Errata to P802.11D5.3

The following corrections apply to P802.11D5.3, 17 March 1997, as submitted for recirculation ballot.

1. **Page ii (Abstract), line 6:**
   
   Change: “2,400 - 2,483.5 MHz band” to “2,400 - 2,500 MHz band”
   
   Reason: The upper end of the frequency range in at least one regulatory domain (Japan) is 2,500MHz.

   Change the last paragraph to read: This standard includes the PICS proforma in Annex A, tables for Frequency Hopping patterns in Annex B, and state diagrams for medium access control using the SDL standard in Annex C, as well as the MIB definition in ASN.1 notation in Annex D.
   
   Reason: The annex structure was changed.

2. **Page vii:**
   
   delete: ERROR! Bookmark not defined.

3. **Clause 3.1 Access point:**
   
   delete extra space at start of paragraph.

4. **Clause 5.1.1.2 paragraphs a) and f):**
   
   Add period at end of sentence.

5. **Clause 7.3.2.2, first 2 paragraphs should read:**

   The Supported Rates element specifies all the rates which this STA is capable of receiving. The information field is encoded as 1 to 8 octets where each octet describes a single Supported Rate in units of 500 kbit/s.

   Within Beacon, Probe Response, Association Response and Reassociation Response Management frames, each Supported Rate belonging to the BSSBasicRateSet as defined in 10.3.10.1, is encoded as an octet with the most significant bit (bit 7) set to 1 (e.g. a 1 Mbit/s rate belonging to the BSSBasicRateSet is encoded as 0x828A). Rates not belonging to the BSSBasicRateSet are encoded with the most significant bit set to 0 (e.g. a 2 Mbit/s rate not belonging to the BSSBasicRateSet is encoded as 0x0414). The most significant bit of each Supported Rate octet in other Management frame types is ignored by receiving STAs.

   Reason: Complete the editing needed to implement the decision to use a uniform encoding of supported data rates throughout the MAC, and to increase the maximum representable data rate from 12.7Mbit/s to 63.5Mbit/s in order to accommodate faster PHYs in the future. The remainder of this change is already correct in D5.3 Clauses 13.1.4.23 and 13.1.4.24, and in Annex C.

   Also, the following changes are needed for consistency in Clauses 10.3.2.2 and 10.3.10.1:

   | BSSBasicRateSet | set of integers | 24 through 12755 inclusive (for each integer in the set) | The set of data rates (in units of 500 kbit/s/400 Kbps) that must be supported by all STAs that desire to join this BSS. The STAs must be able to receive at each of the data rates listed in the set. |

   Also, the following changes are needed for consistency in Clauses 10.3.3.1 and 10.3.10.1:
OperationalRateSet set of integers 21 through 127 255 inclusive (for each integer in the set)


| OperationalRateSet | set of integers | 21 through 127 255 inclusive (for each integer in the set) | The set of data rates (in units of 500 kbit/s 100Kbps) that the STA desires to use for communication within the BSS. The STA must be able to receive at each of the data rates listed in the set. The OperationalRateSet is a superset of the BSSBasicRateSet advertised by the BSS. |

6. **Clause 8.3.2, first decision tree should read:**

When transmitting a frame of type Data, the values of aPrivacyInvoked, aWEPKeyMappings, aWEPDefaultKeys, and aWEPDefaultKeyID in effect at an unspecified time between receipt by the MAC of the MAUNITDATA.request primitive and the time of transmission of that frame shall be used according to the following decision tree:

```plaintext
if aPrivacyInvoked is “false”
    the MPDU is transmitted without encryption
else
    if (the MPDU has an individual RA and
        there is an entry in aWEPKeyMappings for that RA)
        if that entry has WEPOn set to “false”
            the MPDU is transmitted without encryption
        else
            if that entry contains a key that is null
                discard the entire MSDU and generate an
                MA-UNITDATA-STATUS.indication primitive to
                notify LLC that the MSDU was undeliverable due to
                a null WEP key
            else
                encrypt the MPDU using that entry’s key, setting the keyID
                subfield of the IV field to zero
        else
            if (the MPDU has a group RA and
                the Privacy subfield of the Capability Information field in this BSS is set to 0)
                the MPDU is transmitted without encryption
            else
                if aWEPDefaultKeys[aWEPDefaultKeyID] is null
                    discard the MSDU and generate an
                    MA-UNITDATA-STATUS.indication primitive to
                    notify LLC that the entire MSDU was undeliverable
                    due to a null WEP key
                else
                    encrypt the MPDU using aWEPDefaultKeys[aWEPDefaultKeyID],
                    setting the KeyID subfield of the IV field to aWEPDefaultKeyID
```

Also: Add this conditional test to the formal description by placing a decision symbol on the “true” exit from the first decision symbol in the leftmost column of Procedure Encrypt on page C-85.

Reason: Consistency with decision to invoke BSS-wide encryption of broadcasts and multicasts when the Privacy Subfield is set to 1.

7. **Clause 9.2.5.4:**

Add a dash to PHY-CCARESET.

Add a dash to PHY-RXSTART and PHY-RXEND.
8. **Clause 9.2.5.6, second paragraph should read:**

Each frame contains information that defines the duration of the next transmission. The duration information from RTS frames shall be used to update the NAV to indicate busy until the end of ACK 0. The duration information from the CTS frame shall also be used to update the NAV to indicate busy until the end of ACK 0. Both Fragment 0 and ACK 0 shall contain duration information to update the NAV to indicate busy until the end of ACK 1. This shall be done by using the Duration/ID field in the Data and ACK frames. This shall continue until the last Fragment which shall have a duration of one ACK time plus one SIFS time, and its ACK which shall have its Duration/ID field set to zero. Each fragment and ACK acts as a virtual RTS and CTS; therefore no further RTS/CTS frames need to be generated after the RTS/CTS that began the frame exchange sequence even though subsequent fragments may be larger than the aRTSThreshold. At stations using a Frequency Hopping PHY, when there is insufficient time before the next dwell boundary to transmit the subsequent fragment, the station initiating the frame exchange sequence may set the Duration/ID field in the last Data or Management frame to be transmitted before the dwell boundary to the duration of one ACK time plus one SIFS time.

Also: Attach a comment symbol stating that this behavior is permissible to the task symbol that calculates the value of variable “tdur” in the leftmost column on page C-91 for the station formal description and on page C-136 for the access point formal description.

Reason: To remove a constraint which mandated an unnecessarily long duration value near the end of a dwell, which has the effect of setting the NAV at other stations to protect a frame that will not be transmitted. When the shorter duration value is used, another station may be able to transmit a shorter frame prior to the dwell boundary, permitting more efficient use of the time available during the dwell.

9. **Clause 9.2.5.7, second paragraph:**

Add a dash to the PHY primitives (5 occasions).

10. **Clause 9.2.8, last paragraph:**

Add a dash to the PHY primitives (6 occasions).

11. **Clause 9.2.9:**

Remove the space before the period at the end of the fifth paragraph.

12. **Clause 9.2.10, second paragraph following Figure 59:**

Change as follows:

\[ \text{EIFS} = \text{aSIFSTime} + (8 \times \text{ACKSize}) + \text{aPreambleLength} + \text{aPLCPHeaderLength} + \text{DIFS} \]

where ACKSize is the length, in bytes, of an ACK frame and \((8 \times \text{ACKSize}) + \text{aPreambleLength} + \text{aPLCPHeaderLength}\) is expressed in microseconds required to transmit at the PHY’s lowest mandatory rate.

Reason: consistency with MIB.
13. **Clause 9.6, third paragraph from the end:**

Change 100 kBit/s into 500 kbit/s.

14. **Clause 10.3.2.2, add a row to the BSSDescription table, just below Timestamp:**

<table>
<thead>
<tr>
<th>Local Time</th>
<th>integer</th>
<th>N/A</th>
<th>The value of the station’s TSF timer at the start of reception of the first octet of the timestamp field of the received frame (probe response or beacon) from the found BSS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Time</td>
<td>integer</td>
<td>N/A</td>
<td>The value of the station’s TSF timer at the start of reception of the first octet of the timestamp field of the received frame (probe response or beacon) from the found BSS.</td>
</tr>
</tbody>
</table>

14.1. and add a corresponding item to the BssDscr definition on page C-15 of the formal description:

```c
newtype BssDscr struct
    bdBssId    MacAddr;
    bdSsId     Octetstring; /* 1 <= length <= 32 */
    bdType     BssType;
    bdBcnPer   Kusec; /* beacon period in Kusec */
    bdDtimPer  Integer; /* DTIM period in beacon periods */
    bdTstamp   Octetstring; /* 8 Octets from ProbeRsp/Beacon */
    bdStartTs  Octetstring; /* 8 Octets TSF when rx Tstamp */
    bdPhyParms PhyParms; /* empty if not needed by PHY */
    bdCfParms  CfParms; /* empty if not CfPollable/no PCF */
    bdIbssParms IbssParms; /* empty if infrastructure BSS */
    bdCap      Capability; /* capability information */
    bdBrates   RateSet; /* BSS basic rate set */
endnewtype BssDscr;
```

Also: Use the value of “(now - ybd!bdStartTs)” as the offset for setting the TSF in the task symbol which performs the “call TSF” in the second column from the left on page C-73 of the formal description.

**Reason:** A local time reference is needed to use the timestamp returned by MLME-SCAN.indication to synchronize with the BSS when processing the subsequent MLME-JOIN.request. The time reference is included in each BSSDescription because the interval between receipt of the probe response or beacon on which the BSSDescription is based and the generation of the MLME-SCAN.indication is variable and is dependent on factors which cannot be inferred by the SME.
15. **Clause 10.3.2.2, second table:**

change as follows:

| BSSBasicRateSet | set of integers | 210 through 127255 inclusive (for each integer in the set) | The set of data rates (in units of 500 kbit/s) that must be supported by all STAs that desire to join this BSS. The STAs must be able to receive at each of the data rates listed in the set. |

Reason: to reflect changes made to the range of datarates.

16. **Clause 10.3.3.1, table:**

Change as follows:

| OperationalRateSet | set of integers | 240 through 127255 inclusive (for each integer in the set) | The set of data rates (in units of 500 kbit/s) that the STA desires to use for communication within the BSS. The STA must be able to receive at each of the data rates listed in the set. The OperationalRateSet is a superset of the BSSBasicRateSet advertised by the BSS. |

Reason: to reflect changes made to the range of datarates.

17. **Clause 10.3.10.1, table:**

| BSSBasicRateSet | set of integers | 240 through 127255 inclusive (for each integer in the set) | The set of data rates (in units of 500 kbit/s) that must be supported by all STAs that desire to join this BSS. The STA that is creating the BSS must be able to receive at each of the data rates listed in the set. |

| OperationalRateSet | set of integers | 240 through 127255 inclusive (for each integer in the set) | The set of data rates (in units of 500 kbit/s) that the STA desires to use for communication within the BSS. The STA must be able to receive at each of the data rates listed in the set. The OperationalRateSet is a superset of the BSSBasicRateSet advertised by the BSS. |

Reason: to reflect changes made to the range of datarates.

---

Copyright © 1997 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.
18. **Clause 11.4.4.2.15 should read:**

RTSThreshold ATTRIBUTE
WITH APPROPRIATE SYNTAX
integer;

BEHAVIOUR DEFINED AS
"This attribute shall indicate the number of bytes in an MPDU, below which an RTS/CTS handshake shall not be performed. An RTS/CTS handshake shall be performed for at the beginning of any frame exchange sequence where the MPDU is of type Data or MMPDU is of type Management, the MPDU or MMPDU has an individual address in the Address1 field, and all frames where the length of the MPDU or MMPDU is equal to or larger than this threshold. (For additional details, refer to Table 21 in Clause 9.7.) Setting this attribute to be larger than the maximum MSDU size shall have the effect of turning off the RTS/CTS handshake for frames of Data or Management type transmitted by this station. Setting this attribute to zero shall have the effect of turning on the RTS/CTS handshake for all frames of Data or Management type transmitted by this station. The default value of this attribute shall be 3000.";

REGISTERED AS
{ iso(1) member-body(2) us(840) ieee802dot11(10036) MAC(2) attribute(7) RTSThreshold(15) };

Also: In the formal description, add a term to the conditional test at the bottom of the first column on page C-91 to cover the case where transmission of a fragmented MSDU is resumed after a dwell boundary. The logic on C-91 already handles correctly the normal case where the fragment burst is not interrupted.

Reason: Clarify the conditions under which the RTSThreshold is used to be consistent with Clause 9.

19. **Clause 13.1.4.23**

Change to read:
"The transmit bit rates supported by the PLCP and PMD, represented by a count from 00h - 7Fh, corresponding to data rates in increments of 500 kbit/s from 0 to 63.5 Mbit/s subject to limitations of each individual PHY.";

Reason: correct unit names and to reflect changes made to the range of datarates.

20. **Clause 13.1.4.24:**

Change to read:
"The receive bit rates supported by the PLCP and PMD, represented by a count from 00h - 7Fh, corresponding to data rates in increments of 500 kbit/s from 0 to 63.5 Mbit/s.";

Reason: correct unit names and to reflect changes made to the range of datarates.

21. **Clause 13.1.4.40:**

Change nanoseconds into microseconds.

Clause 15.3.1. and 15.3.2:

22. **Clause 15.3.3:**

Update table as follows:

<table>
<thead>
<tr>
<th>Managed Object</th>
<th>Default Value / Range</th>
<th>Operational Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>agPhyOperationGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aPHYType</td>
<td>DSSS-2.4 (02)</td>
<td>Static</td>
</tr>
<tr>
<td>aTempType</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aCWmin</td>
<td>31</td>
<td>Static</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Type</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------------</td>
<td>------------</td>
</tr>
<tr>
<td>aCWmax</td>
<td>1023</td>
<td>Static</td>
</tr>
<tr>
<td>aRegDomainsSupported</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aCurrentRegDomain</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aSlotTime</td>
<td>20 µs</td>
<td>Static</td>
</tr>
<tr>
<td>aCCATime</td>
<td>≤ 15 µs</td>
<td>Static</td>
</tr>
<tr>
<td>aRxTxTurnaroundTime</td>
<td>≤ 5 µs</td>
<td>Static</td>
</tr>
<tr>
<td>aTxPLCPDelay</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aRxTxSwitchTime</td>
<td>≤ 5 µs</td>
<td>Static</td>
</tr>
<tr>
<td>aTXRampOnTime</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aTXRFDelay</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aSIFSTime</td>
<td>10 µs</td>
<td>Static</td>
</tr>
<tr>
<td>aMACProcessingDelay</td>
<td>not applicable</td>
<td>n/a</td>
</tr>
<tr>
<td>aRXRampOffTime</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aPreambleLength</td>
<td>144 bits</td>
<td>Static</td>
</tr>
<tr>
<td>aPLCPHeaderLength</td>
<td>48 bits</td>
<td>Static</td>
</tr>
<tr>
<td>agPhyRateGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aSupportedDataRatesRx</td>
<td>02h, 04h</td>
<td>Static</td>
</tr>
<tr>
<td>aSupportedDataRatesRx</td>
<td>02h, 04h</td>
<td>Static</td>
</tr>
<tr>
<td>aMPDUMaxLength</td>
<td>4 ≤ x ≤ (2^13 - 1)</td>
<td>Static</td>
</tr>
<tr>
<td>agPhyAntennaGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aCurrentTxAntenna</td>
<td>implementation dependent</td>
<td>Dynamic</td>
</tr>
<tr>
<td>aDiversitySupport</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>agPhyTxPowerGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aNumberSupportedPowerLevels</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aTXPowerLevel1</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aTXPowerLevel2</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aTXPowerLevel3</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aTXPowerLevel4</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aTXPowerLevel5</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aTXPowerLevel6</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aTXPowerLevel7</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aTXPowerLevel8</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aCurrentTxPowerLevel</td>
<td>implementation dependent</td>
<td>Dynamic</td>
</tr>
<tr>
<td>agPhyStatusGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aSynthesizerLocked</td>
<td>implementation dependent</td>
<td>Dynamic</td>
</tr>
<tr>
<td>agPhyDSSSSGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aCurrentChannel</td>
<td>implementation dependent</td>
<td>Dynamic</td>
</tr>
<tr>
<td>aCCAModeSupport</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aCurrentCCAMode</td>
<td>implementation dependent</td>
<td>Dynamic</td>
</tr>
<tr>
<td>aEDThreshold</td>
<td>implementation dependent</td>
<td>Dynamic</td>
</tr>
<tr>
<td>agPhyPwrSavingGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aDozeTurnonTime</td>
<td>implementation dependent</td>
<td>Static</td>
</tr>
<tr>
<td>aCurrentPowerState</td>
<td>implementation dependent</td>
<td>Dynamic</td>
</tr>
</tbody>
</table>
Reason: moved NOT GROUPED group to PHYOperations Group.

23. **Annex C, page C-15:**
See under 14. above.

24. **Annex C, page C-27, the definition of nullKey and PrngKey (for WEP) should be:**

```c
/*******************************************************************
* WEP support sorts
*******************************************************************/
syntype KeyIndex = Integer    constants 0:3    endsyntype KeyIndex;
synonym nullKey Octetstring = O3 // O2;
newtypesyntype PrngKey   inherits= Octetstring__operators_all;
    adding literals nullKey;
    axioms nullKey == null;
    default O3 // O2;
endsyntype PrngKey;
```

Reason: nullKey was incorrectly defined to be a 5-octet string with all octets set to zero. In fact, all zeros is a valid PrngKey value. Defining nullKey as a literal of sort PrngKey having value “null” makes nullKey an uninitialized Octetstring rather than having any of the $2^{40}$ possible values for a valid 5-octet string. As a result, the conditional tests for various keys being equal to nullKey in the left center of pages C-85 (encrypt) and C-104 (decrypt) are consistent with the operation described in clause 8.3.2.