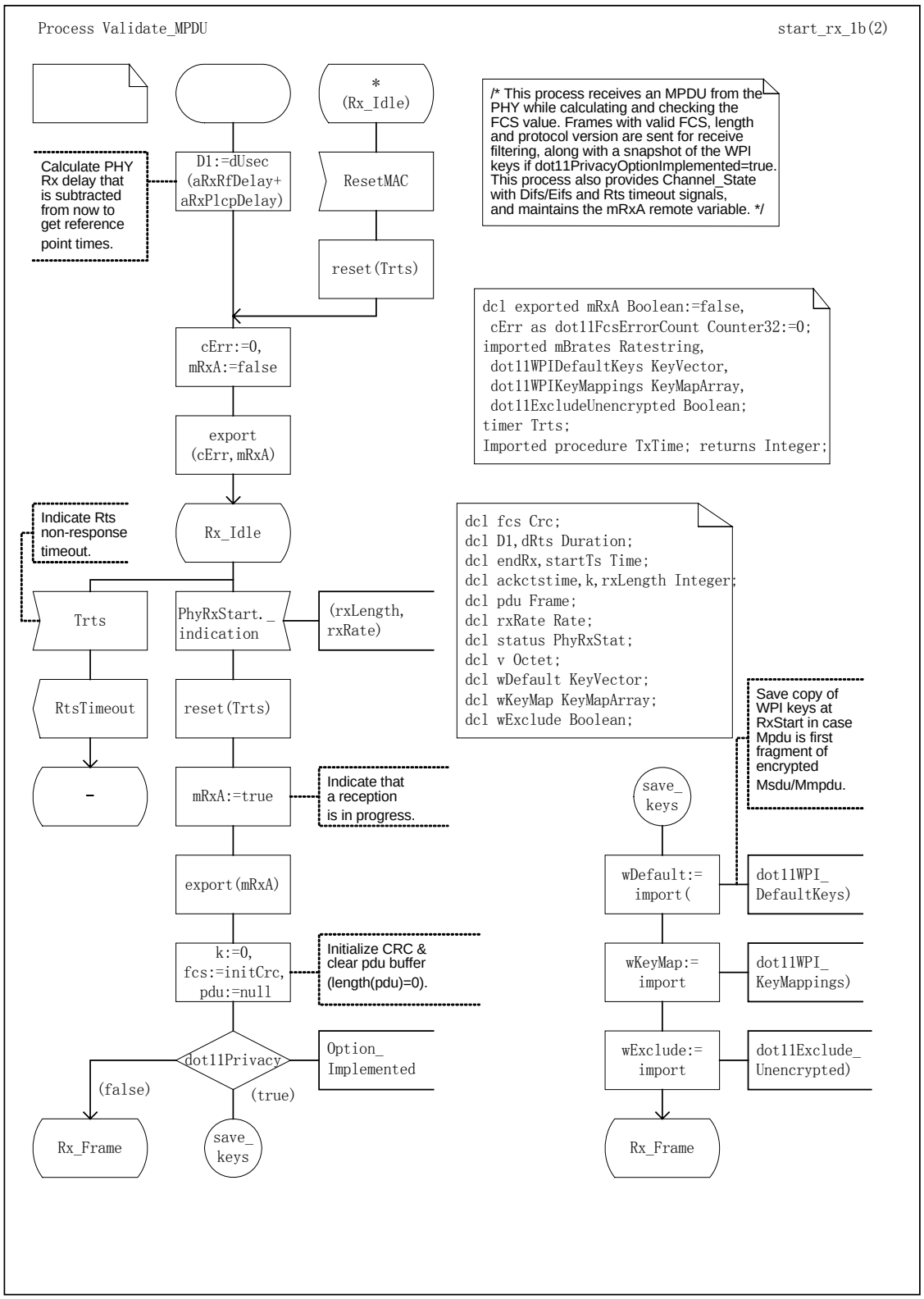
***Replace the diagram on Page 503 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

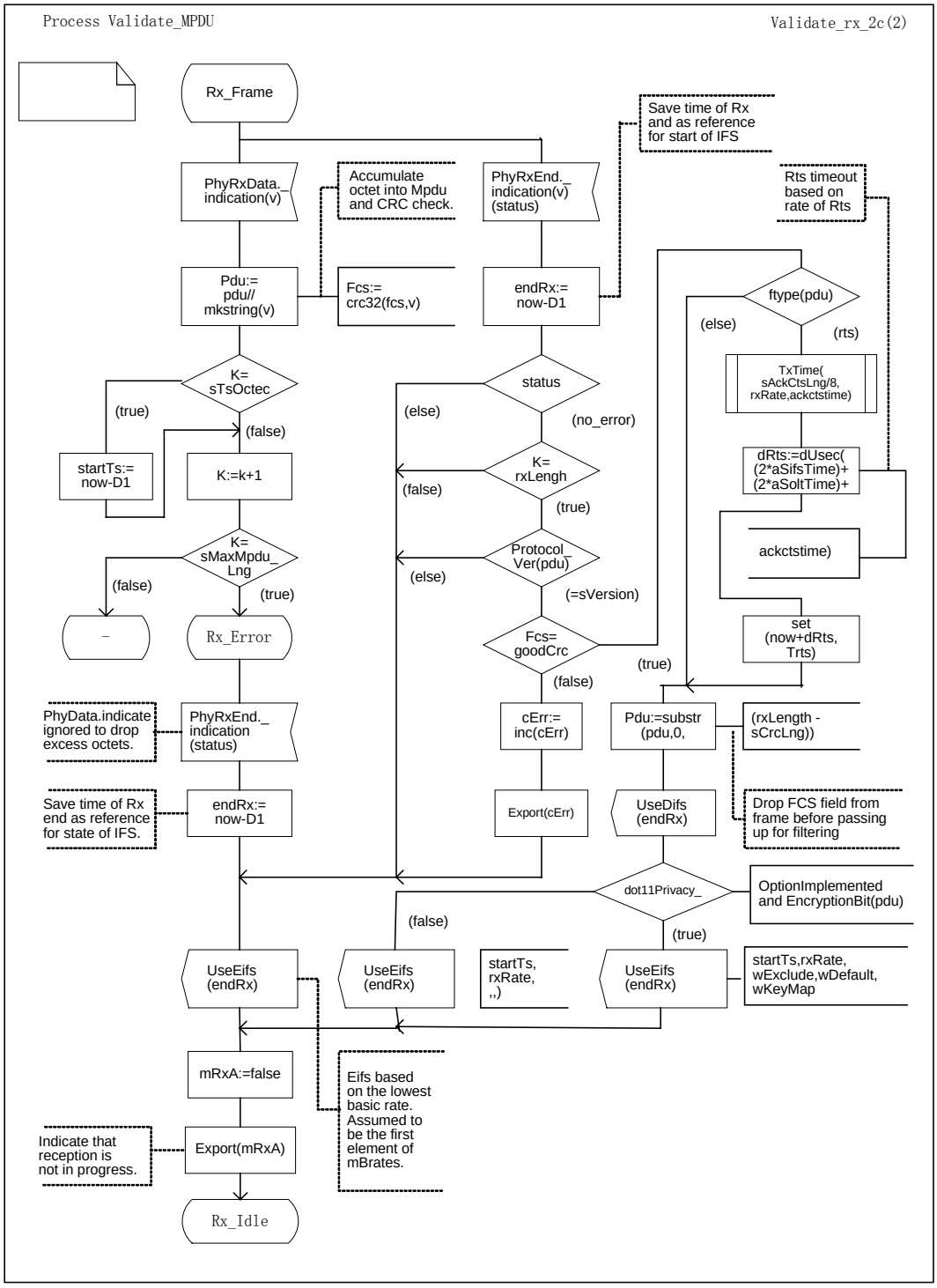
***Replace the diagram on Page 515 as follow:***






--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

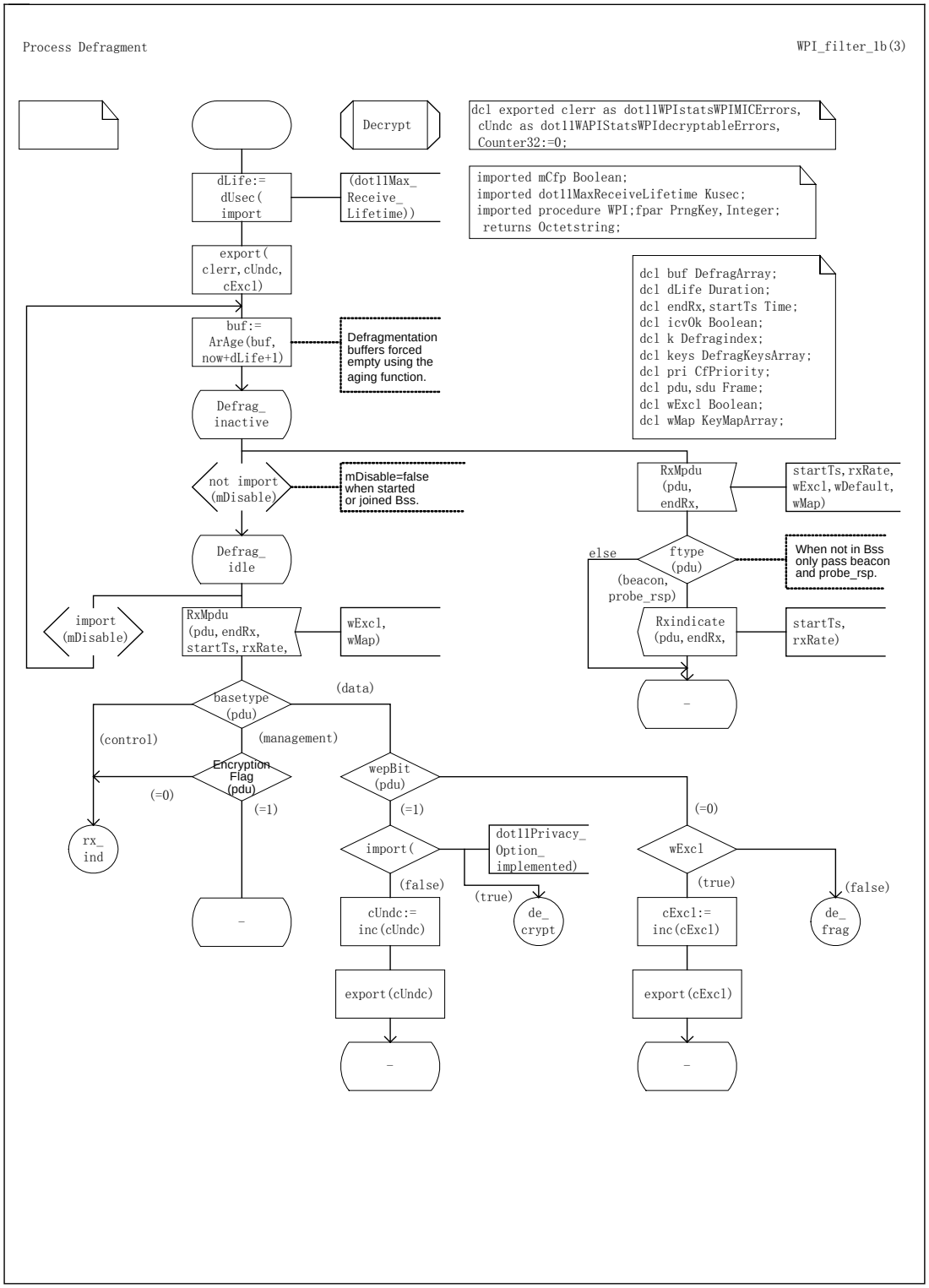
***Replace the diagram on Page 516 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

***Replace the diagram on Page 521 as follow:***

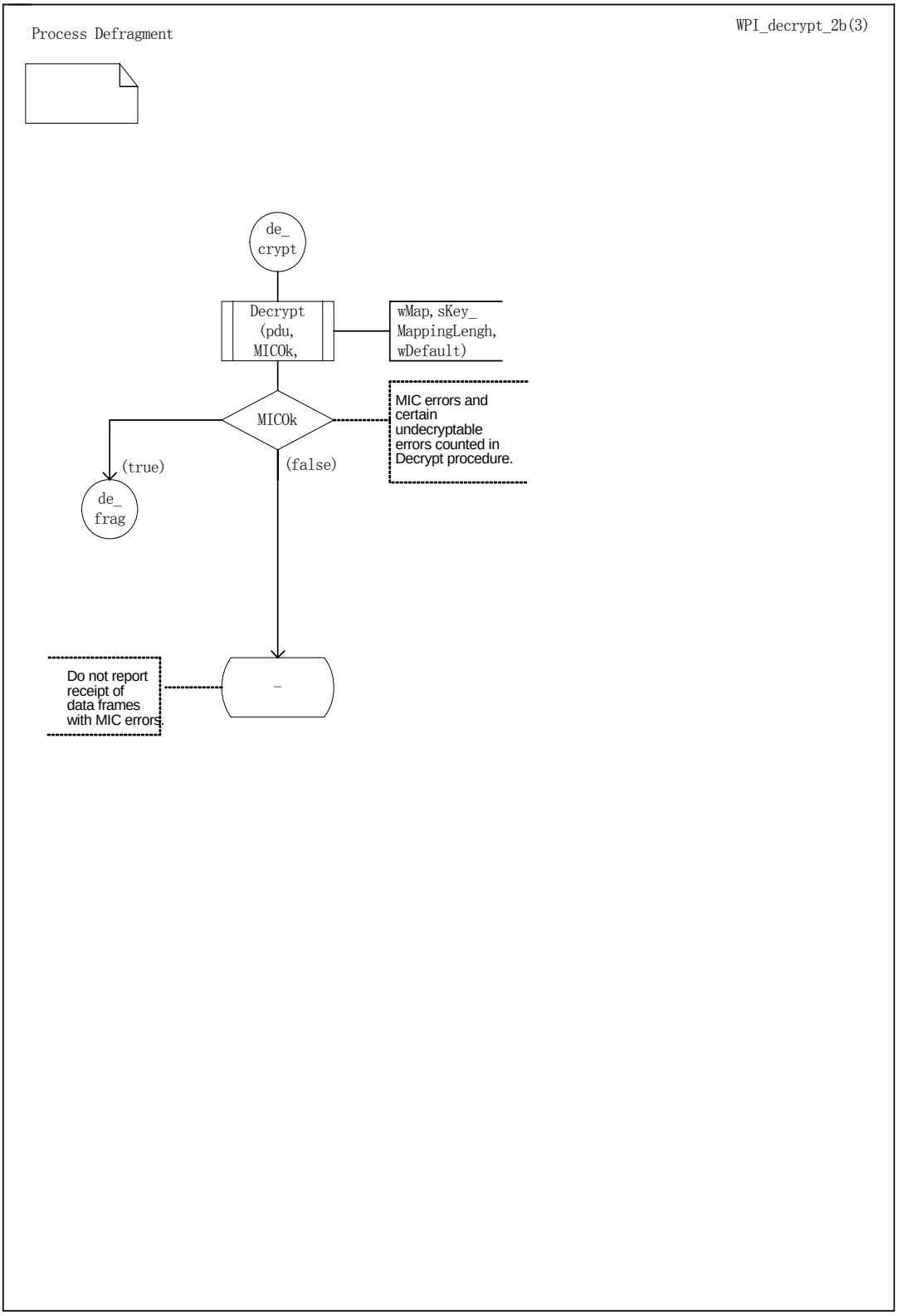





--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

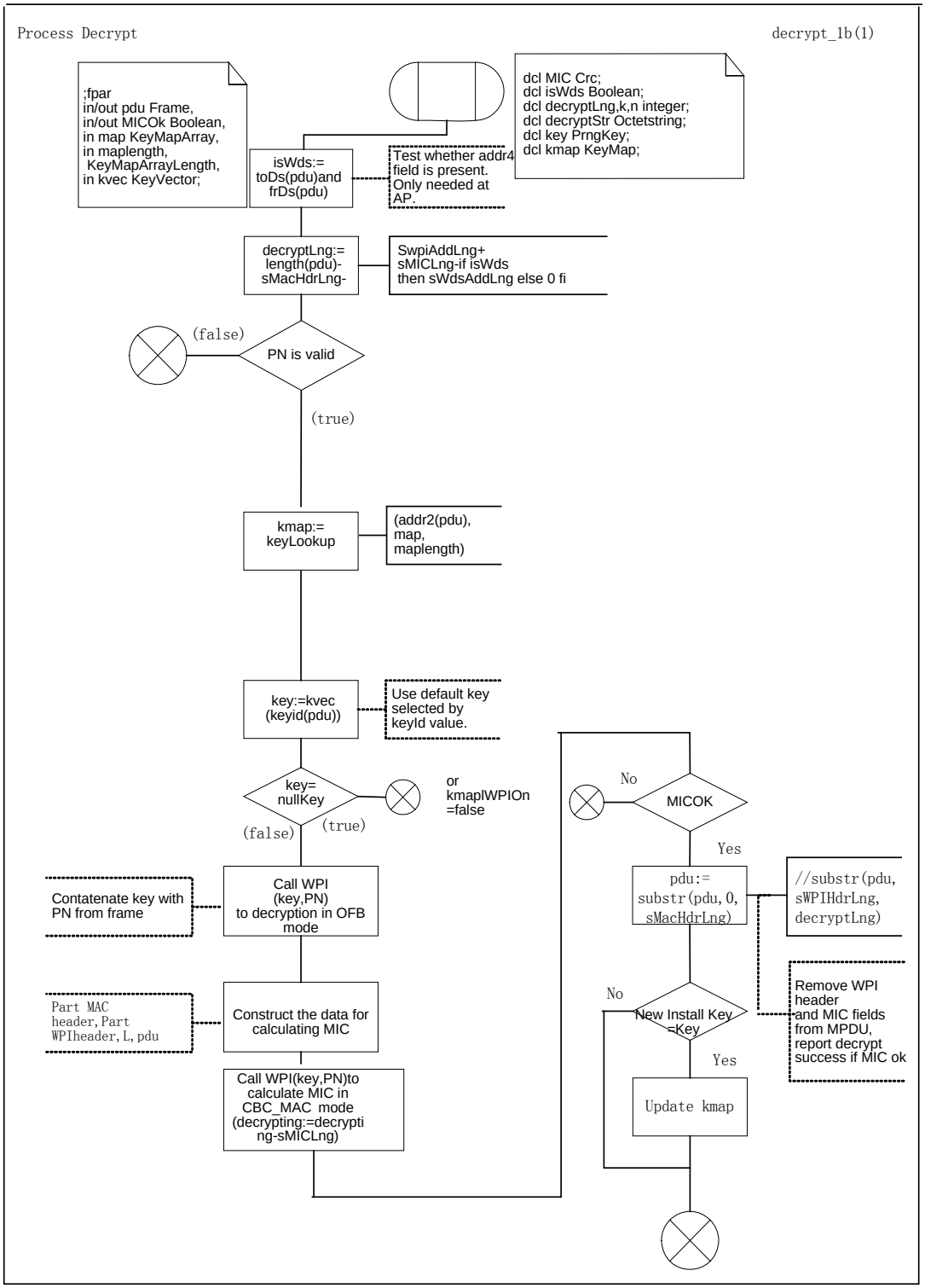
***Replace the diagram on Page 522 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

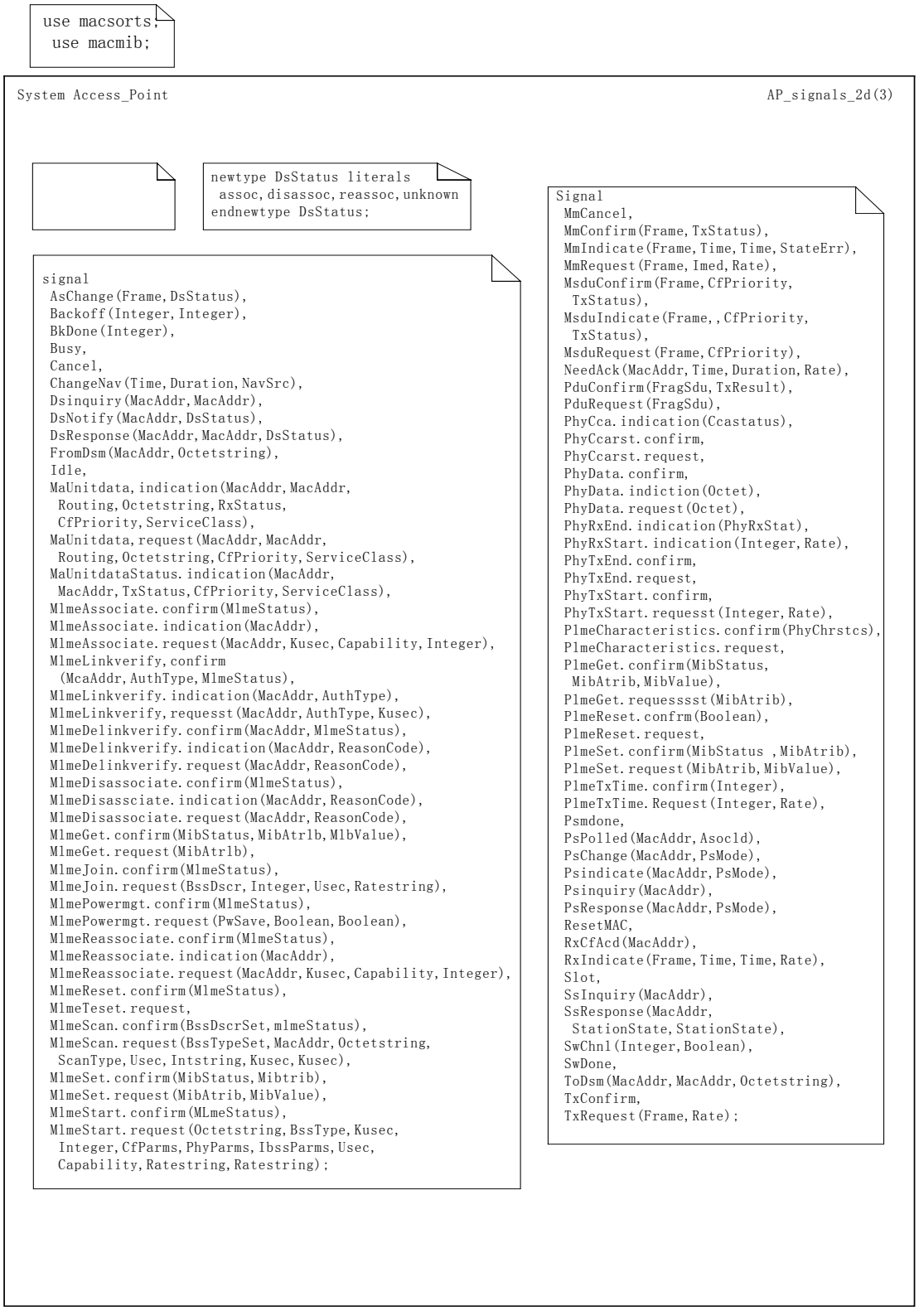
***Replace the diagram on Page 524 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

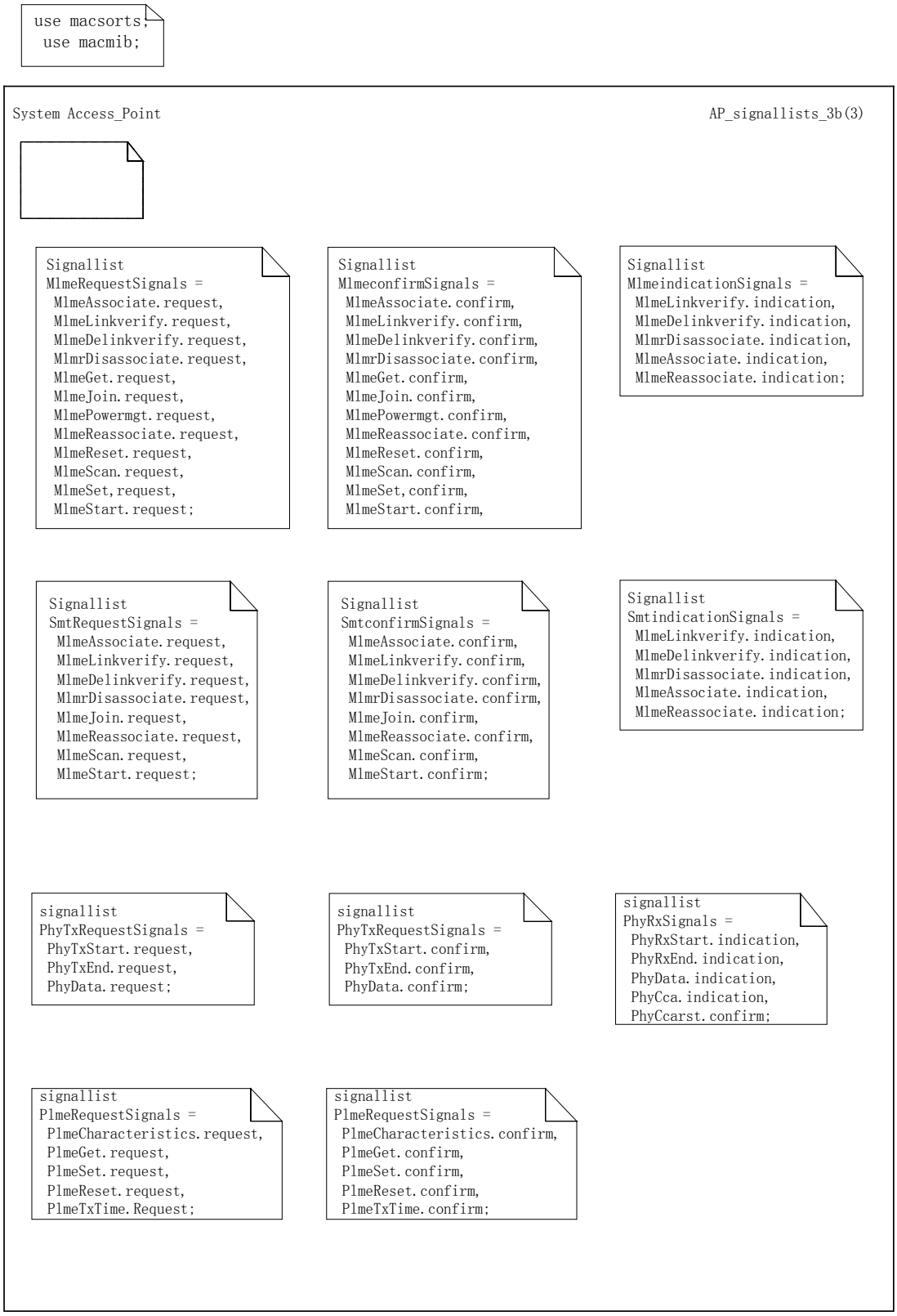
***Replace the diagram on Page 527 as follow:***






--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

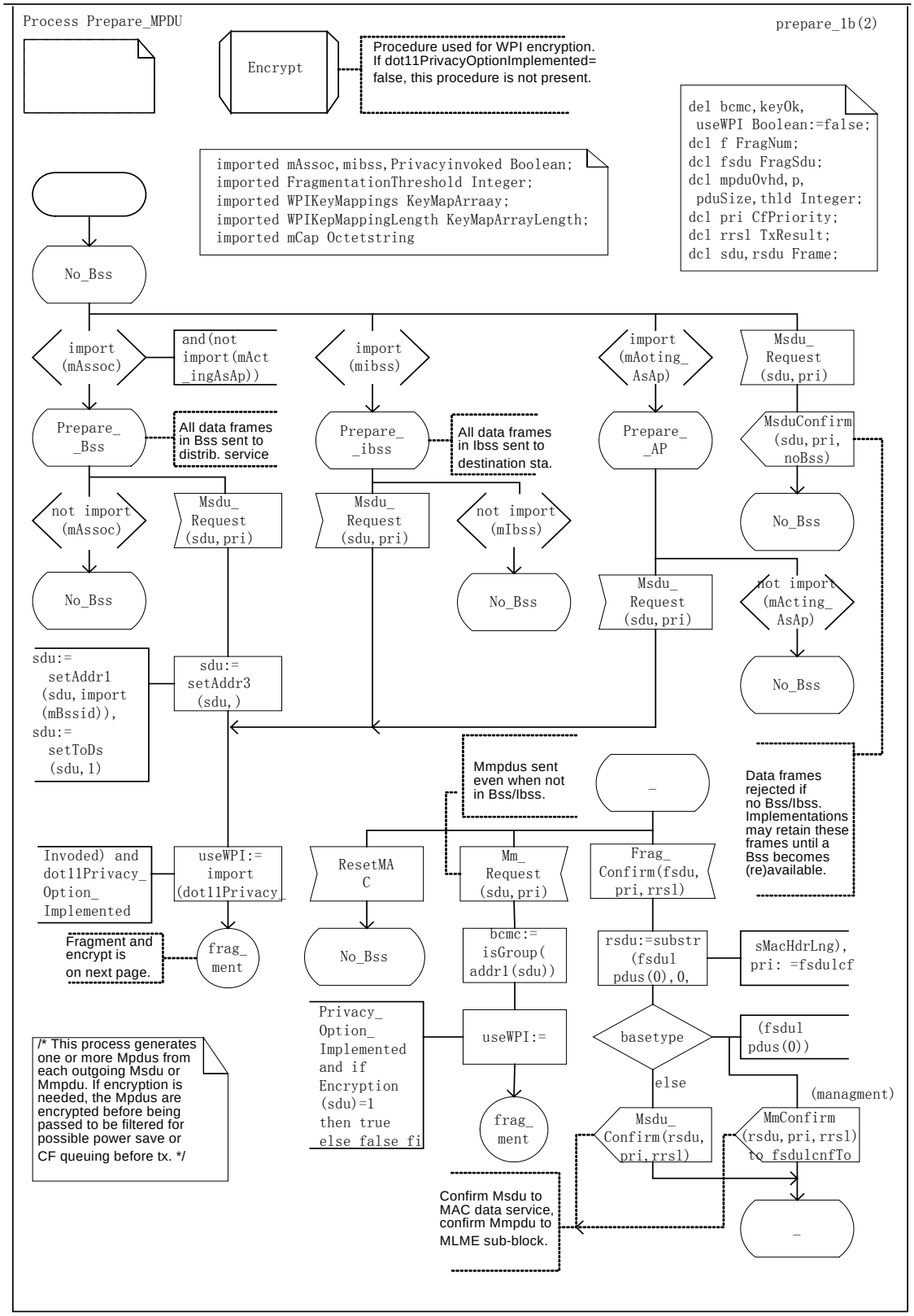
***Replace the diagram on Page 528 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

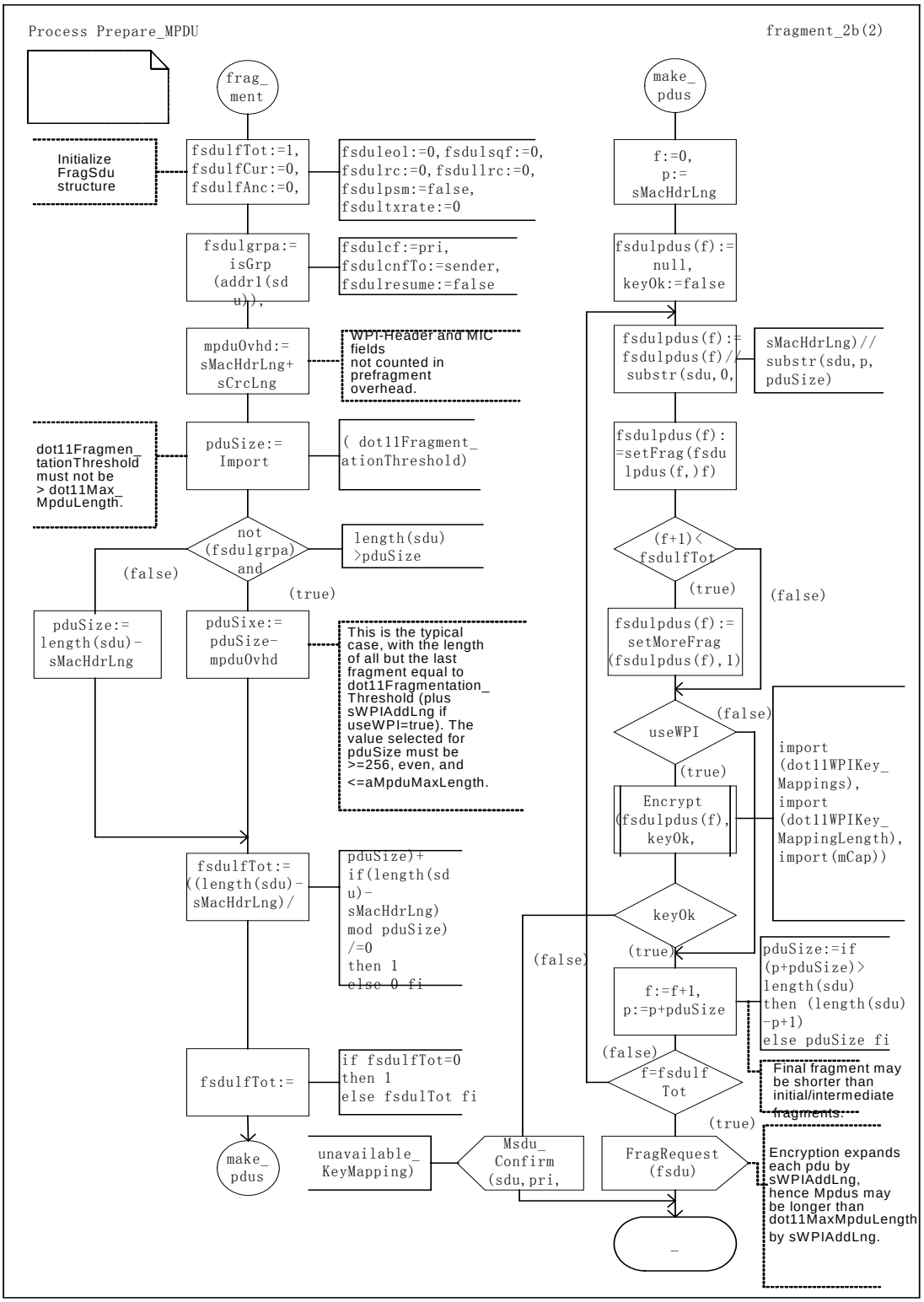
***Replace the diagram on Page 536 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

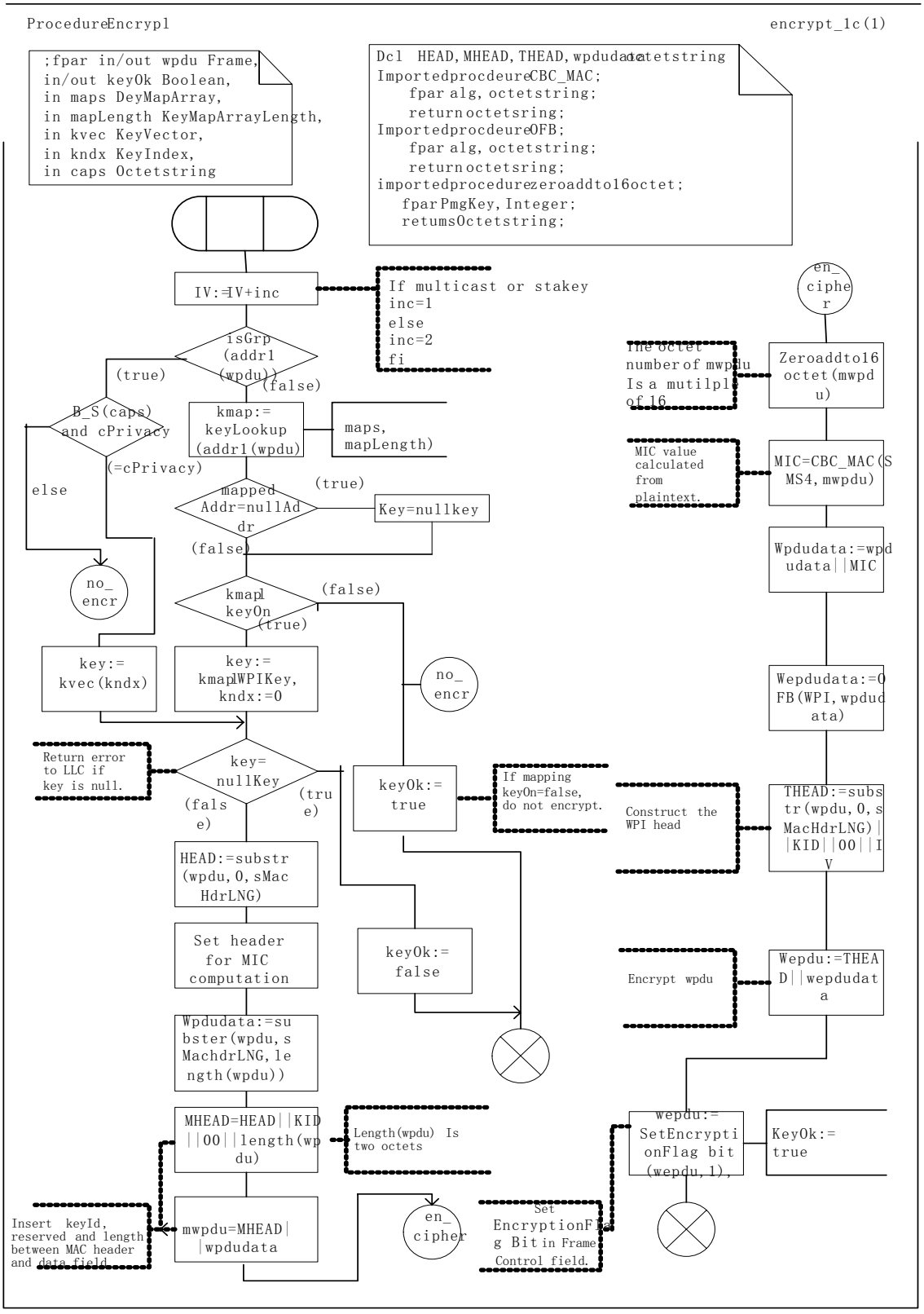
***Replace the diagram on Page 537 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

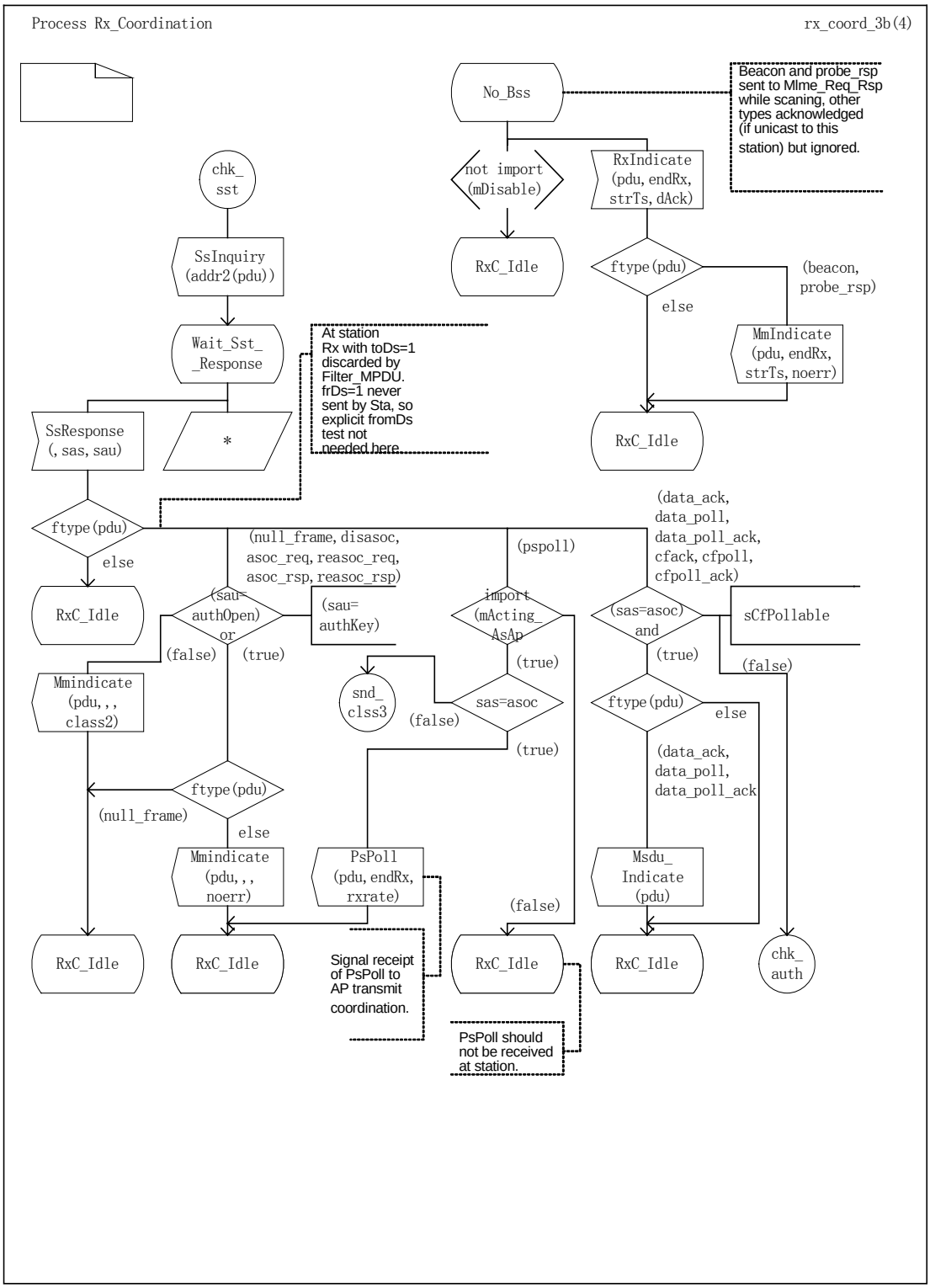
***Replace the diagram on Page 538 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

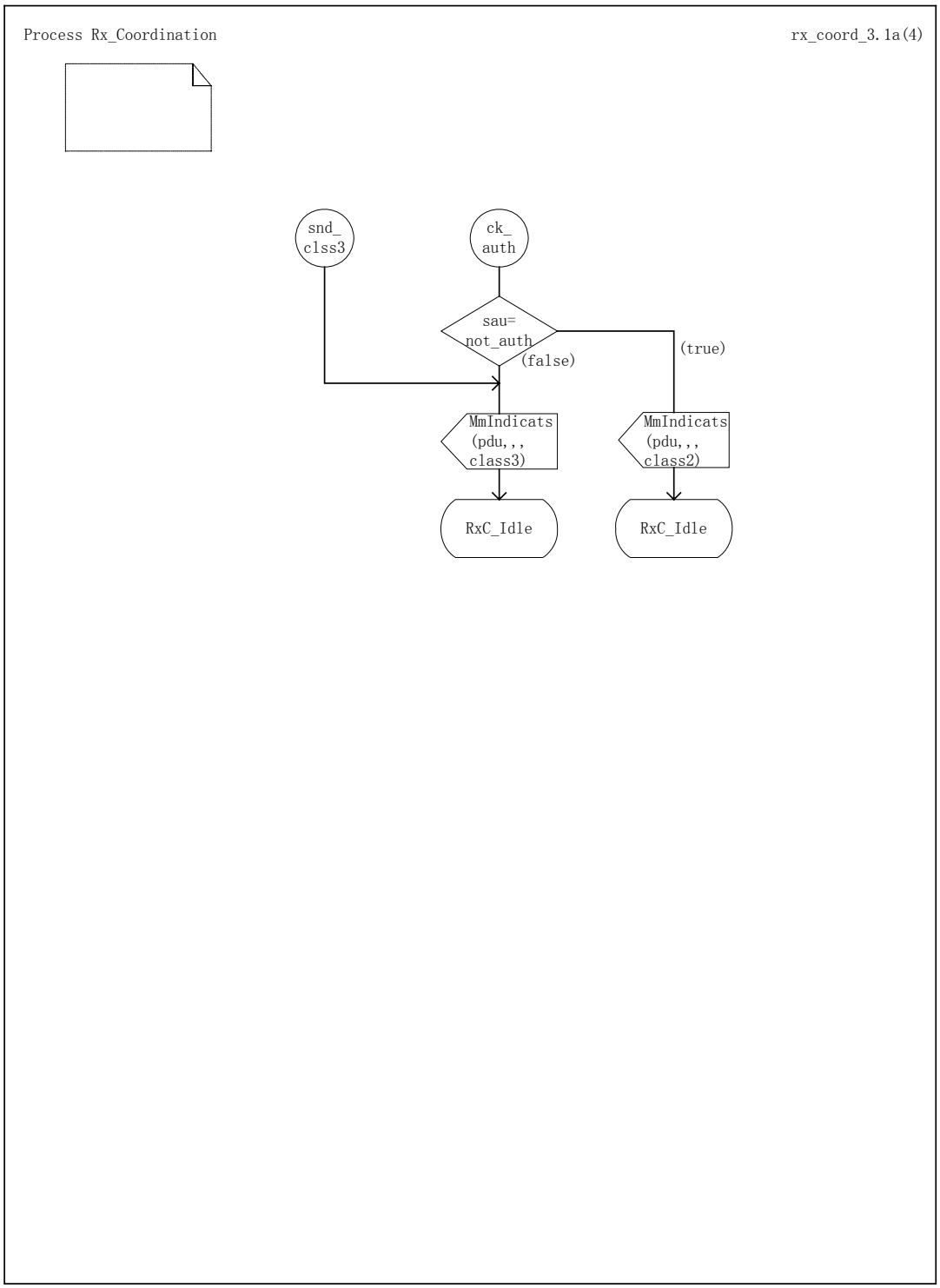
***Replace the diagram on Page 546 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

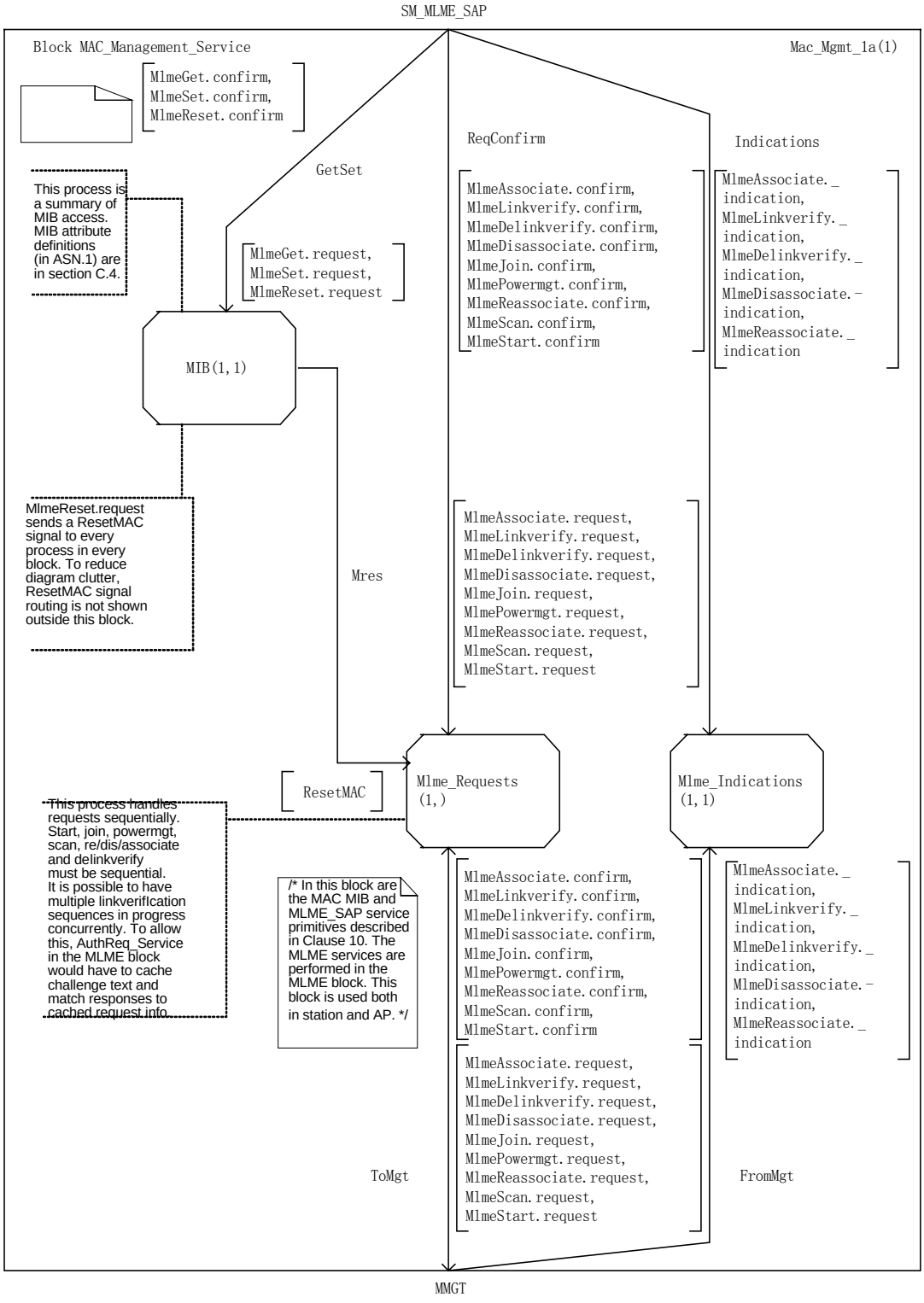
***Replace the diagram on Page 547 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

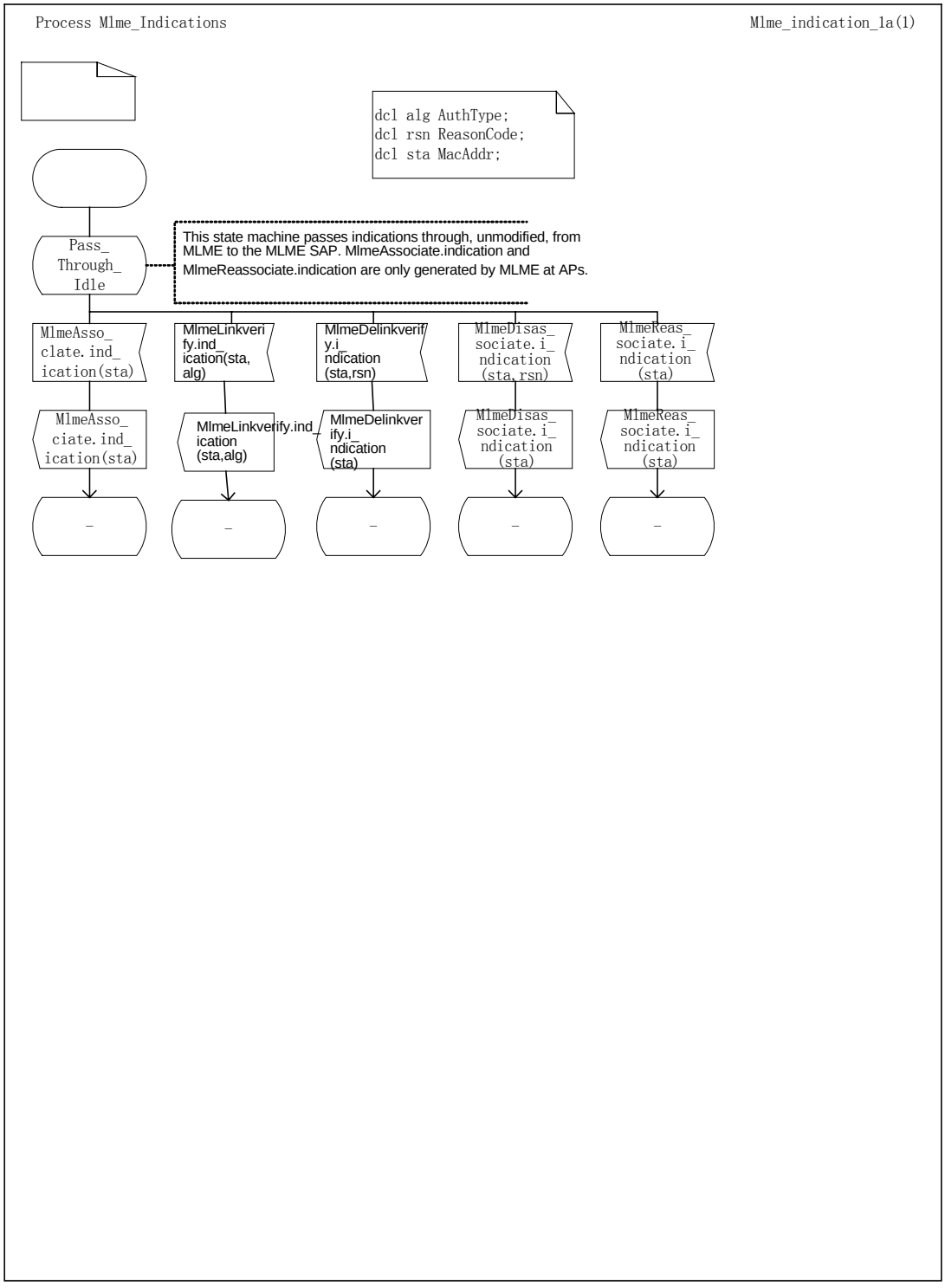
***Replace the diagram on Page 561 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

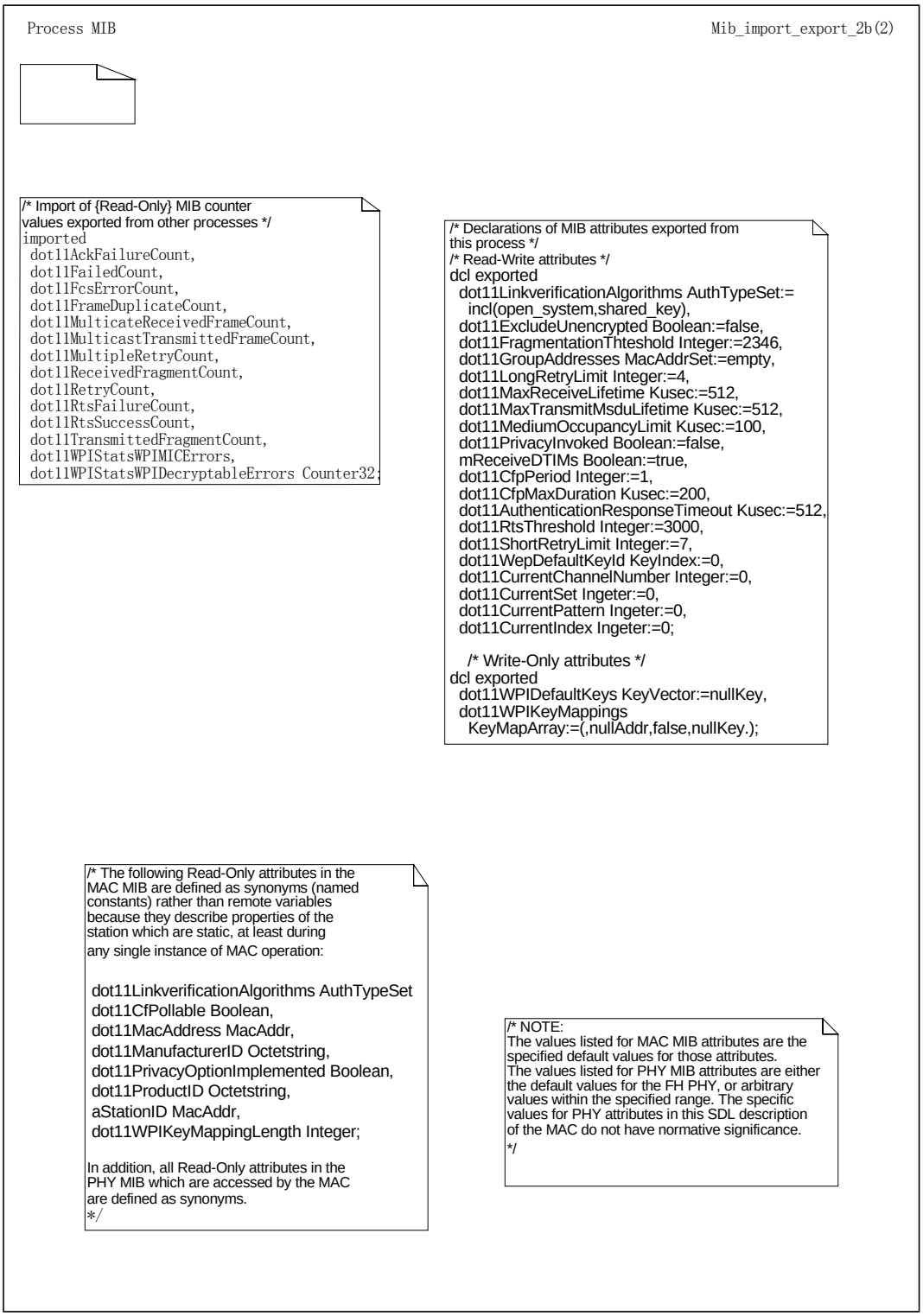
***Replace the diagram on Page 562 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

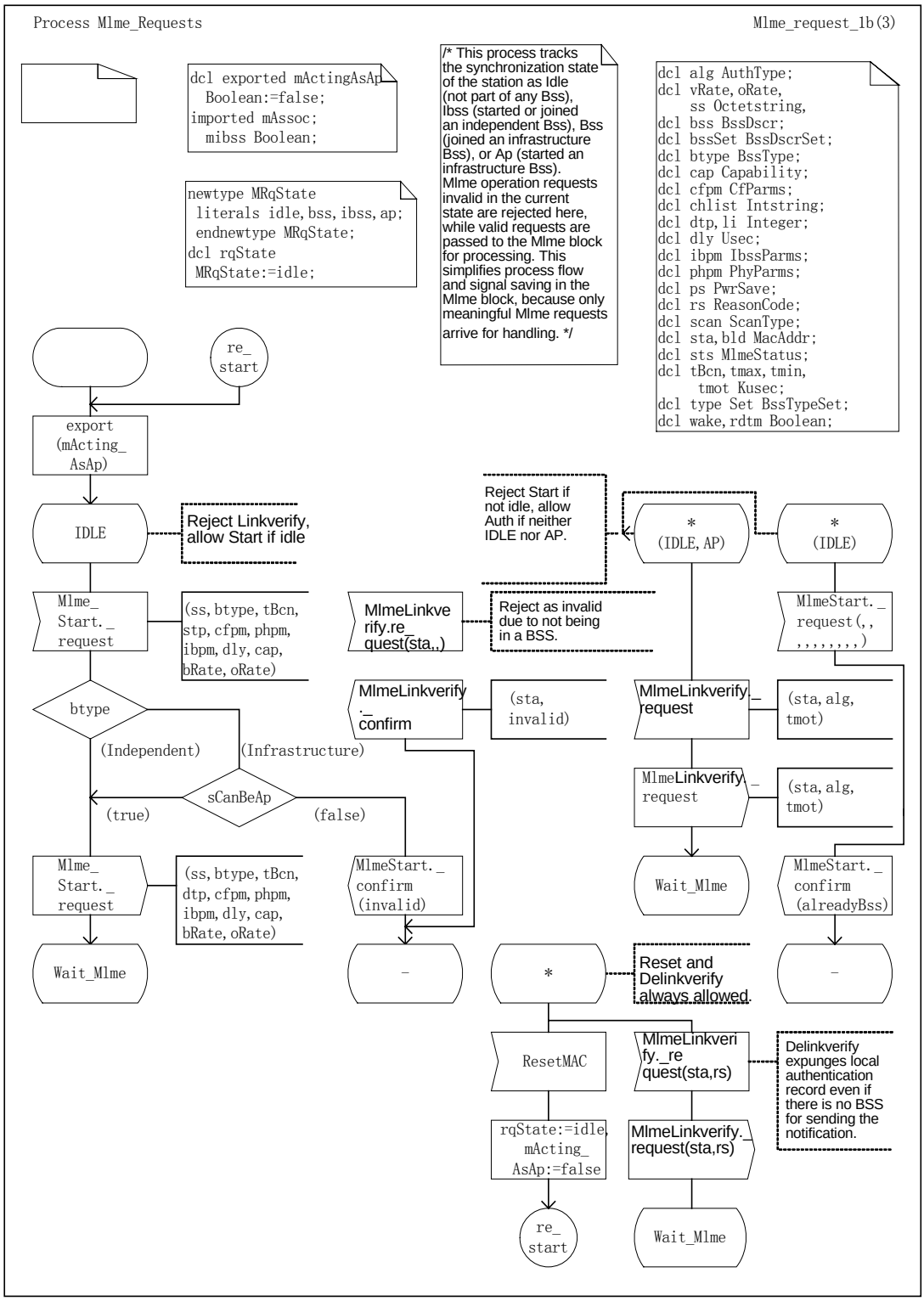
***Replace the diagram on Page 564 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

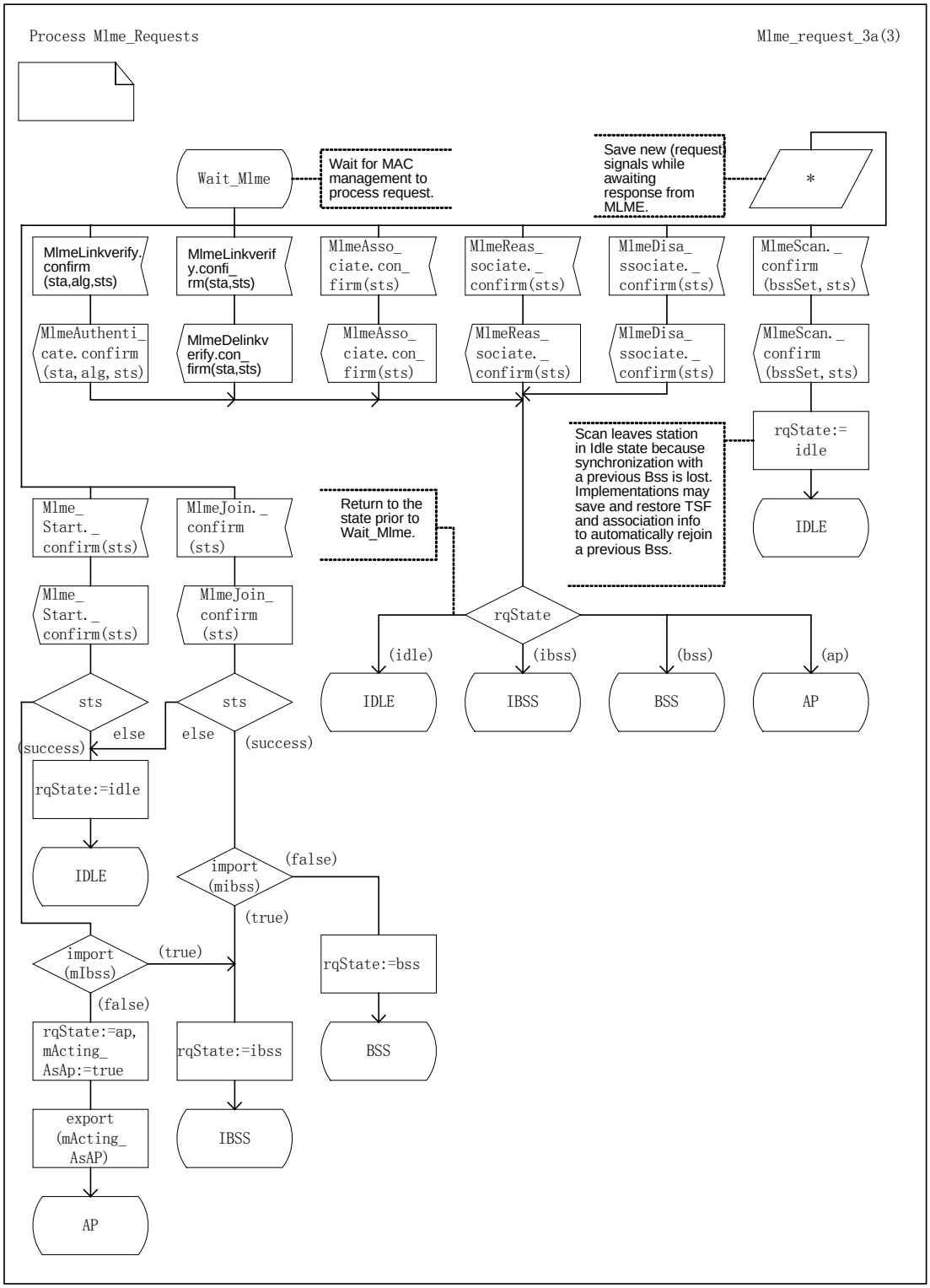
***Replace the diagram on Page 565 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

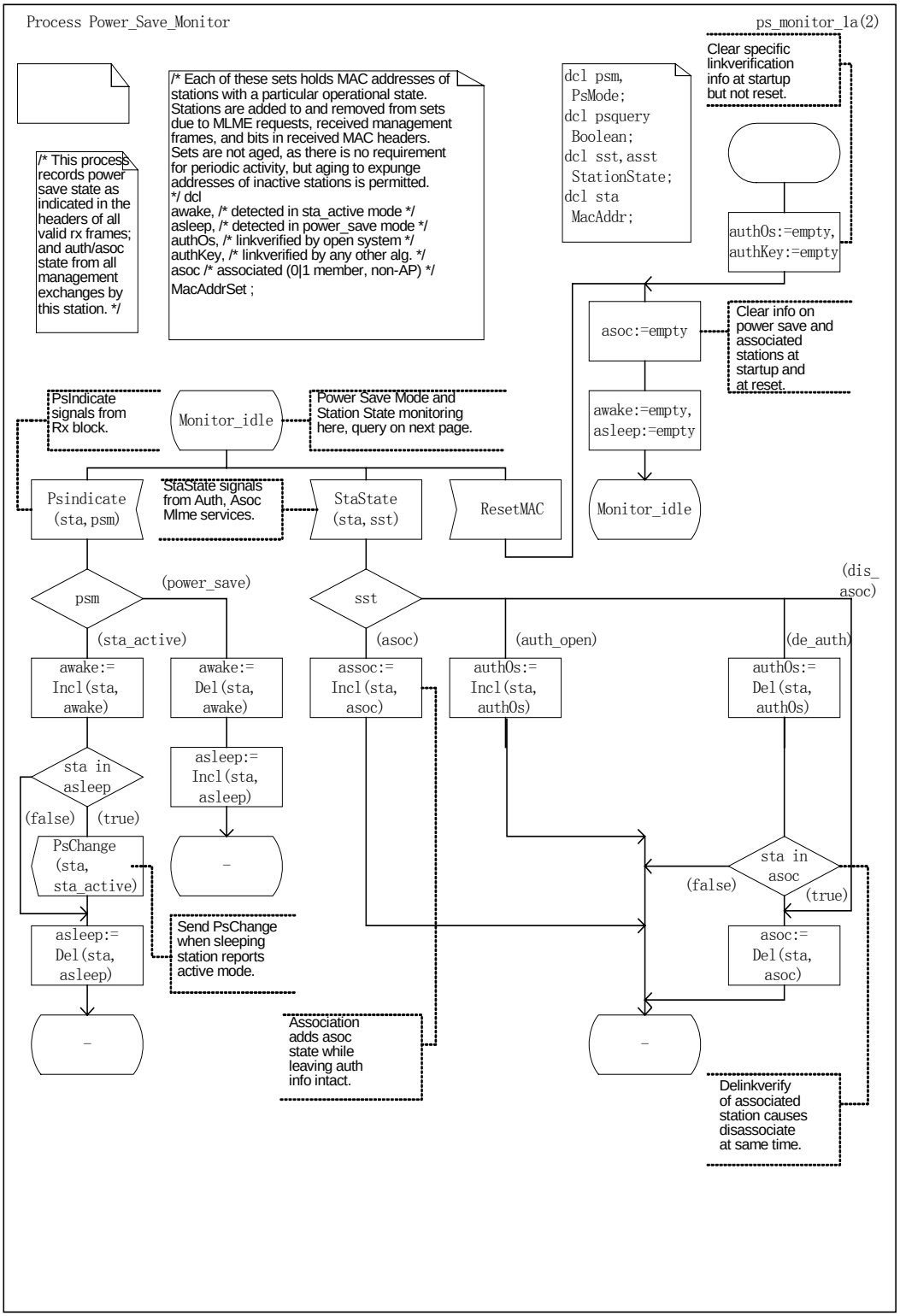
***Replace the diagram on Page 567 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

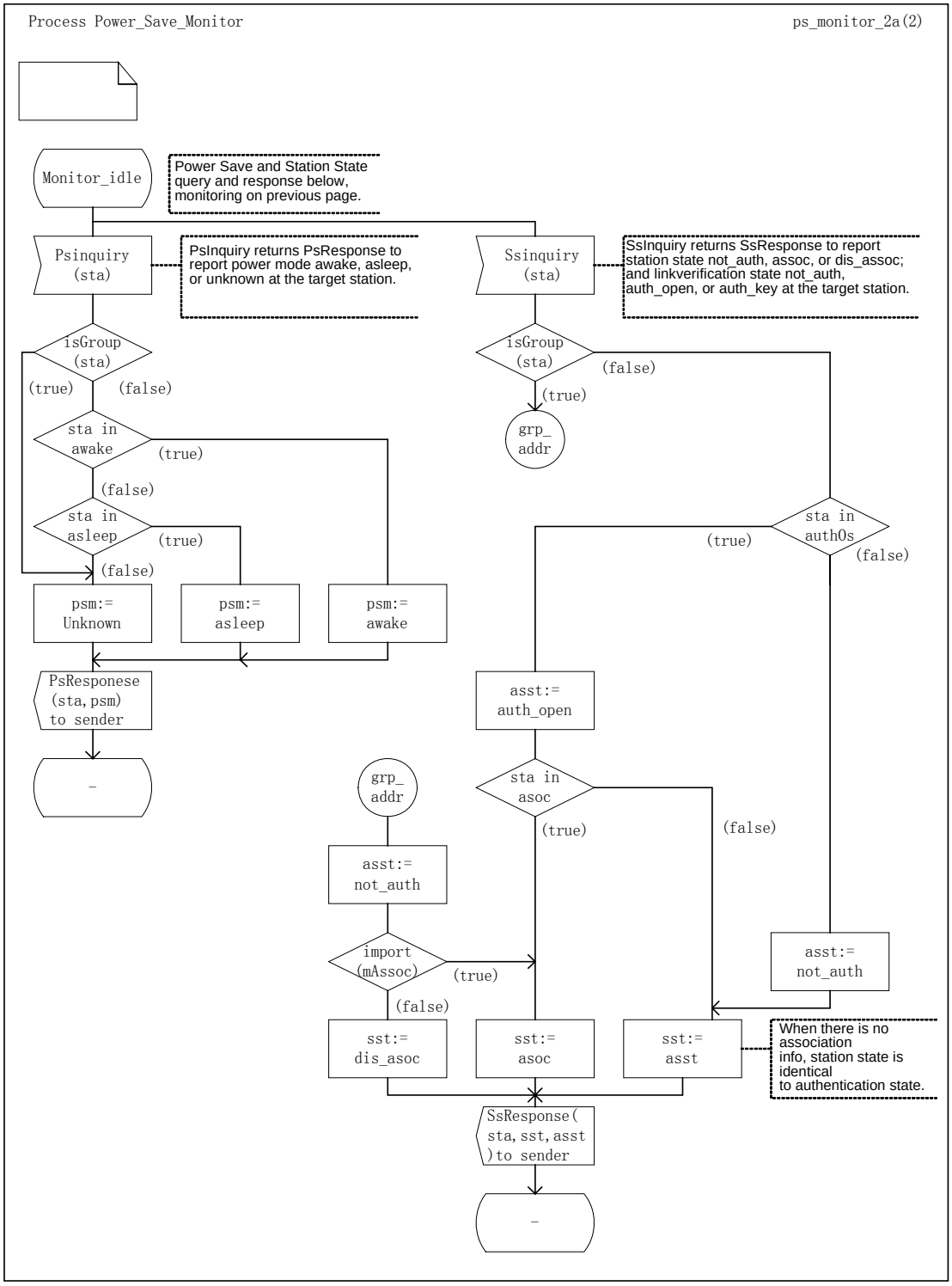
***Replace the diagram on Page 569 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

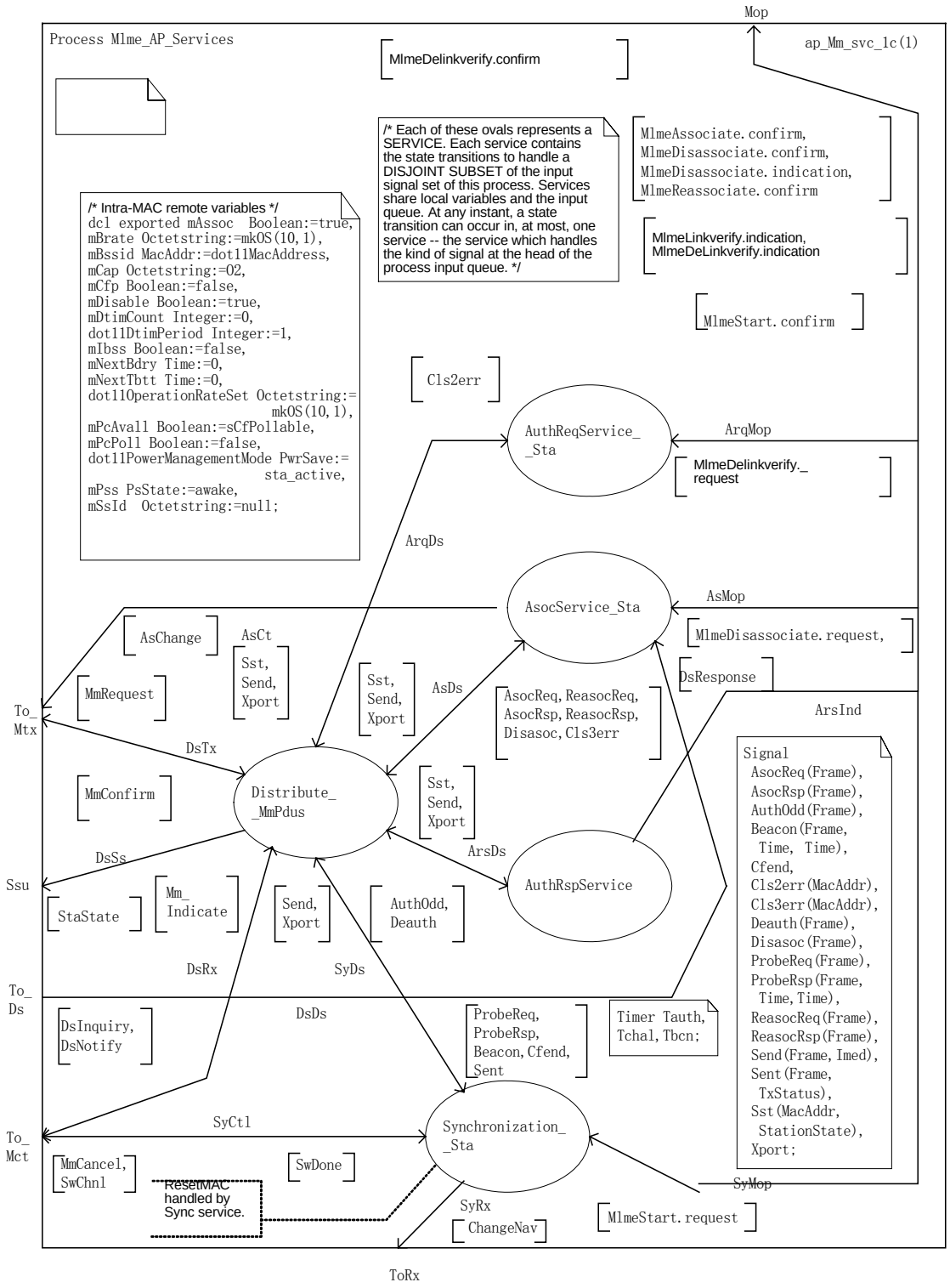
***Replace the diagram on Page 570 as follow:***






--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

***Replace the diagram on Page 571 as follow:***

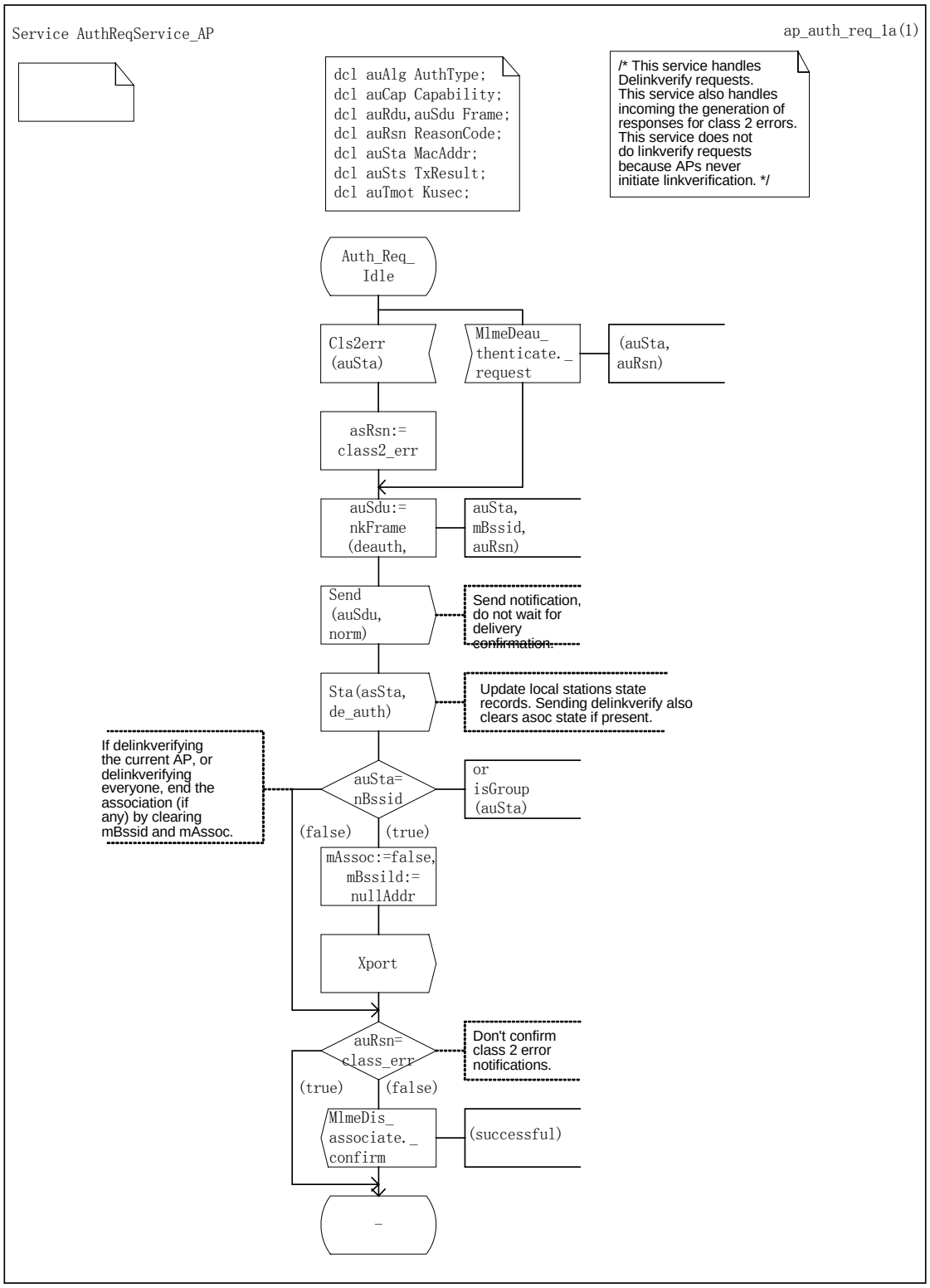





--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

***Replace the diagram on Page 574 as follow:***

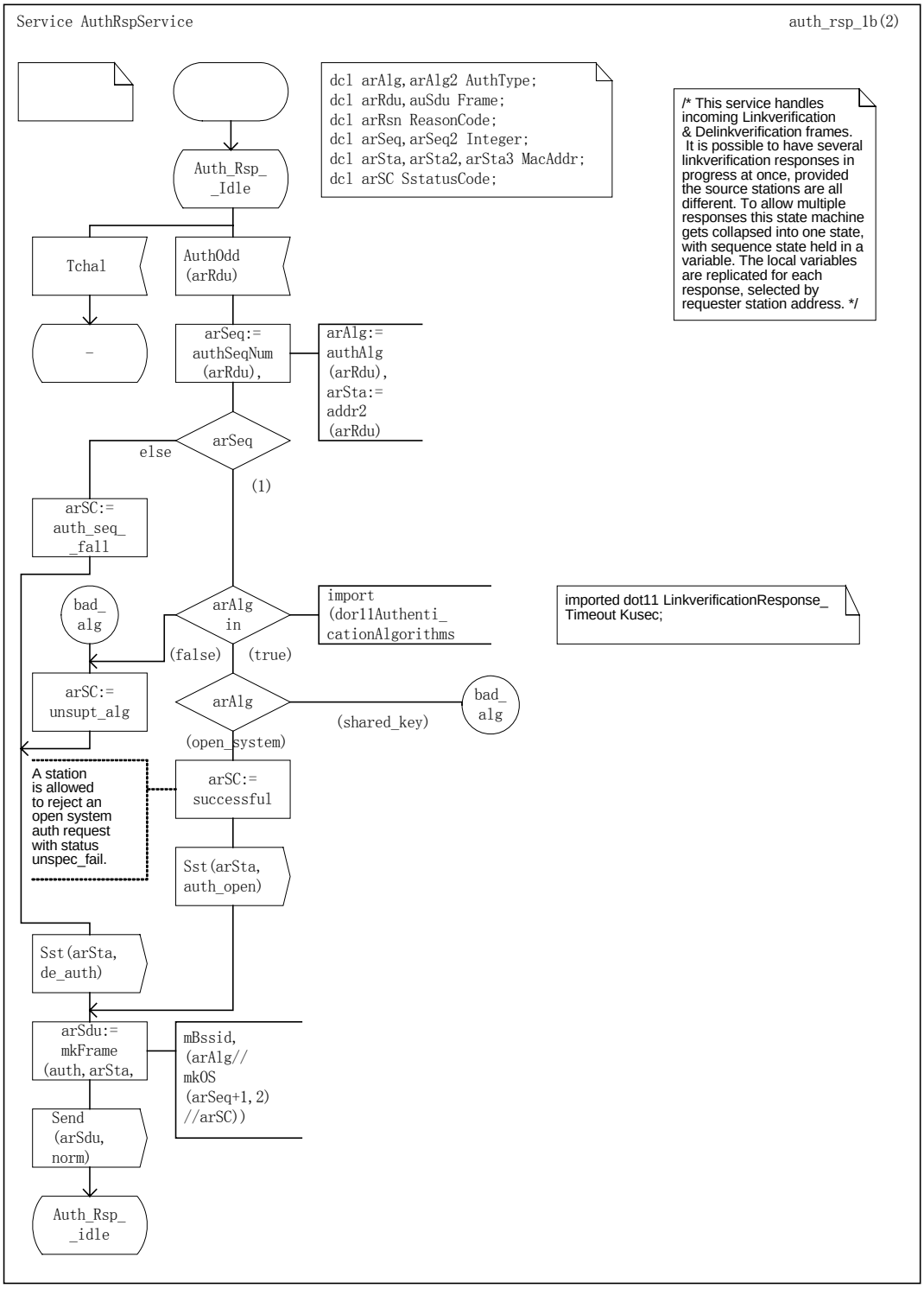





--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

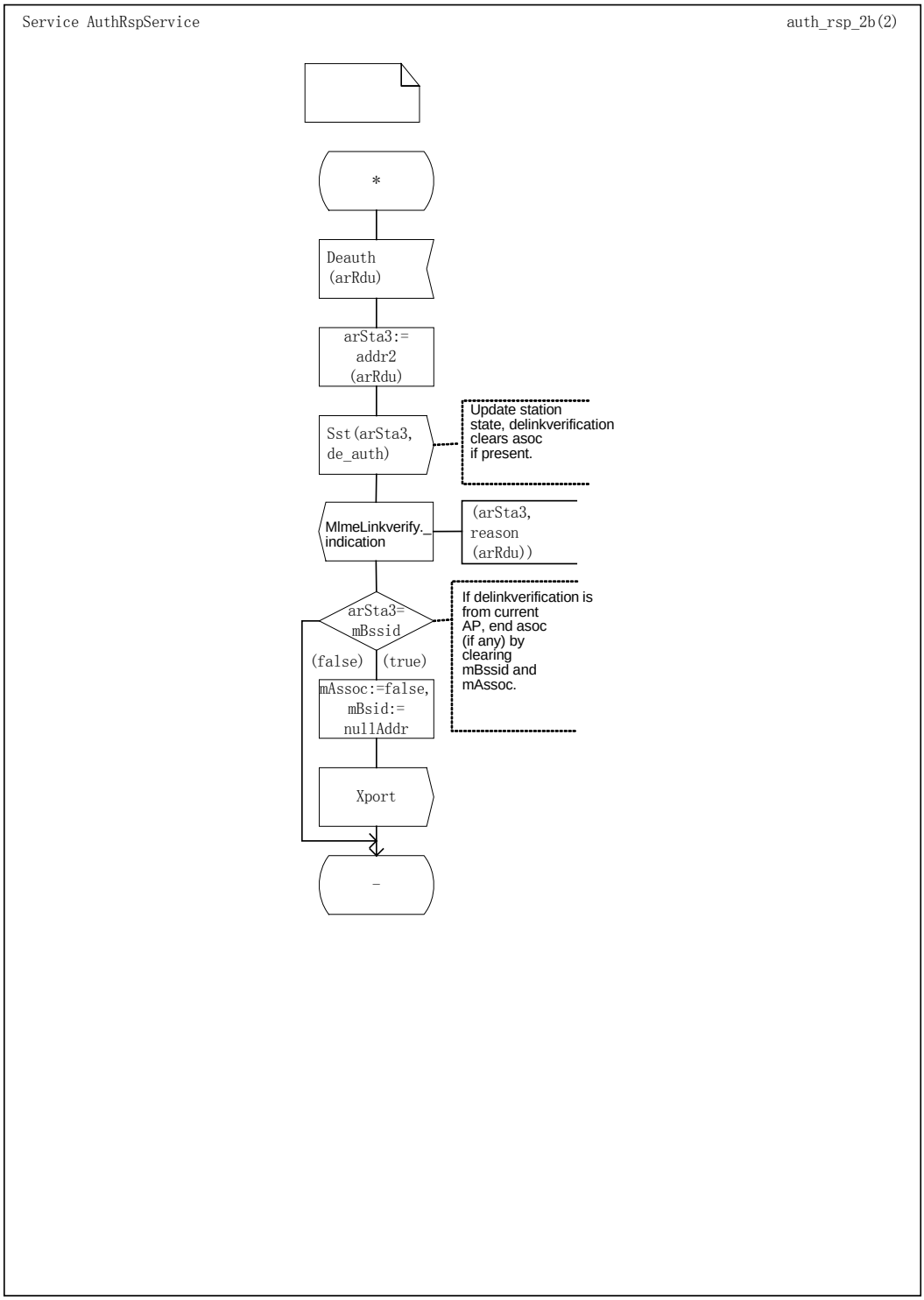
***Replace the diagram on Page 575 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

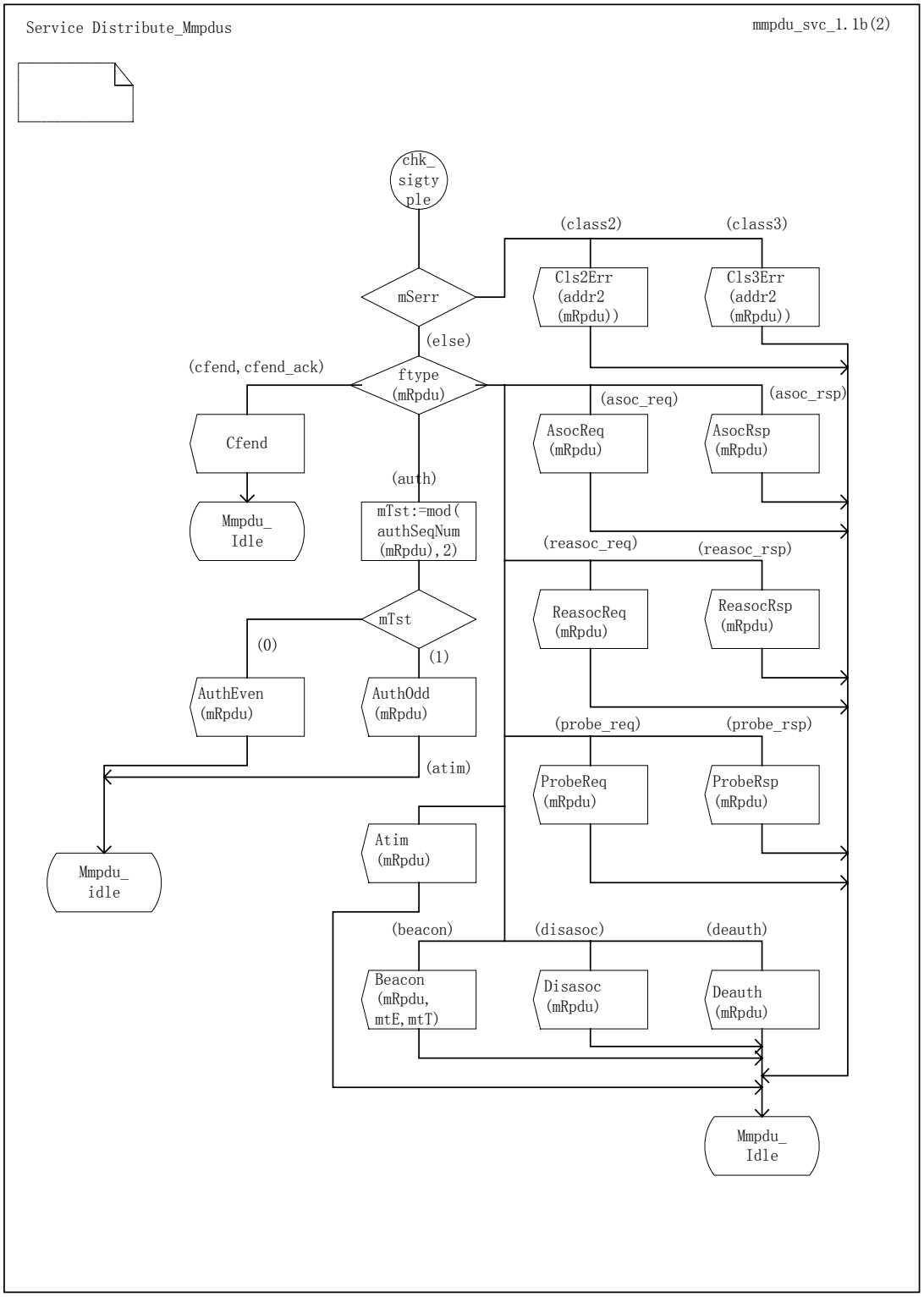
***Replace the diagram on Page 576 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

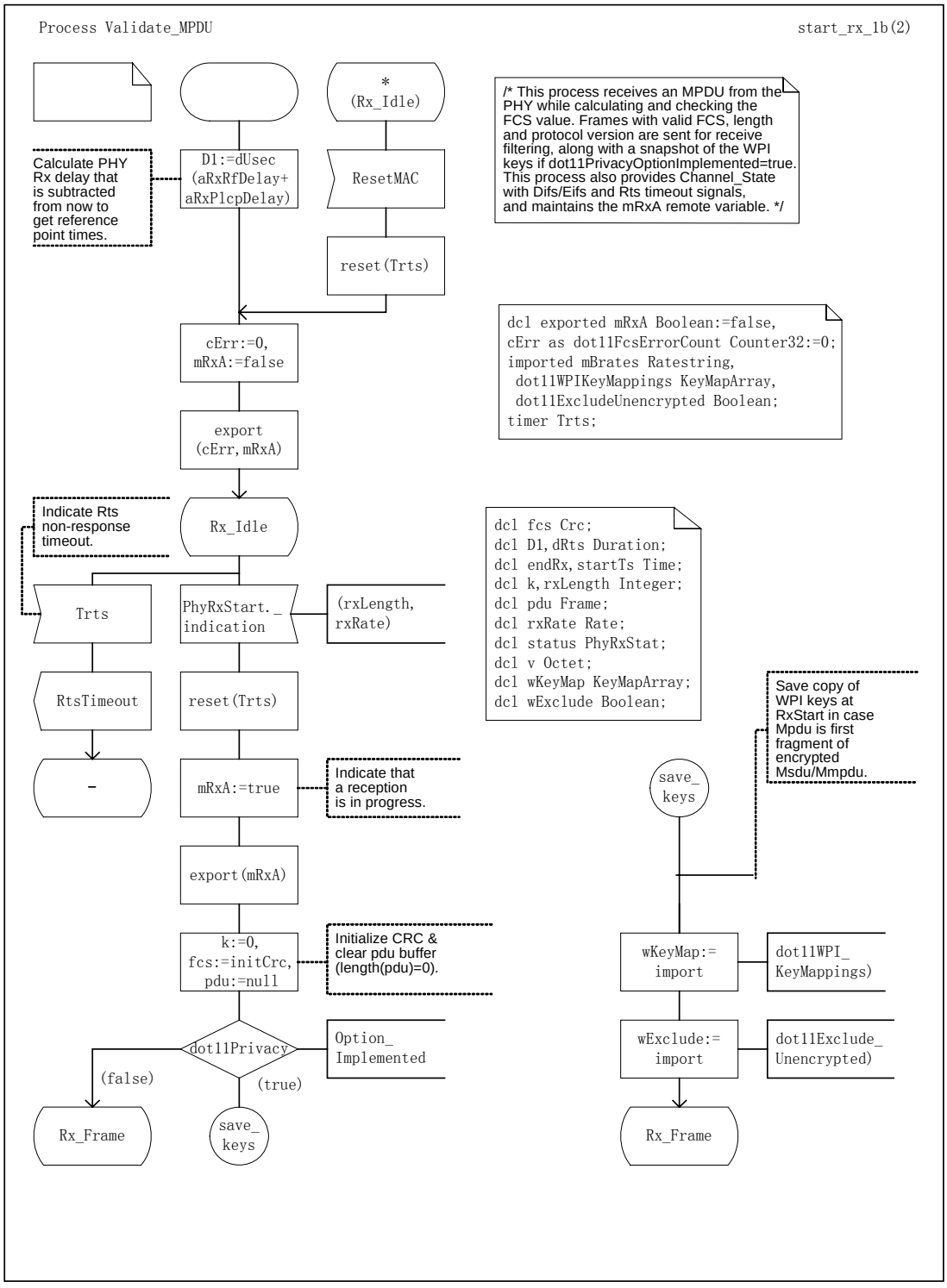
***Replace the diagram on Page 578 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

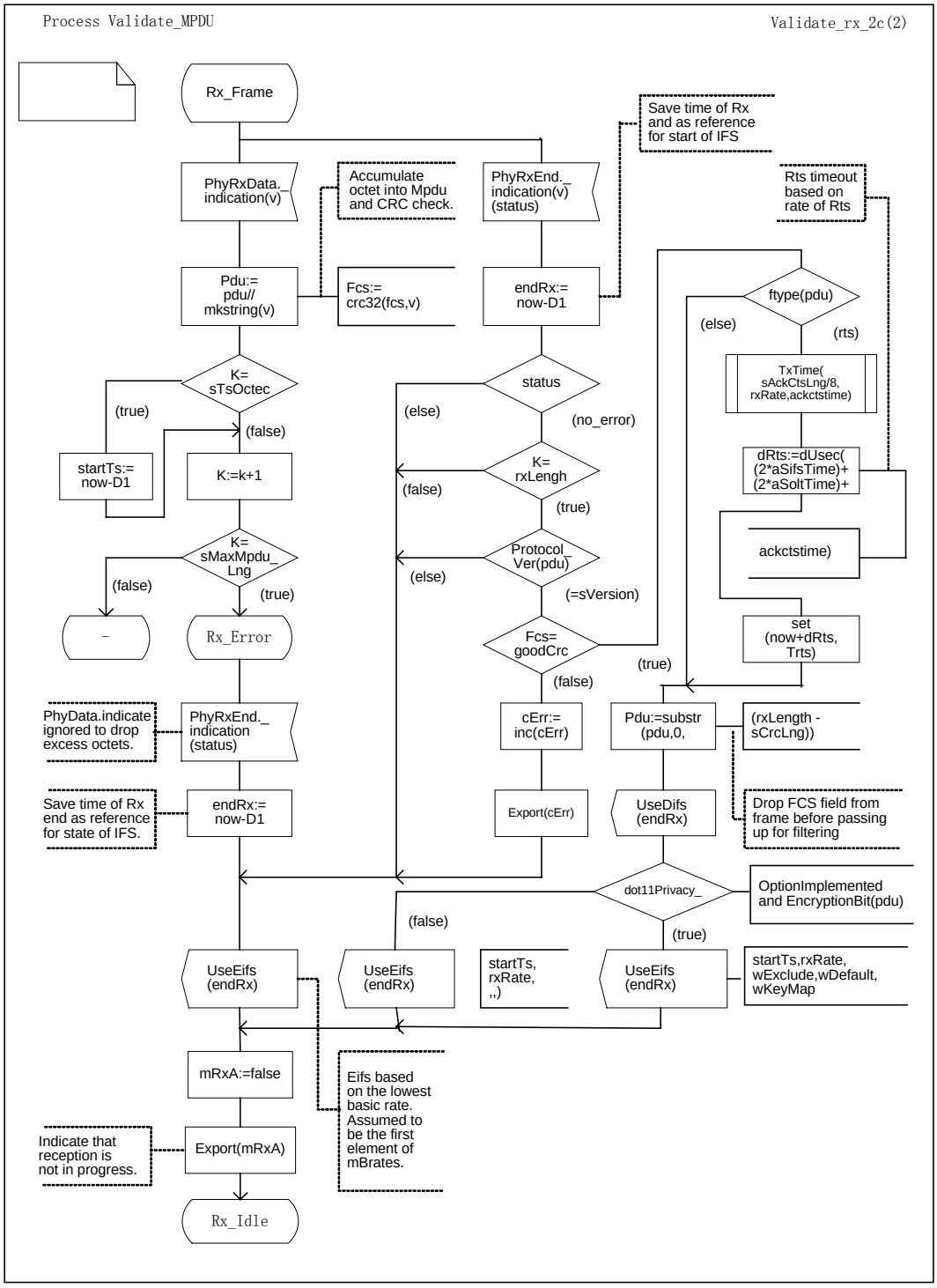
***Replace the diagram on Page 585 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

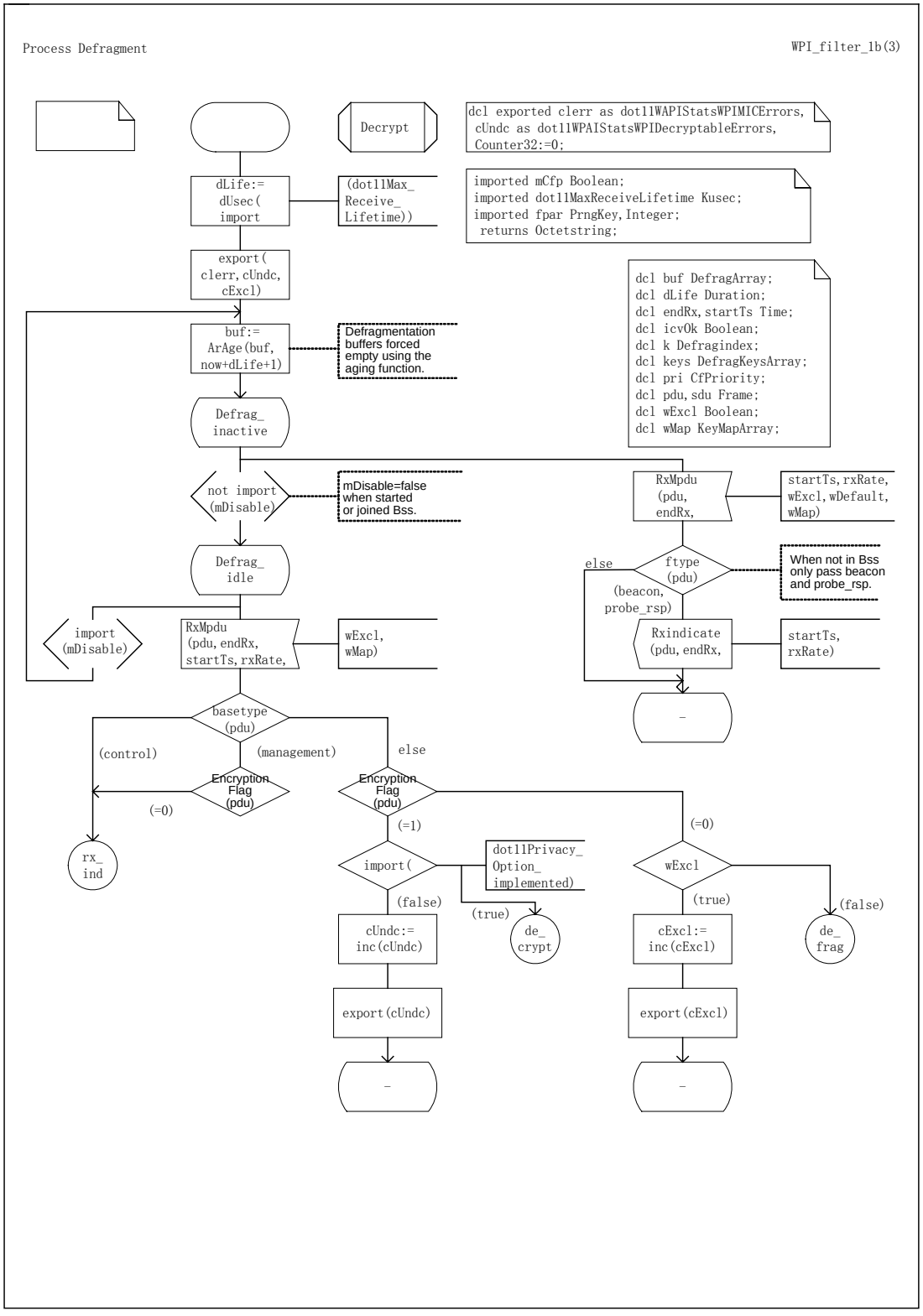
***Change the diagram on Page 586 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

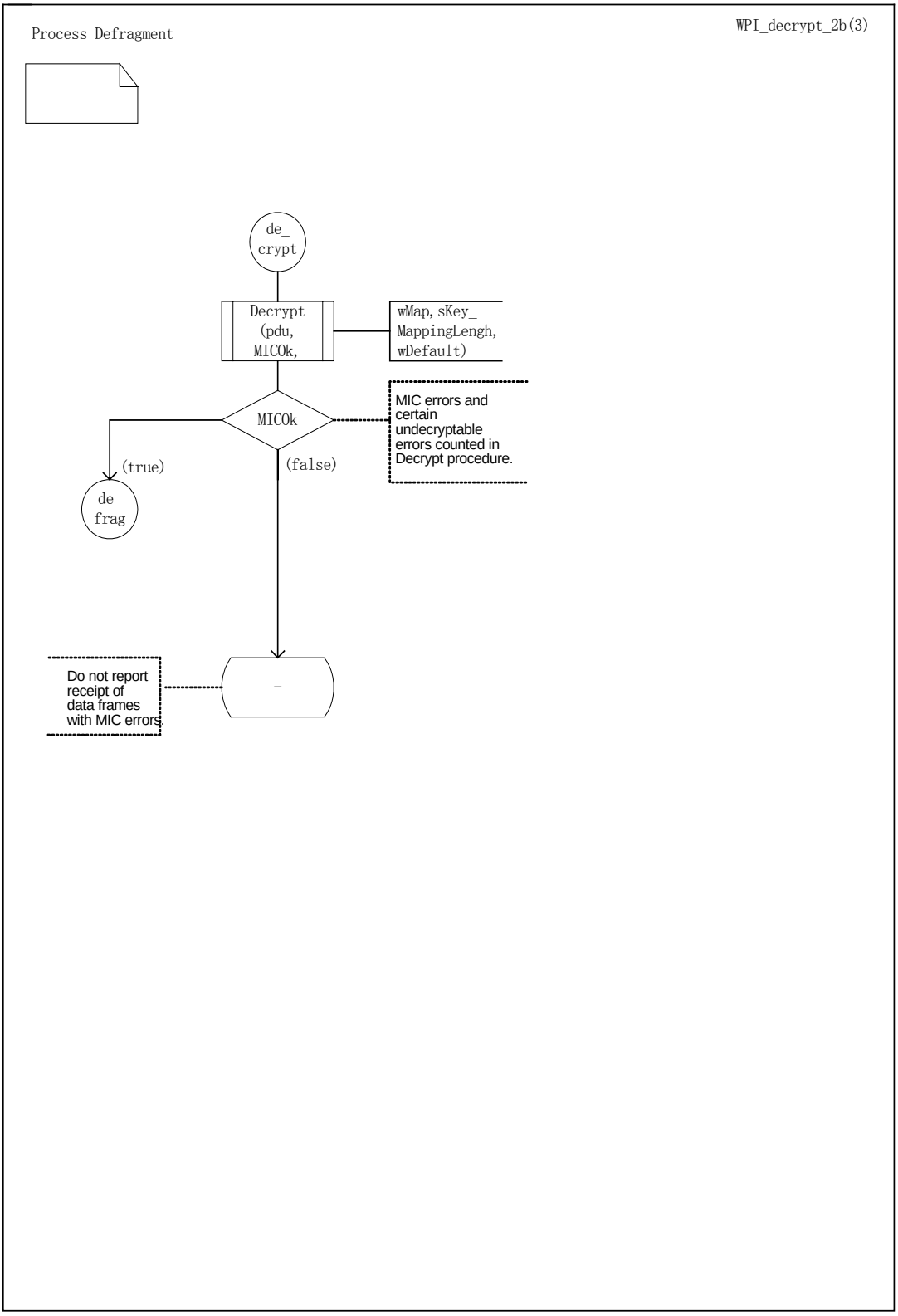
***Replace the diagram on Page 591 as follow:***






--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

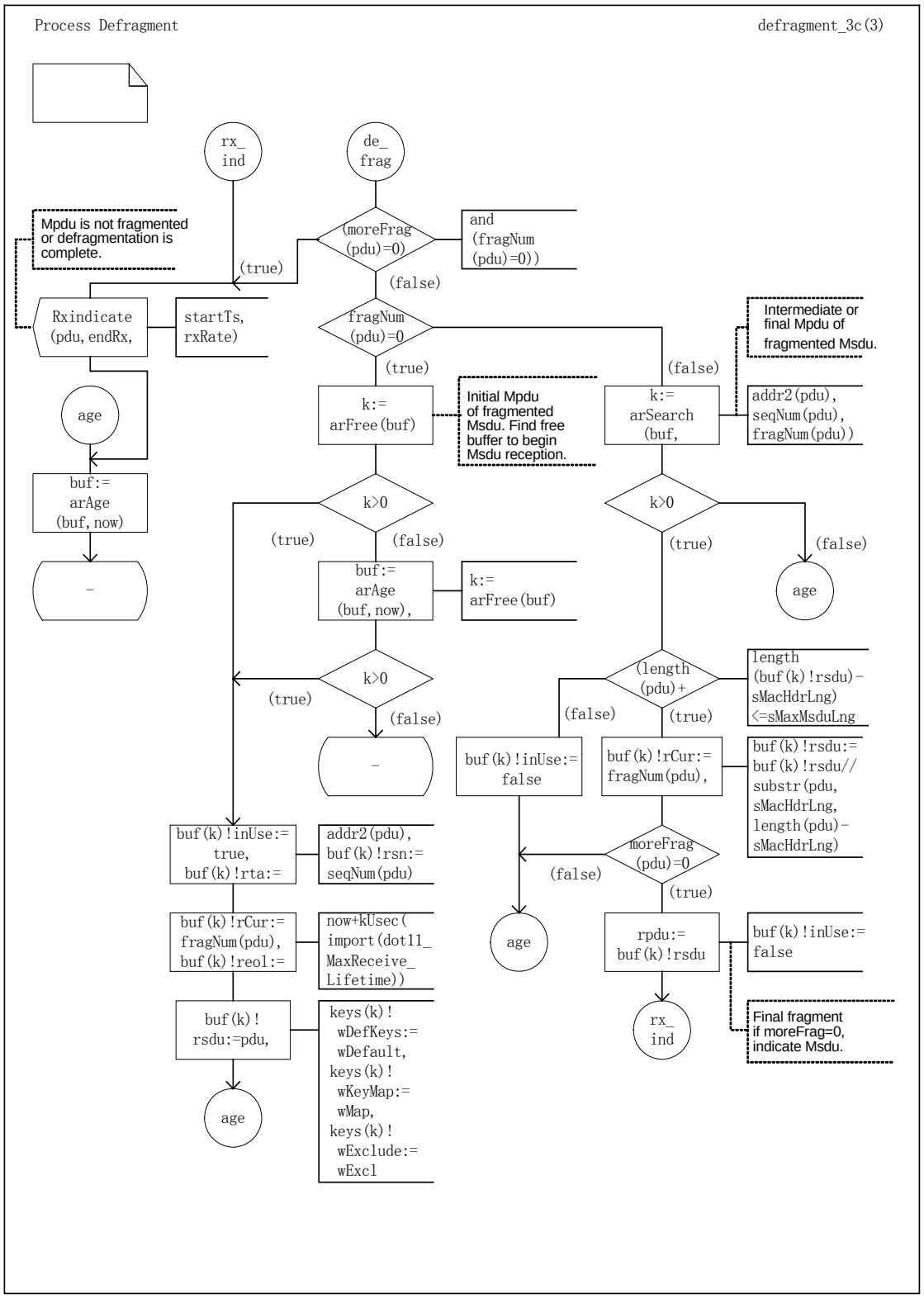
***Replace the diagram on Page 592 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

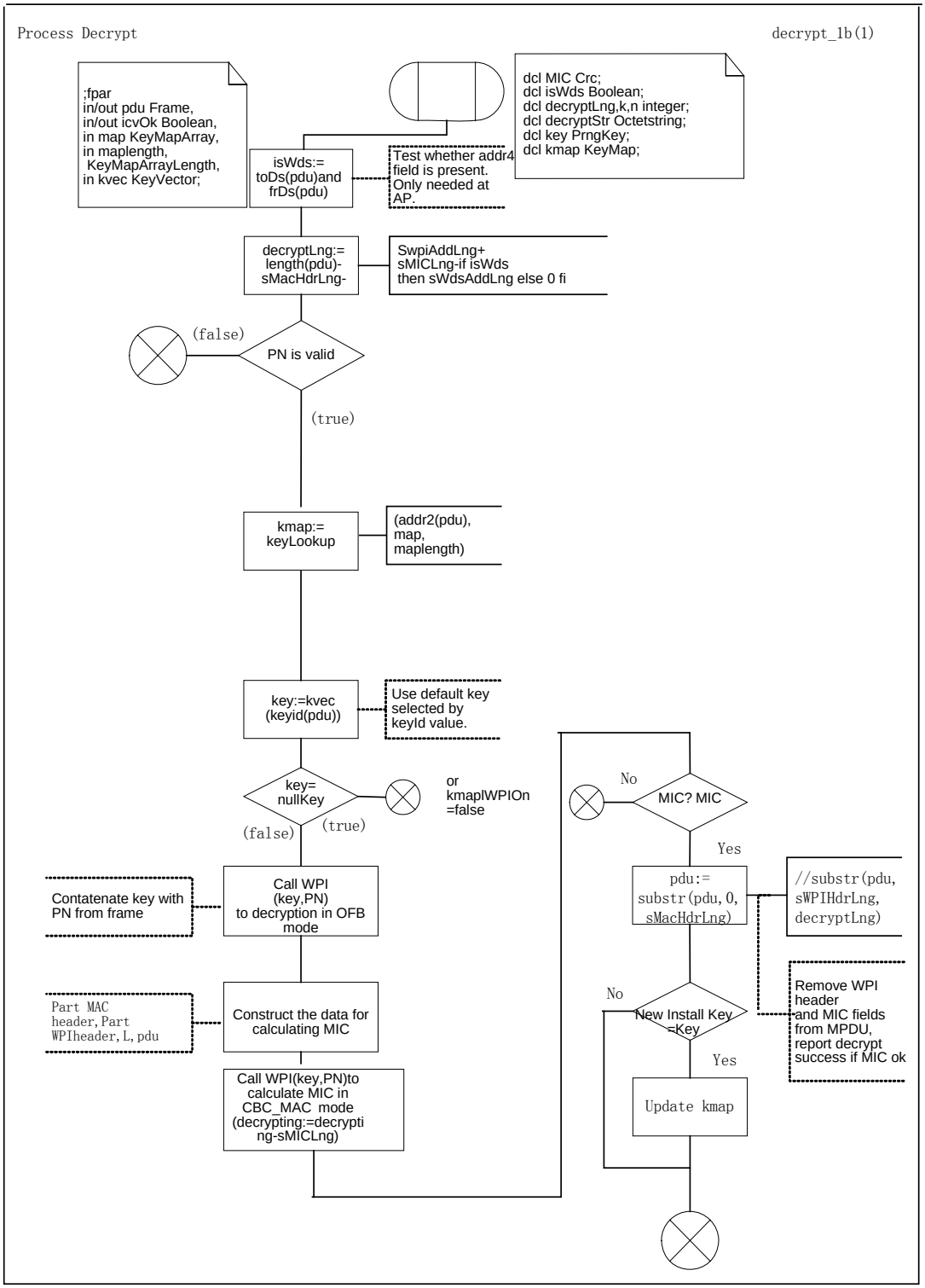
***Replace the diagram on Page 593 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

***Replace the diagram on Page 594 as follow:***






Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

# Annex D

(normative)

# ASN.1 encoding of the MAC and PHY MIB

***In "Major sections" of Annex D, change the dot11smt object identifier list as follows:***

```
-- dot11smt GROUPS
-- dot11StationConfigTable                                  ::= { dot11smt 1 }
-- dot11AuthenticationLinkVerificationAlgorithmsTable       ::= { dot11smt 2 }
-- dot11WEPDefaultKeysTable                                 ::= { dot11smt 3 }
-- dot11WEPWPIKeyMappingsTable                              ::= { dot11smt 4 }
-- dot11PrivacyTable                                        ::= { dot11smt 5 }
-- dot11SMTnotification                                     ::= { dot11smt 6 }
-- dot11MultiDomainCapabilityTable                          ::= { dot11smt 7}
-- dot11SpectrumManagementTable                             ::= { dot11smt 8 }
-- dot11WAPIConfigTable                                     ::= { dot11smt 9}
-- dot11WAPIConfigUnicastCiphersTable                       ::= { dot11smt 10}
-- dot11WAPIConfigAuthenticationSuitesTable                 ::= { dot11smt 11}
-- dot11WAPIStatsTable                                      ::= { dot11smt 12 }
```

***In "SMT Station Config Table" in Annex D, change Dot11StationConfigEntry as follows:***

```
Dot11StationConfigEntry ::=
    SEQUENCE {
        dot11StationID                                  MacAddress,
        dot11MediumOccupancyLimit                       INTEGER,
        dot11CFPollable                                 TruthValue,
        dot11CFPPeriod                                  INTEGER,
        dot11CFPMaxDuration                             INTEGER,
        dot11AuthenticationLinkVerificationResponseTimeOut  Unsigned32,
        dot11PrivacyOptionImplemented                   TruthValue,
        dot11PowerManagementMode                        INTEGER,
        dot11DesiredSSID                                OCTET STRING,
        dot11DesiredBSSType                             INTEGER,
        dot11OperationalRateSet                         OCTET STRING,
        dot11BeaconPeriod                               INTEGER,
        dot11DTIMPeriod                                 INTEGER,
        dot11AssociationResponseTimeOut                 Unsigned32,
```



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

| | |
|---|---|
| dot11DisassociateReason | INTEGER, |
| dot11DisassociateStation | MacAddress, |
| dot11~~Deauthenticate~~DeLinkVerifyReason | INTEGER, |
| dot11~~Deauthenticate~~DeLinkVerifyStation | MacAddress, |
| dot11~~Authenticate~~LinkVerifyFailStatus | INTEGER, |
| dot11~~Authenticate~~LinkVerifyFailStation | MacAddress, |
| dot11MultiDomainCapabilityImplemented | TruthValue, |
| dot11MultiDomainCapabilityEnabled | TruthValue, |
| dot11CountryString | OCTET STRING, |
| dot11SpectrumManagementImplemented | TruthValue, |
| dot11SpectrumManagementRequired | TruthValue, |
| dot11WAPIOptionImplemented | TruthValue, |
| dot11WAPIPreauthenticationImplemented | TruthValue } |

***In "SMT Station Config Table" in Annex D, change dot11AuthenticationResponseTimeOut as follows:***

dot11~~Authentication~~LinkVerificationResponseTimeOut OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"This attribute shall specify the number of time units (TUs) that a responding STA should wait for the next frame in the ~~authentication~~LinkVerification sequence."
::= { dot11StationConfigEntry 6 }

***In "SMT Station Config Table" in Annex D, change dot11PrivacyOptionImplemented as follows:***

dot11PrivacyOptionImplemented OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This attribute, when true, shall indicate that the ~~IEEE 802.11 WEP~~WPI option is implemented. The default value of this attribute shall be false."
::= { dot11StationConfigEntry 7 }

***In "SMT Station Config Table" in Annex D, change dot11DeauthenticateReason, dot11Deauthenticate-Station, dot11AuthenticateFailStatus and dot11AuthenticateFailStation as follows:***

dot11~~Deauthenticate~~DeLinkVerifyReason OBJECT-TYPE
SYNTAX INTEGER(0..65535)
MAX-ACCESS read-only
STATUS current



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

DESCRIPTION

"This attribute holds the most recently transmitted Reason Code in a ~~Deauthentication~~ DeLinkVerification frame. If no ~~Deauthentication~~ DeLinkVerification frame has been transmitted, the value of this attribute shall be 0."

REFERENCE "~~IEEE Std 802.11, 1999~~ISO/IEC 8802-11, 2005 Edition, 7.3.1.7"

::= { dot11StationConfigEntry 17 }

dot11~~Deauthenticate~~DeLinkVerifyStation OBJECT-TYPE

SYNTAX MacAddress

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This attribute holds the MAC address from the Address 1 field of the most recently transmitted ~~Deauthentication~~ DeLinkVerification frame. If no ~~Deauthentication~~ DeLinkVerification frame has been transmitted, the value of this attribute shall be 0."

::= { dot11StationConfigEntry 18 }

dot11~~Authenticate~~LinkVerifyFailStatus OBJECT-TYPE

SYNTAX INTEGER(0..65535)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This attribute holds the most recently transmitted Status Code in a failed ~~Authentication~~LinkVerification frame. If no failed ~~Authentication~~LinkVerification frame has been transmitted, the value of this attribute shall be 0."

REFERENCE "~~IEEE Std 802.11, 1999~~ISO/IEC 8802-11, 2005 Edition, 7.3.1.9"

::= { dot11StationConfigEntry 19 }

dot11~~Authenticate~~LinkVerifyFailStatus OBJECT-TYPE

SYNTAX MacAddress

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This attribute holds the MAC address from the Address 1 field of the most recently transmitted failed Authen~~tication~~LinkVerification frame. If no failed ~~Authentication~~LinkVerification frame has been transmitted, the value of this attribute shall be 0."

::= { dot11StationConfigEntry 20 }

***In "SMT Station Config Table" in Annex D, insert the following attributes after dot11Spectrum-ManagementRequired { dot11StationConfigEntry 25 }:***

dot11WAPIOptionImplemented OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

" This variable indicates whether the entity is WAPI-capable."
::= { dot11StationConfigEntry 26 }

dot11WAPIPreauthenticationImplemented OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
" This variable indicates whether the entity supports WAPI pre-authentication. This cannot be TRUE unless dot11WAPIOptionImplemented is TRUE."
::= { dot11StationConfigEntry 27 }

***Change "AuthenticationAlgorithms TABLE" as follows:***

-- ************************************************************************
-- * ~~Authentication~~LinkVerificationAlgorithms TABLE
-- ************************************************************************

dot11~~Authentication~~LinkVerificationAlgorithmsTable OBJECT-TYPE
SYNTAX SEQUENCE OF Dot11~~Authentication~~LinkVerificationAlgorithmsEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"This (conceptual) table of attributes shall be a set of all the authentication algorithms supported by the stations. The following are the default values and the associated algorithm:
Value = 1: Open System
~~Value = 2: Shared Key~~"
REFERENCE "~~IEEE Std 802.11, 1999~~ISO/IEC 8802-11, 2005 Edition, 7.3.1.1"
::= { dot11smt 2 }
dot11~~Authentication~~LinkVerification AlgorithmsEntry OBJECT-TYPE
SYNTAX Dot11~~Authentication~~LinkVerification AlgorithmsEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"An Entry (conceptual row) in the ~~Authentication~~LinkVerification Algorithms Table.
ifIndex - Each ~~IEEE 802.11~~ISO/IEC 8802-11 interface is represented by an ifEntry. Interface tables in this MIB module are indexed by ifIndex."
INDEX { ifIndex,
dot11~~Authentication~~LinkVerificationAlgorithmsIndex }
::= { dot11~~Authentication~~LinkVerificationAlgorithmsTable 1 }
Dot11~~Authentication~~LinkVerificationAlgorithmsEntry ::=
SEQUENCE { dot11~~Authentication~~LinkVerificationAlgorithmsIndex Integer32,
dot11~~Authentication~~LinkVerificationAlgorithm INTEGER,



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                dot11~~Authentication~~LinkVerificationAlgorithmsEnable  TruthValue  }
        dot11~~Authentication~~LinkVerificationAlgorithmsIndex  OBJECT-TYPE
                SYNTAX Integer32
                MAX-ACCESS not-accessible
                STATUS current
                DESCRIPTION
                        "The auxiliary  variable used to identify instances of the columnar objects in the
                          ~~Authentication~~LinkVerification  Algorithms Table."
::= { dot11~~Authentication~~LinkVerificationAlgorithmsEntry 1 }
        dot11~~Authentication~~LinkVerificationAlgorithm OBJECT-TYPE
                SYNTAX  INTEGER { openSystem(1)~~, sharedKey(2) ~~}
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "This attribute shall be a set of all the  ~~Authentication~~LinkVerification algorithms supported by the
                          STAs. The following are the default values and the associated algorithm.
                          Value = 1: Open System
                          ~~Value = 2: Shared Key"~~
::= { dot11~~Authentication~~LinkVerificationAlgorithmsEntry 2 }
        dot11~~Authentication~~LinkVerificationAlgorithmsEnable  OBJECT-TYPE
                SYNTAX TruthValue
                MAX-ACCESS read-write
                STATUS current
                DESCRIPTION
                        "This attribute, when true at a station, shall enable the acceptance of the
                          ~~Authentication~~LinkVerification  algorithm described in the corresponding table entry in
                          ~~Authentication~~LinkVerification frames received by the station that have odd
                          ~~Authentication~~LinkVerification  sequence numbers. The default value of this attribute shall be 1
                          for the Open System table entry~~ and 2 for all other table entries~~ . "
        ::= { dot11~~Authentication~~LinkVerification AlgorithmsEntry 3 }

-- ************************************************************************
-- * End of ~~Authentication~~LinkVerification Algorithms TABLE
-- ************************************************************************
```

***Delete "WEPDefaultKeys TABLE".***

***Change "WEPKeyMappings" as follows:.***

```
-- ************************************************************************
-- * ~~WEP~~WPIKeyMappings TABLE
-- ************************************************************************
dot11~~WEP~~WPIKeyMappingsTable OBJECT-TYPE
   SYNTAX SEQUENCE OF Dot11~~WEP~~WPIKeyMappingsEntry
```



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"Conceptual table for WEPWPI Key Mappings. The MIB supports the ability to share a separate WEPWPI key for each RA/TA pair. The Key Mappings Table contains one entry for each MAC address and contains two fields for each entry: ~~WEP~~WPIOn and the corresponding ~~WEP~~WPIkey. The ~~WEP~~WPI key mappings are logically WRITE-ONLY. Attempts to read the entries in this table shall return unsuccessful status and values of null or zero. The default value for all ~~WEP~~WPIOn fields is false."
REFERENCE "~~IEEE Std 802.11, 1999~~ISO/IEC 8802-11, 2005 Edition, 8.3.2"
::= { dot11smt 4 }

dot11~~WEP~~WPIKeyMappingsEntry OBJECT-TYPE
SYNTAX Dot11~~WEP~~WPIKeyMappingsEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"An Entry (conceptual row) in the ~~WEP~~WPI Key Mappings Table.
ifIndex - Each ~~IEEE 802.11~~ISO/IEC 8802-11 interface is represented by an ifEntry. Interface tables in this MIB module are indexed by ifIndex."
INDEX { ifIndex,
dot11~~WEP~~WPIKeyMappingIndex }
::= { dot11~~WEP~~WPIKeyMappingsTable 1 }

Dot11~~WEP~~WPIKeyMappingsEntry ::=
SEQUENCE {dot11~~WEP~~WPIKeyMappingIndex Integer32,
dot11~~WEP~~WPIKeyMappingAddress MacAddress,
dot11~~WEP~~WPIKeyMappingWEPOn TruthValue,
dot11~~WEP~~WPIKeyMappingValue WEPKeytype,
dot11~~WEP~~WPIKeyMappingStatus RowStatus }

dot11~~WEP~~WPIKeyMappingIndex OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The auxiliary variable used to identify instances of the columnar objects in the ~~WEP~~WPI KeyMappings Table."
::= { dot11~~WEP~~WPIKeyMappingsEntry 1 }

dot11~~WEP~~WPIKeyMappingAddress OBJECT-TYPE
SYNTAX MacAddress
MAX-ACCESS read-create
STATUS current
DESCRIPTION



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

"The MAC address of the STA for which the values from this key mapping entry are to be used."
::= { dot11~~WEP~~WPIKeyMappingsEntry 2 }

dot11~~WEP~~WPIKeyMapping~~WEP~~WPIOn OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"Boolean as to whether ~~WEP~~WPI is to be used when communicating with the dot11~~WEP~~WPIKeyMappingAddress STA."
::= { dot11~~WEP~~WPIKeyMappingsEntry 3 }

dot11~~WEP~~WPIKeyMappingValue OBJECT-TYPE
SYNTAX WEPKeytype
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"A ~~WEP~~WPI secret key value."
::= { dot11~~WEP~~WPIKeyMappingsEntry 4 }

dot11~~WEP~~WPIKeyMappingStatus OBJECT-TYPE
SYNTAX RowStatus
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"The status column used for creating, modifying, and deleting instances of the columnar objects in the ~~WEP~~WPI key mapping Table."
DEFVAL { active }
::= { dot11~~WEP~~WPIKeyMappingsEntry 5 }
-- **********
-- * End of ~~WEP~~WPIKeyMappings TABLE
-- **********

***In "dot11PrivacyTable TABLE" in Annex D, change Dot11PrivacyEntry as follows:***

Dot11PrivacyEntry ::=
SEQUENCE {
dot11PrivacyInvoked TruthValue,
~~dot11WEPDefaultKeyID INTEGER,~~
~~dot11WEPKeyMappingLength Unsigned32,~~
dot11ExcludeUnencrypted TruthValue,
~~dot11WEPICVErrorCount Counter32,~~
~~dot11WEPExcludedCount Counter32,~~



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

dot11WAPIEnabled TruthValue,
dot11WAPIPreauthenticationEnabled TruthValue }

***In the "dot11PrivacyTable TABLE" of Annex D, change dot11PrivacyInvoked as follows:***

dot11PrivacyInvoked OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"When this attribute is ~~true~~TRUE, it shall indicate that ~~the IEEE 802.11 WEP mechanism is used~~ WPI is invoked for transmitting frames of type Data. The default value of this attribute shall be false.
For WAPI-capable clients, an additional variable dot11WAPI Enabled indicates whether WAPI is enabled. If dot11WAPIEnabled is FALSE or the MIB variable does not exist, the security mechanism invoked is undefined; if dot11WAPI Enabled is TRUE, WAPI security mechanisms invoked are configured in the dot11WAPIConfigTable. The default value of this attribute shall be FALSE."
::= { dot11PrivacyEntry 1 }

***In "dot11PrivacyTable TABLE" of Annex D, delete dot11WEPDefaultKeyID and dot11WEPKeyMappingLength.***

***In "dot11PrivacyTable TABLE" of Annex D, change dot11ExcludeUnencrypted as follows:***

dot11ExcludeUnencrypted OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"When this attribute is true, the STA shall not indicate at the MAC service interface received MSDUs that have the ~~WEP~~ Encryption subfield of the Frame Control field equal to zero. When this attribute is false, the STA may accept MSDUs that have the ~~WEP~~ Encryption subfield of the Frame Control field equal to zero. The default value of this attribute shall be false."
::= { dot11PrivacyEntry 4 }

***In "dot11PrivacyTable TABLE" of Annex D, delete WEPICVErrorCount and WEPExcludedCount.***

***In "dot11PrivacyTable TABLE" in Annex D, insert the following attributes after dot11WEPExcludedCount { dot11PrivacyEntry 6 }:***

dot11WAPIEnabled OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-write
STATUS current



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

DESCRIPTION

"When this object is set to TRUE, this shall indicate that WAPI is enabled on this entity. The entity will advertise the WAPI Information Element in its Beacon and Probe Response frames. Configuration variables for WAPI operation are found in the dot11WAPIConfigTable. This object requires that dot11PrivacyInvoked also be set to TRUE."

::= { dot11PrivacyEntry 7 }

dot11WAPIPreauthenticationEnabled OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"When this object is set to TRUE, this shall indicate that WAPI pre-authentication is enabled on this entity. This object requires that dot11WAPI Enabled also be set to TRUE."

::= { dot11PrivacyEntry 8 }

***In "SMT notification Objects" of Annex D, change dot11Deauthenticate as follows:***

dot11~~Deauthenticate~~DeLinkVerify NOTIFICATION-TYPE

OBJECTS { ifIndex, dot11~~Deauthenticate~~DeLinkVerify Reason, dot11 ~~Deauthenticate~~DeLinkVerify Station }

STATUS current

DESCRIPTION

"The ~~deauthenticate~~delinkverified notification shall be sent when the STA sends a ~~Deauthentication~~DeLinkVerification frame. The value of the notification shall include the MAC address of the MAC to which the ~~Deauthentication~~DeLinkVerification frame was sent and the reason for the ~~deauthentication~~delinkverification .

ifIndex - Each ~~IEEE 802.11~~ISO/IEC 8802-11 interface is represented by an ifEntry. Interface tables in this MIB module are indexed by ifIndex."

::= { dot11SMTnotification 0 2 }

***In "SMT notification Objects" of Annex D, change dot11AuthenticateFail as follows:***

dot11~~Authenticate~~LinkVerifyFail NOTIFICATION-TYPE

OBJECTS { ifIndex, dot11~~Authenticate~~LinkVerifyFailStatus, dot11~~Authenticate~~LinkVerifyFailStation }

STATUS current

DESCRIPTION

"The ~~authenticate~~linkverify failure notification shall be sent when the STA sends an ~~Authentication~~LinkVerification frame with a status code other than 'successful'. The value of the notification shall include the MAC address of the MAC to which the ~~Authentication~~LinkVerification frame was sent and the reason for the ~~Authentication~~LinkVerification failure.

ifIndex - Each ~~IEEE 802.11~~ISO/IEC 8802-11 interface is represented by an ifEntry. Interface tables in this MIB module are indexed by ifIndex."

::= { dot11SMTnotification 0 3 }



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

**Specifications for Enhanced Security**

***In "dot11CountersEntry TABLE" of Annex D, change Dot11CountersEntry as follows:***

```
Dot11CountersEntry ::=
        SEQUENCE { dot11TransmittedFragmentCount                Counter32,
        dot11MulticastTransmittedFrameCount                     Counter32,
        dot11FailedCount                                        Counter32,
        dot11RetryCount                                         Counter32,
        dot11MultipleRetryCount                                 Counter32,
        dot11FrameDuplicateCount                                Counter32,
        dot11RTSSuccessCount                                    Counter32,
        dot11RTSFailureCount                                    Counter32,
        dot11ACKFailureCount                                    Counter32,
        dot11ReceivedFragmentCount                              Counter32,
        dot11MulticastReceivedFrameCount                        Counter32,
        dot11FCSErrorCount                                      Counter32,
        dot11TransmittedFrameCount                              Counter32,
        ~~dot11WEPUndecryptableCount~~                          ~~Counter32~~ }
```

***In "dot11Counters TABLE" of Annex D, delete dot11WEPUndecryptableCount Counter32.***

***In "Compliance Statements" of Annex D, change OPTIONAL-GROUPS as follows:***

```
-- OPTIONAL-GROUPS { dot11SMTprivacy, dot11MACStatistics,
                        -- dot11PhyAntennaComplianceGroup, dot11PhyTxPowerComplianceGroup,
                        -- dot11PhyRegDomainsSupportGroup,
                        -- dot11PhyAntennasListGroup, dot11PhyRateGroup,
                        -- dot11SMTbase3, dot11MultiDomainCapabilityGroup,
                        -- dot11PhyFHSSComplianceGroup2, dot11WAPIadditions }
::= { dot11Compliances 1 }
```

***In "Compliance Statements" of Annex D, change the dot11Compliance as follows:***

```
dot11Compliance MODULE-COMPLIANCE
        STATUS current
        DESCRIPTION
                "The compliance statement for SNMPv2 entities that implement the ~~IEEE 802.11~~ISO/IEC 8802-11 MIB."
        MODULE -- this module
        MANDATORY-GROUPS {
                                dot11SMTbase24,
                                dot11MACbase, dot11CountersGroup,
                                dot11SmtAuthenticationAlgorithms,
                                dot11ResourceTypeID,
```



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

dot11PhyOperationComplianceGroup }

***In "Compliance Statements" of Annex D, change OPTIONAL-GROUPS as follows:***

-- OPTIONAL-GROUPS { dot11SMTprivacy, dot11MACStatistics,
-- dot11PhyAntennaComplianceGroup, dot11PhyTxPowerComplianceGroup,
-- dot11PhyRegDomainsSupportGroup,
-- dot11PhyAntennasListGroup, dot11PhyRateGroup,
-- dot11SMTbase3, dot11MultiDomainCapabilityGroup,
-- dot11PhyFHSSComplianceGroup2, dot11WAPIadditions }
::= { dot11Compliances 1 }

***In "Groups - units of conformance" of Annex D, change dot11SMTbase2 as follows:***

dot11SMTbase2 OBJECT-GROUP
OBJECTS { dot11MediumOccupancyLimit,
dot11CFPollable,
dot11CFPPeriod,
dot11CFPMaxDuration,
dot11~~Authenticate~~LinkVerifyResponseTimeOut,
dot11PrivacyOptionImplemented,
dot11PowerManagementMode,
dot11DesiredSSID,
dot11DesiredBSSType,
dot11OperationalRateSet,
dot11BeaconPeriod,
dot11DTIMPeriod,
dot11AssociationResponseTimeOut,
dot11DisassociateReason,
dot11DisassociateStation,
dot11~~Deauthenticate~~DeLinkVerifyReason,
dot11~~Deauthenticate~~DeLinkVerifyStation,
dot11~~Authenticate~~LinkVerifyFailStatus,
dot11~~Authenticate~~LinkVerifyFailStation }
STATUS ~~current~~ deprecated
DESCRIPTION
"The SMTbase2 object class provides the necessary support at the STA to manage the processes in the STA such that the STA may work cooperatively as a part of an ~~IEEE 802.11~~ISO/IEC 8802-11 network."
::= { dot11Groups 18 }

***In "Groups - units of conformance" in Annex D, insert the following objects after dot11PhyERPComplianceGroup { dot11Groups 24 }:***



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

**Specifications for Enhanced Security**

dot11WAPIadditions OBJECT-GROUP
OBJECTS { dot11WAPIEnabled,
dot11WAPIPreAuthenticationEnabled }
STATUS current
DESCRIPTION
"This object class provides the objects required to manage WAPI functionality. "
::= { dot11Groups 25 }

dot11SMTbase4 OBJECT-GROUP
OBJECTS { dot11MediumOccupancyLimit,
dot11CFPollable,
dot11CFPPeriod,
dot11CFPMaxDuration,
dot11~~Authenticate~~LinkVerifyResponseTimeOut,
dot11PrivacyOptionImplemented,
dot11PowerManagementMode,
dot11DesiredSSID,
dot11DesiredBSSType,
dot11OperationalRateSet,
dot11BeaconPeriod,
dot11DTIMPeriod,
dot11AssociationResponseTimeOut,
dot11DisassociateReason,
dot11DisassociateStation,
dot11~~Deauthenticate~~DeLinkVerifyReason,
dot11~~Deauthenticate~~DeLinkVerifyStation,
dot11~~Authenticate~~LinkVerifyFailStatus,
dot11~~Authenticate~~LinkVerifyFailStation,
dot11MultiDomainCapabilityImplemented,
dot11MultiDomainCapabilityEnabled,
dot11CountryString,
dot11WAPIOptionImplemented }
STATUS current
DESCRIPTION
"The SMTbase4 object class provides the necessary support at the STA to manage the processes in the STA so that the STA may work cooperatively as a part of an ~~IEEE 802.11~~ISO/IEC 8802-11 network."
::= { dot11Groups 26 }

***After the "Groups - units of conformance" section (and before the "End of 802.11 MIB" line) in Annex D, insert the new sections "dot11WAPIConfig Table," "dot11WAPIConfigUnicastCiphers Table," "dot11WAPIConfigAuthenticationSuites Table," "dot11WAPIStats Table," "Conformance information - WAPI," "Compliance statements - WAPI," and "Groups - units of conformance - WAPI" as follows:***



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
--*****************************************************************************
-- *dot11WAPIConfig TABLE
-- *****************************************************************************

dot11WAPIConfigTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11WAPIConfigEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "The table containing WAPI configuration objects."
::={ dot11smt 9 }

dot11WAPIConfigEntry OBJECT-TYPE
        SYNTAX Dot11WAPIConfigEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "An entry in the dot11WAPIConfig Table. There may be multiple interfaces on a proxy, each of which has
                a unique MAC address. The interfaces in this standard and in the Internet standard MIB is one-to-one, so
                the value of the ifIndex object example can be used to identify the corresponding example object directly.
                ifIndex - Each interface in this stardard is represented by an ifEntry.Interface tables in this MIB module
                are indexed by ifIndex."
        INDEX{ifIndex}
::={ dot11WAPIConfigTable 1 }

Dot11WAPIConfigEntry::=
        SEQUENCE {
                dot11WAPIConfigVersion                          Integer32,
                dot11WAPIConfigUnicastKeysSupported             Unsigned32,
                dot11WAPIConfigUnicastRekeyMethod               INTEGER,
                dot11WAPIConfigUnicastRekeyTime                 Unsigned32,
                dot11WAPIConfigUnicastRekeyPackets              Unsigned32,
                dot11WAPIConfigMulticastCipher                  OCTET STRING,
                dot11WAPIConfigMulticastRekeyMethod             INTEGER,
                dot11WAPIConfigMulticastRekeyTime               Unsigned32,
                dot11WAPIConfigMulticastRekeyPackets            Unsigned32,
                dot11WAPIConfigMulticastRekeyStrict             TruthValue,
                dot11WAPIConfigPSKValue                         OCTET STRING,
                dot11WAPIConfigPSKPassPhrase                    DisplayString,
                dot11WAPIConfigCertificateUpdateCount           Unsigned32,
                dot11WAPIConfigMulticastUpdateCount             Unsigned32,
                dot11WAPIConfigUnicastUpdateCount               Unsigned32,
```



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

dot11WAPIConfigMulticastCipherSize Unsigned32,
dot11WAPIConfigBKLifetime Unsigned32,
dot11WAPIConfigBKReauthThreshold Unsigned32,
dot11WAPIConfigSATimeout Unsigned32,
dot11WAPIAuthenticationSuiteSelected OCTET STRING,
dot11WAPIUnicastCipherSelected OCTET STRING,
dot11WAPIMulticastCipherSelected OCTET STRING,
dot11WAPIBKIDUsed OCTET STRING,
dot11WAPIAuthenticationSuiteRequested OCTET STRING,
dot11WAPIUnicastCipherRequested OCTET STRING,
dot11WAPIMulticastCipherRequested OCTET STRING }

dot11WAPIConfigVersion OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object indicates the highest version that this entity supports. "
::= { dot11WAPIConfigEntry 1}

dot11WAPIConfigUnicastKeysSupported OBJECT-TYPE
SYNTAX Unsigned32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object indicates the number of the unicast keys that a WAPI entity supports."
::= { dot11WAPIConfigEntry 2 }

dot11WAPIConfigUnicastRekeyMethod OBJECT-TYPE
SYNTAX INTEGER { disabled(1), timeBased(2), packetBased(3), timepacket-
Based(4) }
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"This object selects a mechanism for rekeying the WAPI USK. The default is time-based, once per day.
Rekeying the USK is only applicable to an entity acting in the AE or ASUE role."
DEFVAL { timeBased }
::= { dot11WAPIConfigEntry 3 }

dot11WAPIConfigUnicastRekeyTime OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
UNITS "seconds"
MAX-ACCESS read-write



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

STATUS current
DESCRIPTION
"The time in seconds after which the WAPI USK shall be refreshed. The timer shall start at the moment the USK was set using the MLME-SETWPIKEYS.request primitive."
DEFVAL { 86400 } -- once per day
::= { dot11WAPIConfigEntry 4 }

dot11WAPIConfigUnicastRekeyPackets OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
UNITS "1000 packets"
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"A packet count (in 1000s of packets) after which the WAPI USK shall be refreshed. The packet counter shall start at the moment the USK was set using the MLME-SETWPIKEYS.request primitive and it shall count all packets encrypted using the current USK."
::= { dot11WAPIConfigEntry 5 }

dot11WAPIConfigMulticastCipher OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(4))
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"This object indicates the multicast cipher suite that this entity must adopt. The WAPI information element shall adopt the value of this variable, which contains a 3-octet OUI and a one-octet cipher suite identifier."
::= { dot11WAPIConfigEntry 6 }

dot11WAPIConfigMulticastRekeyMethod OBJECT-TYPE
SYNTAX INTEGER { disabled(1), timeBased(2), packetBased(3), timepacket-Based(4) }
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"This object selects a mechanism for rekeying the WAPI MSK. The default is time-based, once per day. Rekeying the MSK is only applicable to an entity acting in the AE role."
DEFVAL { timeBased }
::= { dot11WAPIConfigEntry 7 }

dot11WAPIConfigMulticastRekeyTime OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
UNITS "seconds"
MAX-ACCESS read-write
STATUS current
DESCRIPTION



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

"The time in seconds after which the WAPI MSK shall be refreshed. The timer shall start at the moment the MSK was set using the MLME-SETWPIKEYS.request primitive."

DEFVAL { 86400 } -- once per day

::= { dot11WAPIConfigEntry 8 }

dot11WAPIConfigMulticastRekeyPackets OBJECT-TYPE

SYNTAX Unsigned32 (1..4294967295)

UNITS "1000 packets"

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"A packet count (in 1000s of packets) after which the WAPI MSK shall be refreshed. The packet counter shall start at the moment the MSK was set using the MLME-SETWPIKEYS.request primitive and it shall count all packets encrypted using the current MSK."

::= { dot11WAPIConfigEntry 9 }

dot11WAPIConfigMulticastRekeyStrict OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This object signals that the MSK shall be refreshed whenever a STA leaves the BSS that possesses the MSK."

::= { dot11WAPIConfigEntry 10 }

dot11WAPIConfigPSKValue OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(32))

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"The PSK for when WAPI in PSK mode is the selected AKM suite. In that case, the BK will obtain its value from this object. This object is logically write-only. Reading this variable shall return unsuccessful status or null or zero."

::= { dot11WAPIConfigEntry 11 }

dot11WAPIConfigPSKPassPhrase OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"The PSK, for when WAPI in PSK mode is the selected AKM suite, is configured by dot11WAPIConfigPSKValue. An alternative manner of setting the PSK uses the password-to-key algorithm. This variable provides a means to enter a pass-phrase. When this object is written, the WAPI



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

entity shall use the password-to-key algorithm to derive a pre-shared and populate dot11WAPIConfigPSKValue with this key. This object is logically write-only. Reading this variable shall return unsuccessful status or null or zero."

::= { dot11WAPIConfigEntry 12 }

dot11WAPIConfigCertificateUpdateCount OBJECT-TYPE

SYNTAX Unsigned32 (1..4294967295)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"The number of times messages in the WAPI Certificate Authentication Handshake protocol will be retried per Certificate Authentication Handshake attempt."

DEFVAL { 3 } --

::= { dot11WAPIConfigEntry 13 }

dot11WAPIConfigMulticastUpdateCount OBJECT-TYPE

SYNTAX Unsigned32 (1..4294967295)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"The number of times Message 1 in the WAPI Multicast Key Announcement Handshake will be retried per MSK Handshake attempt."

DEFVAL { 3 } --

::= { dot11WAPIConfigEntry 14 }

dot11WAPIConfigUnicastUpdateCount OBJECT-TYPE

SYNTAX Unsigned32 (1..4294967295)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"The number of times Message 1 and Message 3 in the WAPI Unicast Key Negotiation Handshake will be retried per USK Handshake attempt."

DEFVAL { 3 } --

::= { dot11WAPIConfigEntry 15 }

dot11WAPIConfigMulticastCipherSize OBJECT-TYPE

SYNTAX Unsigned32 (0..4294967295)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object indicates the length in bit of the multicast cipher key. This should be 256 for WPI."

::= { dot11WAPIConfigEntry 16 }



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
dot11WAPIConfigBKLifetime OBJECT-TYPE
        SYNTAX Unsigned32 (1..4294967295)
        UNITS "seconds"
        MAX-ACCESSread-write
        STATUS current
        DESCRIPTION
                "The maximum lifetime of a BK in the BK cache."
        DEFVAL { 43200 } --
::= { dot11WAPIConfigEntry 17 }

dot11WAPIConfigBKReauthThreshold OBJECT-TYPE
        SYNTAX Unsigned32 (1..100)
        UNITS "percentage"
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "The percentage of the BK lifetime that should expire before a WAI re-authentication occurs."
        DEFVAL { 70 } --
::= { dot11WAPIConfigEntry 18 }

dot11WAPIConfigSATimeout OBJECT-TYPE
        SYNTAX Unsigned32 (1..4294967295)
        UNITS "seconds"
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "The maximum time a security association shall take to set up."
        DEFVAL { 60 } --
::= { dot11WAPIConfigEntry 19 }

dot11WAPIAuthenticationSuiteSelected OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(4))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The selector of the last AKM suite negotiated."
::= { dot11WAPIConfigEntry 20 }

dot11WAPIUnicastCipherSelected OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(4))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
```



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

"The selector of the last unicast cipher suite negotiated."

::= { dot11WAPIConfigEntry 21 }

dot11WAPIMulticastCipherSelected OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(4))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The selector of the last multicast cipher suite negotiated."

::= { dot11WAPIConfigEntry 22 }

dot11WAPIBKIDUsed OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(16))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The selector of the last BKID used in the last Unicast Key Negotiation Handshake."

::= { dot11WAPIConfigEntry 23 }

dot11WAPIAuthenticationSuiteRequested OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(4))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The selector of the last AKM suite requested."

::= { dot11WAPIConfigEntry 24 }

dot11WAPIUnicastCipherRequested OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(4))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The selector of the last unicast cipher suite requested."

::= { dot11WAPIConfigEntry 25 }

dot11WAPIMulticastCipherRequested OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(4))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The selector of the last multicast cipher suite requested."

::= { dot11WAPIConfigEntry 26 }



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
--*****************************************************************************
-- *End of dot11WAPIConfig TABLE
-- *****************************************************************************

-- *****************************************************************************
-- * dot11WAPIConfigUnicastCiphers TABLE
-- *****************************************************************************

dot11WAPIConfigUnicastCiphersTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11WAPIConfigUnicastCiphersEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "This table lists the unicast cipher suites supported by this entity. It allows enabling and disabling of each
                unicast cipher suite by network management. The unicast cipher suite list in the WAPI Information
                Element is formed using the information in this table."
::= { dot11smt 10 }

dot11WAPIConfigUnicastCiphersEntry OBJECT-TYPE
        SYNTAX Dot11WAPIConfigUnicastCiphersEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "The table entry, indexed by the interface index (or all interfaces) and the unicast cipher suite."
        INDEX { dot11WAPIConfigIndex, dot11WAPIConfigUnicastCipherIndex }
::= { dot11WAPIConfigUnicastCiphersTable 1 }

Dot11WAPIConfigUnicastCiphersEntry ::=
        SEQUENCE {
                dot11WAPIConfigUnicastCipherIndex          Unsigned32,
                dot11WAPIConfigUnicastCipher               OCTET STRING,
                dot11WAPIConfigUnicastCipherEnabled        TruthValue,
                dot11WAPIConfigUnicastCipherSize           Unsigned32 }

dot11WAPIConfigUnicastCipherIndex OBJECT-TYPE
        SYNTAX Unsigned32 (1..4294967295)
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "The auxiliary index into the dot11WAPIConfigUnicastCiphersTable."
::= { dot11WAPIConfigUnicastCiphersEntry 1 }

dot11WAPIConfigUnicastCipher OBJECT-TYPE
```



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        SYNTAX OCTET STRING (SIZE(4))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The selector of a supported unicast cipher suite. It consists of an OUI (the first 3 octets) and a cipher suite
                identifier (the last octet)."
::= { dot11WAPIConfigUnicastCiphersEntry 2 }

dot11WAPIConfigUnicastCipherEnabled OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "This object enables or disables the unicast cipher."
::= { dot11WAPIConfigUnicastCiphersEntry 3 }

dot11WAPIConfigUnicastCipherSize OBJECT-TYPE
        SYNTAX Unsigned32 (0..4294967295)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This object indicates the length in bit of the unicast cipher key. This should be 256 for WPI."
   ::= { dot11WAPIConfigUnicastCiphersEntry 4 }

-- ****************************************************************
-- * End of dot11WAPIConfigUnicastCiphers TABLE
--****************************************************************

-- ****************************************************************
-- * dot11WAPIConfigAuthenticationSuites TABLE
-- ****************************************************************

dot11WAPIConfigAuthenticationSuitesTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11WAPIConfigAuthenticationSuitesEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "This table lists the AKM suites supported by this entity. Each AKM suite can be individually enabled and
                disabled. The AKM suite list in the WAPI information element is formed using the information in this
                table."
::= { dot11smt 11 }

dot11WAPIConfigAuthenticationSuitesEntry OBJECT-TYPE
```



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        SYNTAX Dot11WAPIConfigAuthenticationSuitesEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "An entry in the dot11WAPIConfigAuthenticationSuitesTable."
        INDEX { dot11WAPIConfigAuthenticationSuiteIndex }
::= { dot11WAPIConfigAuthenticationSuitesTable 1 }

Dot11WAPIConfigAuthenticationSuitesEntry ::=
        SEQUENCE {
                dot11WAPIConfigAuthenticationSuiteIndex         Unsigned32,
                dot11WAPIConfigAuthenticationSuite               OCTET STRING,
                dot11WAPIConfigAuthenticationSuiteEnabled        TruthValue }

dot11WAPIConfigAuthenticationSuiteIndex OBJECT-TYPE
        SYNTAX Unsigned32 (1..4294967295)
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "The auxiliary variable used as an index into the dot11WAPIConfigAuthenticationSuitesTable."
::= { dot11WAPIConfigAuthenticationSuitesEntry 1 }

dot11WAPIConfigAuthenticationSuite OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(4))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The selector of an AKM suite. It consists of an OUI (the first 3 octets) and a cipher suite identifier (the
                last octet)."
::= { dot11WAPIConfigAuthenticationSuitesEntry 2 }

dot11WAPIConfigAuthenticationSuiteEnabled OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "This variable indicates whether the corresponding AKM suite is enabled/disabled."
::= { dot11WAPIConfigAuthenticationSuitesEntry 3 }

-- ************************************************************************
-- * End of dot11WAPIConfigAuthenticationSuites TABLE
--************************************************************************
```



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
--*********************************************************************
-- * dot11WAPIStats TABLE
-- *********************************************************************

dot11WAPIStatsTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Dot11WAPIStatsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "This table maintains per-STA statistics in a WAPI. The entry with dot11WAPIStatsSTAAddress set to
                FF-FF-FF-FF-FF-FF shall contain statistics for broadcast/multicast traffic."
::= { dot11smt 12 }

dot11WAPIStatsEntry OBJECT-TYPE
        SYNTAX Dot11WAPIStatsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "An entry in the dot11WAPIStatsTable."
        INDEX { dot11WAPIConfigIndex, dot11WAPIStatsIndex }
::= { dot11WAPIStatsTable 1 }

Dot11WAPIStatsEntry ::=
        SEQUENCE {
                dot11WAPIStatsIndex                             Unsigned32,
                dot11WAPIStatsSTAAddress                        MacAddress,
                dot11WAPIStatsVersion                           Unsigned32,
                dot11WAPIStatsSelectedUnicastCipher             OCTET STRING,
                dot11WAPIStatsWPIReplayCounters                 Counter32,
                dot11WAPIStatsWPIDecryptableErrors              Counter32,
                dot11WAPIStatsWPIMICErrors                      Counter32,
                dot11WAPIStatsWAISignatureErrors                Counter32,
                dot11WAPIStatsWAIHMACErrors                     Counter32,
                dot11WAPIStatsWAIAuthenticationResultFailures   Counter32,
                dot11WAPIStatsWAIDiscardCounters                Counter32,
                dot11WAPIStatsWAITimeoutCounters                Counter32,
                dot11WAPIStatsWAIFormatErrors                   Counter32,
                dot11WAPIStatsWAICertificateHandshakeFailures   Counter32,
                dot11WAPIStatsWAIUnicastHandshakeFailures       Counter32,
                dot11WAPIStatsWAIMulticastHandshakeFailures     Counter32
                }

dot11WAPIStatsIndex OBJECT-TYPE
```



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

SYNTAX Unsigned32 (1..4294967295)
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"An auxiliary index into the dot11WAPIStatsTable."
::= { dot11WAPIStatsEntry 1 }

dot11WAPIStatsSTAAddress OBJECT-TYPE
SYNTAX MacAddress
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"STA’s MAC address"
::= { dot11WAPIStatsEntry 2 }

dot11WAPIStatsVersion OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The WAPI version with which the STA associated."
::= { dot11WAPIStatsEntry 3 }

dot11WAPIStatsSelectedUnicastCipher OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(4))
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The unicast cipher suite selector used during association."
::= { dot11WAPIStatsEntry 4 }

dot11WAPIStatsWPIReplayCounter OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of received WPI MPDUs discarded by the replay mechanism."
::= { dot11WAPIStatsEntry 5 }

dot11WAPIStatsWPIDecryptableErrors OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

DESCRIPTION

"The number of received MPDUs discarded by the WPI decryption algorithm because of no valid key."

::= { dot11WAPIStatsEntry 6 }

dot11WAPIStatsWPIMICErrors OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of received MPDUs discarded because of the MIC check error during the WPI decryption."

::= { dot11WAPIStatsEntry 7 }

dot11WAPIStatsWAISignatureErrors OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

“This counter shall increment when the signature in the received WAI packet is wrong.”

::={ dot11WAPIStatsEntry 8}

dot11WAPIStatsWAIHMACErrors OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

“This counter shall increment when the message authentication code in the received WAI packet is wrong.”

::={ dot11WAPIStatsEntry 9}

dot11WAPIStatsWAIAuthenticationResultFailures OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

“This counter shall increment when the WAI authentication is unsuccessful.”

::={ dot11WAPIStatsEntry 10}

dot11WAPIStatsWAIDiscardCounters OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

“This counter shall increment when the received WAI packet is discarded.”



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

::={ dot11WAPIStatsEntry 11}

dot11WAPIStatsWAITimeoutCounters OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This counter shall increment when the WAI packet is timeout."
::={ dot11WAPIStatsEntry 12}

dot11WAPIStatsWAIFormatErrors OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This counter shall increment when there exists format error in the WAI packet."
::={ dot11WAPIStatsEntry 13}

dot11WAPIStatsWAICertificateHandshakeFailures OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This counter shall increment when the WAI certificate authentication is unsuccessful."
::={ dot11WAPIStatsEntry 14}

dot11WAPIStatsWAIUnicastHandshakeFailures OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This counter shall increment when the WAI unicast key negotiation process is unsuccessful."
::={ dot11WAPIStatsEntry 15}

dot11WAPIStatsWAIMulticastHandshakeFailures OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This counter shall increment when the WAI multicast key announcement process is unsuccessful."
::={ dot11WAPIStatsEntry 16}

-- **********



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
-- * End of dot11WAPIStats TABLE
-- ************************************************************************

__************************************************************************
-- * Conformance information – WAPI
-- ************************************************************************

dot11WAPICompliance MODULE-COMPLIANCE
        STATUS current
        DESCRIPTION
                "Implement comformance statement of the SNMPv2 entity in the WAPI MIB. "
        MODULE -- this module
        MANDATORY-GROUPS {
                                dot11WAPIBase }
        -- OPTIONAL-GROUPS {dot11WAPIBKcachingGroup }
::= { dot11Compliances 2 }

-- ************************************************************************
-- * Groups - units of conformance– WAPI
-- ************************************************************************

dot11WAPIBase OBJECT-GROUP
        OBJECTS {
                dot11WAPIConfigVersion,
                dot11WAPIConfigUnicastKeysSupported,
                dot11WAPIConfigUnicastRekeyMethod,
                dot11WAPIConfigUnicastRekeyTime,
                dot11WAPIConfigUnicastRekeyPackets,
                dot11WAPIConfigMulticastCipher,
                dot11WAPIConfigMulticastRekeyMethod,
                dot11WAPIConfigMulticastRekeyTime,
                dot11WAPIConfigMulticastRekeyPackets,
                dot11WAPIConfigMulticastRekeyStrict,
                dot11WAPIConfigPSKValue,
                dot11WAPIConfigPSKPassPhrase,
                dot11WAPIConfigCertificateUpdateCount,
                dot11WAPIConfigMulticastUpdateCount,
                dot11WAPIConfigUnicastUpdateCount,
                dot11WAPIConfigMulticastCipherSize,
                dot11WAPIConfigUnicastCipher,
                dot11WAPIConfigUnicastCipherEnabled,
                dot11WAPIConfigUnicastCipherSize,
                dot11WAPIConfigAuthenticationSuite,
```



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                dot11WAPIConfigAuthenticationSuiteEnabled,
                dot11WAPIConfigSATimeout,
                dot11WAPIAuthenticationSuiteSelected,
                dot11WAPIUnicastCipherSelected,
                dot11WAPIMulticastCipherSelected,
                dot11WAPIBKIDUsed,
                dot11WAPIAuthenticationSuiteRequested,
                dot11WAPIUnicastCipherRequested,
                dot11WAPIMulticastCipherRequested,
                dot11WAPIStatsSTAAddress,
                dot11WAPIStatsVersion,
                dot11WAPIStatsSelectedUnicastCipher,
                dot11WAPIStatsWPIReplayCounters,
                dot11WAPIStatsWPIDecryptableErrors,
                dot11WAPIStatsWPIMICErrors,
                dot11WAPIStatsWAISignatureErrors,
                dot11WAPIStatsWAIHMACErrors,
                dot11WAPIStatsWAIAuthenticationResultFailures,
                dot11WAPIStatsWAIDiscardCounters,
                dot11WAPIStatsWAITimeoutCounters,
                dot11WAPIStatsWAIFormatErrors,
                dot11WAPIStatsWAICertificateHandshakeFailures,
                dot11WAPIStatsWAIUnicastHandshakeFailures,
                dot11WAPIStatsWAIMulticastHandshakeFailures
                }
        STATUS current
        DESCRIPTION
                        " The dot11WAPIBase class provides the necessary support to manage the WAPI function of STA. "
::= { dot11Groups 26 }

dot11WAPIBKcachingGroup OBJECT-GROUP
        OBJECTS {
                  dot11WAPIConfigBKLifetime,
                  dot11WAPIConfigBKReauthThreshold
                  }
        STATUS current
        DESCRIPTION
                    "The dot11WAPIBKcachingGroup class provides the necessary support to manage the BK cache function
                    of STA."
::= { dot11Groups 27 }
```



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

*Insert the following text for Annex H and Annex I following Annex G:*

# Annex H

(Informative )

# Reference implementations of the frame authentication algorithm and the key derivation algorithm and the test vectors

## H.1 Frame authentication algorithm

### H.1.1 Reference implementation

*int hmac_sha256(unsigned char *text, int text_len, byte *key, unsigned key_len, byte *digest,unsigned digest_length)*

/*

a) unsigned char *text , indicates the text to be calculated by HMAC;

b) unsigned text_len , indicates the length of the text to be calculated by HMAC (in octet);

c) byte *key , indicates the key used in the HMAC computation;

d) unsigned key_len, indicates the length of the key used in the HMAC computation (in octet);

e) byte *digest , indicates the output digest of HMAC;

f) unsigned digest_length , specifies the length of the output digest of HMAC (in octet) which must be less than that of the hash algorithm sha256 (in octet);

g) The returned value. The non-zero values indicate the practical length of the output digest, and the value of 0 indicates a failure.

*/

*{*

```
byte real_key[SHA256_BLOCK_SIZE];
byte ipad[SHA256_BLOCK_SIZE];
byte opad[SHA256_BLOCK_SIZE];
byte temp_digest1[SHA256_DIGEST_SIZE];
byte temp_digest2[SHA256_DIGEST_SIZE];
CONTX input_data[2];
unsigned i;

if digest_length>SHA256_DIGEST_SIZE)
    return 0;
```



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
for(i=0;i< SHA256_BLOCK_SIZE;i++){
    real_key[i]=0;
    ipad[i]=0x36;
    opad[i]=0x5c;
}

/* if key_len is larger than hash block size, key is hashed first to make its length is equal hash block size */
if(key_len> SHA256_BLOCK_SIZE){
    input_data[0].buff=key;
    input_data[0].length=key_len;
    SHA256(input_data,1,real_key);
    key_len= SHA256_ BLOCK _SIZE;
}
else
    memcpy(real_key,key,key_len);

for(i=0;i< SHA256_BLOCK_SIZE;i++){
    ipad[i]^=real_key[i];
    opad[i]^=real_key[i];
}

/*sha256(Key xor ipad,text)=temp_digest1 */
input_data[0].buff=ipad;
input_data[0].length= SHA256_BLOCK_SIZE;
input_data[1].buff=text;
input_data[1].length=text_len;
SHA256(input_data,2,temp_digest1);

/*sha256(Key xor opad,temp_digest1)=temp_digest2 */
input_data[0].buff=opad;
input_data[0].length= SHA256_BLOCK_SIZE;
input_data[1].buff=temp_digest1;
input_data[1].length SHA256_DIGEST_SIZE;
```



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

*SHA256(input_data,2,temp_digest2);*

*/*output the digest of required length */*

*memcpy(digest,temp_digest2,digest_length);*

*return digest_length;*

}

### H.1.2 Test vectors

| **Test vector 1** | |
|---|---|
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 |
| Length of key | 32 |
| Data | **abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqabcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq**<br>0x61 62 63 64 62 63 64 65 63 64 65 66 64 65 66 67 65 66 67 68 66 67 68 69 67 68 69 6a 68 69 6a 6b 69 6a 6b 6c 6a 6b 6c 6d 6b 6c 6d 6e 6c 6d 6e 6f 6d 6e 6f 70 6e 6f 70 71 61 62 63 64 62 63 64 65 63 64 65 66 64 65 66 67 65 66 67 68 66 67 68 69 67 68 69 6a 68 69 6a 6b 69 6a 6b 6c 6a 6b 6c 6d 6b 6c 6d 6e 6c 6d 6e 6f 6d 6e 6f 70 6e 6f 70 71 |
| Length of data | 112 |
| Digest | 0x47 03 05 fc 7e 40 fe 34 d3 ee b3 e7 73 d9 5a ab 73 ac f0 fd 06 04 47 a5 eb 45 95 bf 33 a9 d1 a3 |

| **Test vector 2** | |
|---|---|
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 |
| Length of key | 37 |
| Data | 0xcd repeated by 50 times |
| Length of data | 50 |
| Digest | 0xd4 63 3c 17 f6 fb 8d 74 4c 66 de e0 f8 f0 74 55 6e c4 af 55 ef 07 99 85 41 46 8e b4 9b d2 e9 17 |

| **Test vector 3** | |
|---|---|



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

| | |
|---|---|
| Key | 0x0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b |
| Length of key | 32 |
| Data | **Hi There**<br>0x48 69 20 54 68 65 72 65 |
| Length of data | 8 |
| Digest | 0x19 8a 60 7e b4 4b fb c6 99 03 a0 f1 cf 2b bd c5 ba 0a a3 f3 d9 ae 3c 1c 7a 3b 16 96 a0 b6 8c f7 |

| | |
|---|---|
| **Test vector 4** | |
| Key | 0x4a 65 66 65 |
| Length of key | 4 |
| Data | **what do ya want for nothing?**<br>0x77 68 61 74 20 64 6f 20 79 61 20 77 61 6e 74 20 66 6f 72 20 6e 6f 74 68 69 6e 67 3f |
| Length of data | 28 |
| Digest | 0x5b dc c1 46 bf 60 75 4e 6a 04 24 26 08 95 75 c7 5a 00 3f 08 9d 27 39 83 9d ec 58 b9 64 ec 38 43 |

## H.2 Key derivation algorithm

### H.2.1 Reference implementation

**void KD_hmac_sha256(byte *text,unsigned text_len,byte *key,unsigned key_len,byte *output,unsigned length)**

**/***

a) byte *text , indicates the input text of the key derivation algorithm;

b) unsigned text_len , indicates the length of the input text (in octet);

c) byte *key, indicates the input key of the key derivation algorithm;

d) unsigned key_len , indicates the length of the input key (in octet);

e) byte *output , indicates the output of the key derivation algorithm;

f) unsigned length , indicates the length of the output of the key derivation algorithm (in octet).

***/**

***{***

***int i;***



--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
for(i=0;length/SHA256_DIGEST_SIZE;i++,length-=SHA256_DIGEST_SIZE){
        hmac(MHASH_SHA256,text,text_len,key,key_len,&output[i*SHA256_DIGEST_SIZE
    ],SHA256_DIGEST_SIZE);
  text=&output[i*SHA256_DIGEST_SIZE];
  text_len=SHA256_DIGEST_SIZE;
}
if(length>0)
   hmac(MHASH_SHA256,text,text_len,key,key_len,&output[i*SHA256_DIGEST_SIZE],length);
}
```

### H.2.2 Test vectors

| **Test vector 1** | |
|---|---|
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 |
| Length of key | 32 |
| Data | **pairwise key expansion for infrastructure unicast**<br>0x70 61 69 72 77 69 73 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 69 6e 66 72 61 73 74 72 75 63 74 75 72 65 20 75 6e 69 63 61 73 74 |
| Length of data | 49 |
| Output | 0xe3 a6 45 46 f2 d1 f5 ee b7 d1 ee 06 d2 c9 e5 4a 2c c9 d6 ce c3 b7 6f fd 62 63 f4 26 dc 25 39 af bd 98 80 a5 27 a1 b5 85 59 4b 57 ce 33 21 4f 0c |
| Length of output | 48 |

| **Test vector 2** | |
|---|---|
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 |
| Length of key | 37 |
| Data | **pairwise key expansion for infrastructure unicast**<br>0x70 61 69 72 77 69 73 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 69 6e 66 72 61 73 74 72 75 63 74 75 72 65 20 75 6e 69 63 61 73 74 |
| Length of data | 49 |



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

| | |
|---|---|
| Output | 0x3b 6e ca 4f 08 76 c4 3a b3 1b 26 3f 2c 38 b8 81 21 b5 68 e5 f8 fd 1d 4c fa 4c 7f 8c 60 97 04 3d 7b 40 a8 63 b9 43 b9 f5 bb 37 2f 3a dd a5 da 27 |
| Length of output | 48 |

| | |
|---|---|
| **Test vector 3** | |
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 |
| Length of key | 16 |
| Data | **pairwise key expansion for infrastructure unicast** 0x70 61 69 72 77 69 73 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 69 6e 66 72 61 73 74 72 75 63 74 75 72 65 20 75 6e 69 63 61 73 74 |
| Length of data | 49 |
| Output | 0xbc 29 f3 e6 09 1f 6a c9 0b a0 20 61 92 12 48 69 5f ee ff 1a 4c ab 53 3b 11 67 d8 54 5f 93 5f 28 11 84 c9 bb 32 f9 87 b9 86 81 0f fb 17 c4 10 f5 |
| Length of output | 48 |

| | |
|---|---|
| **Test vector 4** | |
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 |
| Length of key | 32 |
| Data | **group key expansion for multicast and broadcast** 0x67 72 6f 75 70 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 6d 75 6c 74 69 63 61 73 74 20 61 6e 64 20 62 72 6f 61 64 63 61 73 74 |
| Length of data | 47 |
| Output | 0x20 8f 72 54 a4 bf 56 f0 fa 49 5f e1 0c 99 15 05 92 ed 79 df 57 74 a9 6e 13 97 1e c4 a1 5e 16 a7 ed 75 f5 e5 44 bb d3 35 67 eb 88 e7 83 24 a9 d2 |
| Length of output | 48 |
| **Test vector 5** | |
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 |



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

| | |
|---|---|
| Length of key | 37 |
| Data | **group key expansion for multicast and broadcast**<br>0x67 72 6f 75 70 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 6d 75 6c 74 69 63 61 73 74 20 61 6e 64 20 62 72 6f 61 64 63 61 73 74 |
| Length of data | 47 |
| Output | 0x33 32 61 7a 90 8e a5 a0 7f fa 1d 23 79 f3 d8 3e 8b e9 14 1f 15 53 8f d3 ef de 58 01 19 e8 c5 09 5d 25 b2 d3 0a c7 a6 35 ad b4 3c 6c ac f0 aa 2b |
| Length of output | 48 |

| **Test vector 6** | |
|---|---|
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 |
| Length of key | 16 |
| Data | **group key expansion for multicast and broadcast**<br>0x67 72 6f 75 70 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 6d 75 6c 74 69 63 61 73 74 20 61 6e 64 20 62 72 6f 61 64 63 61 73 74 |
| Length of data | 47 |
| Output | 0xf2 cb f1 1c 6d 40 b8 09 d0 c0 ed 48 2a 4a 1b 6a 15 1a f1 fb 4c 80 f9 80 5c 93 e5 6e b1 cf 5c b5 ec c1 3e 7a bc af e0 a7 d2 59 5d 51 9b 76 9a 24 |
| Length of output | 48 |

| **Test vector 7** | |
|---|---|
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 |
| Length of key | 32 |
| Data | **pre-share key expansion for adhoc network**<br>0x70 72 65 2d 73 68 61 72 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 61 64 68 6f 63 20 6e 65 74 77 6f 72 6b |
| Length of data | 41 |
| Output | 0xc0 7a d8 32 25 2a 0c 14 76 18 f4 c0 d0 6b 35 f4 f6 d6 73 5d 1a a3 8e 47 9a 7e e0 ac 1c 0c 38 5b 2d 33 28 74 1e 4d a0 c8 76 fc 6c c9 e3 60 c8 d7 |



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

| | |
|---|---|
| Length of output | 48 |

| **Test vector 8** | |
|---|---|
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 |
| Length of key | 37 |
| Data | **pre-share key expansion for adhoc network**<br>0x70 72 65 2d 73 68 61 72 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 61 64 68 6f 63 20 6e 65 74 77 6f 72 6b |
| Length of data | 41 |
| Output | 0x f0 0b ee f2 f5 5f 85 d8 ee b0 6f 8c c4 1b e6 0e c2 69 f5 82 9a 0b 6e fb 2d 9b 49 5e b1 87 d3 58 59 68 88 c3 d2 6f 94 9f 8d 2e 41 fe bc bb b9 9a |
| Length of output | 48 |

| **Test vector 9** | |
|---|---|
| Key | 0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 |
| Length of key | 16 |
| Data | **pre-share key expansion for adhoc network**<br>0x70 72 65 2d 73 68 61 72 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 61 64 68 6f 63 20 6e 65 74 77 6f 72 6b |
| Length of data | 41 |
| Output | 0x05 8e b8 7c ff 82 66 47 de 50 7b 14 17 ac 99 6e b5 7f cf 11 fd fc 83 be 59 d5 85 f4 a7 3e 69 7d d4 38 e3 34 fe bb 06 7d 14 6f 01 31 a6 96 4f 26 |
| Length of output | 48 |



Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

# Annex I

**(Informative)**

# The example of WAI parameters and WPI block cryptographic algorithm

## I.1 Principle

Cryptographic algorithms and ECC parameter to be applied in information security mechanism may be subject to national and regional regulations. It shall conform to national laws and regulations, and can be chosen according to specific requirements in different countries and regions. An appropriate block algorithm shall be adopted in WPI, and which type of algorithm to be adopted is optional. An appropriate ECC parameters shall be adopted in WAI, and which parameters to be adopted is optional.

An appropriate algorithm means that this algorithm should meet the requirements about the block size and the key length, i.e. 128 bits.

## I.2 Algorithm used in China

In this standard, no specific cryptographic algorithms are defined. Cryptographic algorithms to be applied in WPI may be subject to national and regional regulations, and they should have a 128-bit block size and a 128-bit key length.
In China, the adopted cryptographic algorithm is SMS4, which has a 128-bit block size and a 128-bit key length.
The corresponding OUI of SMS4 is 00-14-72 and the Suite Type is 1. The algorithm SMS4 is owned by Beijing Data Security Technology Co. Ltd.(BDST) (E-Mail: chinabdst@126.com) .

## I.3 ECC parameters used in China

In this standard, no specific ECC parameters are defined. ECC parameters to be applied in WAI may be subject to national and regional regulations.
The OID corresponding to ECC parameters chosen in China is 1.2.156.1001.5.40.1. The OID 1.2.156.1001.5.40.1 is also applied in GBW and ECDH. The detailed information can be obtained from Beijing Data Security Technology Co. Ltd.(BDST). (E-Mail: chinabdst@126.com).



--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale