



Identification cards — Integrated circuit card programming interfaces —

Part 4: Application programming interface (API) administration

TECHNICAL CORRIGENDUM 1

Cartes d'identification — Interfaces programmables de cartes à puce —

Partie 4: Administration d'interface de programmation (API)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 24727-4:2008 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*

Page 55, B.1

Replace lines 1–11 with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
  version="1.1">

  <import namespace="urn:oasis:names:tc:dss:1.0:core:schema"
    schemaLocation="oasis-dss-core-schema-v1.0-os.xsd"></import>
```

```
<!--
-- Version 1.1, 10-Jan-2011
--
-- © ISO/IEC 2008-2011
-- All rights reserved. Unless otherwise specified, no part of this
-- publication may be reproduced or utilized in any form or by any means,
-- electronic or mechanical, including photocopying and microfilm,
-- without permission in writing from 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
-->

<element name="COMMONSchemaVersion">
  <complexType attribute name="schemaVersion" type="decimal"
    use="required"/>
</element>

<!-- Definition of Basic Types -->

<simpleType name="SlotHandleType">
```

Page 56, B.2

Replace lines 1–10 with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
  version="1.1">

<!--
-- Version 1.1, 10-Jan-2011
--
-- © ISO/IEC 2008-2011
-- All rights reserved. Unless otherwise specified, no part of this
-- publication may be reproduced or utilized in any form or by any means,
-- electronic or mechanical, including photocopying and microfilm,
-- without permission in writing from 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
-->

<!-- Definition of Basic Types -->

<include schemaLocation="ISOCommon.xsd"></include>
```

```

<element name="IFDSchemaVersion">
  <complexType attribute name="schemaVersion" type="decimal"
    use="required"/>
</element>

<!-- Card terminal related functions -->
...

```

Page 68, B.3

Replace lines 1–6 with the following:

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
  version="1.1">

<!--
-- Version 1.1, 10-Jan-2011
--
-- © ISO/IEC 2008-2011
-- All rights reserved. Unless otherwise specified, no part of this
-- publication may be reproduced or utilized in any form or by any means,
-- electronic or mechanical, including photocopying and microfilm,
-- without permission in writing from 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
-->

...

```

Page 78, C.1

Replace lines 1–6 with the following:

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
  version="1.1">

<!--
-- Version 1.1, 10-Jan-2011
--
-- © ISO/IEC 2008-2011
-- All rights reserved. Unless otherwise specified, no part of this
-- publication may be reproduced or utilized in any form or by any means,
-- electronic or mechanical, including photocopying and microfilm,

```

```
-- without permission in writing from 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
-->
```

```
<element name="IFDCallbackSchemaVersion">
  <complexType attribute name="schemaVersion" type="decimal"
    use="required"/>
</element>
```

```
<!-- Definition of Basic Types -->
```

...

Page 78, C.2

Replace lines 1–6 with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:iso="urn:iso:std:iso-iec:24727:tech:schema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  version="1.1">
```

```
<!--
-- Version 1.1, 10-Jan-2011
--
-- © ISO/IEC 2008-2011
-- All rights reserved. Unless otherwise specified, no part of this
-- publication may be reproduced or utilized in any form or by any means,
-- electronic or mechanical, including photocopying and microfilm,
-- without permission in writing from 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
-->
```

...

Page 80

Insert the following new annexes after Annex C:

Annex D (normative)

ISO24727-4-IFDAPI module

```

ISO24727-4-IFDAPI { iso(1) standard(0) iso24727(24727) part4(4) ifdapi (74) }
-- Version 1.5, 24-May-2010
--
-- IF-PROFILE value '01'
--
-- *According to ISO/IEC 24727-2, the optional IF-PROFILE field in the CCD is
-- used to indicate that a card provides an implementation of ISO/IEC 24727-3.

-- © ISO/IEC 2008-2010
-- All rights reserved. Unless otherwise specified, no part of this publication
-- may be reproduced or utilized in any form or by any means, electronic or
-- mechanical, including photocopying and microfilm, without permission in
-- writing from either ISO at the address below or ISO's member body in the
-- country of the requester.
--
--      ISO copyright office
--      Case postale 56 i 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

DEFINITIONS AUTOMATIC TAGS EXTENSIBILITY IMPLIED ::=
BEGIN
-- EXPORTS (all)
IMPORTS NonNegativeInt, PositiveInt, GenericHandleType, IFDName, IFDAction,
        URIType, IFDSessionIdentifier, TransactionIdentifier
        FROM ISO24727-COMMON { iso(1) standard(0) iso24727(24727) };

-- Major and Minor Revision values for this ASN.1 Module
revMajISO24727-4-IFDAPI INTEGER ::= 1
revMinISO24727-4-IFDAPI INTEGER ::= 5

ChannelHandleType ::= SEQUENCE {

        protocolTerminationPoint URIType                OPTIONAL,
        sessionIdentifier          GenericIdentifierType  OPTIONAL,
        binding                     URIType                OPTIONAL
}

IFDNameList          ::= SEQUENCE OF IFDName
IFDContextHandle ::= GenericHandleType

IFDSlotHandle        ::= GenericHandleType
IFDCallbackChannelHandle ::= GenericHandleType
IFDPadChar ::= NULL
IFDDateTime ::= UTCTime

```

```

IFDCapabilities ::= SEQUENCE {
    slotCapability          SEQUENCE (SIZE (1..1000)) OF IFDSlotCapability,
    displayCapability SEQUENCE (SIZE (1..1000)) OF IFDDisplayCapability,
    keypadCapability SEQUENCE (SIZE (1..1000)) OF IFDKeypadCapability,
    bioSensorCapability SEQUENCE (SIZE (1..1000)) OF IFDBioSensorCapability
    opticalSignalUnit BOOLEAN
    acousticSignalUnit     BOOLEAN
}

IFDInputUnit ::= CHOICE {
    pinInput          IFDPINInput,
    biometricInput    IFDBiometricInput
}

IFDPINInput ::= SEQUENCE {
    index                NonNegativeInt,
    passwordAttributes   IFDPassWordAttributes
}

IFDPassWordAttributes ::= SEQUENCE {
    pwdFlags              IFDPassWordFlags,
    pwdType                IFDPassWordType,
    minLength              NonNegativeInt,
    storedLength           NonNegativeInt,
    maxLength              NonNegativeInt OPTIONAL,
    padChar                IFDPadChar OPTIONAL,
    lastPasswordChange     IFDDateTime OPTIONAL
}

IFDPassWordFlags ::= BIT STRING {
    case-sensitive(0),
    local(1),
    change-disabled(2),
    unblock-disabled(3),
    initialized(4),
    needs-padding(5),
    unblockingPassword(6),
    soPassword(7),
    disable-allowed(8),
    integrity-protected(9),
    confidentiality-protected(10),
    exchangeRefData(11),
    resetRetryCounter1(12),
    resetRetryCounter2(13)
}

IFDPassWordType ::= CHOICE {
    bcd NULL,
    ascii-numeric NULL,
    utf8 NULL,
    half-nibble-bcd NULL,
    iso9564-1 NULL
}

IFDAltVUMessages ::= SEQUENCE {
    authenticationRequestMessage OCTET STRING OPTIONAL,
    successMessage                OCTET STRING OPTIONAL,
    authenticationFailedMessage   OCTET STRING OPTIONAL,
    requestConfirmationMessage     OCTET STRING OPTIONAL,
    cancelMessage                 OCTET STRING OPTIONAL
}

```

```

IFDBiometricInput ::= SEQUENCE {
    index                NonNegativeInt,
    biometricSubType     NonNegativeInt
}

IFDSlotCapability ::= SEQUENCE {
    index                NonNegativeInt,
    contactBased         BOOLEAN
}

IFDSlotStatus ::= SEQUENCE {
    index                NonNegativeInt,
    cardAvailable        BOOLEAN,
    aTRorATS             OCTET STRING OPTIONAL
}
IFDSlotStatusList ::= SEQUENCE OF IFDSlotStatus

IFDDisplayCapability ::= SEQUENCE {
    index                NonNegativeInt,
    lines                NonNegativeInt,
    columns              NonNegativeInt,
    virtualLines         NonNegativeInt OPTIONAL,
    virtualColumns       NonNegativeInt OPTIONAL
}

IFDKeypadCapability ::= SEQUENCE {
    index NonNegativeInt,
    numKeys              PositiveInt
}

IFDBioSensorCapability ::= SEQUENCE {
    index                NonNegativeInt,
    biometricType        NonNegativeInt
}

IFDStatus ::= SEQUENCE {
    iFDName              IFDName,
    connected             BOOLEAN OPTIONAL,
    slotStatus            IFDSlotStatusList,
    activeAntenna         BOOLEAN OPTIONAL,
    displayStatus         IFDSimpleFUStatusList,
    keypadStatus          IFDSimpleFUStatusList,
    bioSensorStatus       IFDSimpleFUStatusList
}
IFDStatusList ::= SEQUENCE OF IFDStatus

IFDSimpleFUStatus ::= SEQUENCE {
    index                NonNegativeInt,
    available             BOOLEAN
}
IFDSimpleFUStatusList ::= SEQUENCE OF IFDSimpleFUStatus

IFDOutputInfo ::= SEQUENCE {
    timeout               PositiveInt OPTIONAL,
    displayIndex          NonNegativeInt OPTIONAL,
    message               VisibleString OPTIONAL,
    acousticSignal        BOOLEAN OPTIONAL,
    opticalSignal         BOOLEAN OPTIONAL
}

```

```

-- 7.4.1 IFD_API_EstablishContext
IFDAPIEstablishContextArgument ::= SEQUENCE {

    channelHandle ChannelHandleType OPTIONAL
}
IFDAPIEstablishContextResult ::= SEQUENCE {

    contextHandle IFDContextHandle OPTIONAL
}

IFDAPIEstablishContextReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CHANNEL_HANDLE"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIEstablishContext ::= SEQUENCE {

    arugment IFDAPIEstablishContextArgument,
    result    IFDAPIEstablishContextResult    OPTIONAL,
    return    IFDAPIEstablishContextReturnCode
}

IFDAPIEstablishContextCall ::= [APPLICATION 4022] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              IFDAPIEstablishContextArgument
}

IFDAPIEstablishContextReturn ::= [APPLICATION 4023] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    result                IFDAPIEstablishContextResult OPTIONAL,
    returnCode            IFDAPIEstablishContextReturnCode
}

-- 7.4.2 IFD_API_ReleaseContext
IFDAPIReleaseContextArgument ::= SEQUENCE {

    contextHandle IFDContextHandle
}
IFDAPIReleaseContextReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CHANNEL_HANDLE"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIReleaseContext ::= SEQUENCE {

    arugment IFDAPIReleaseContextArgument,
    result    NULL                                OPTIONAL,
    return    IFDAPIReleaseContextReturnCode
}

IFDAPIReleaseContextCall ::= [APPLICATION 4027] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              IFDAPIReleaseContextArgument
}

```



```

IFDAPIReleaseContextReturn ::= [APPLICATION 4028] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    returnCode             IFDAPIReleaseContextReturnCode

}

-- 7.4.3 IFD_API_ListIFDs
IFDAPIListIFDsArgument ::= SEQUENCE {

    contextHandle IFDContextHandle

}
IFDAPIListIFDsResult ::= SEQUENCE {

    iFDNameList IFDNameList

}

IFDAPIListIFDsReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CONTEXT_HANDLE"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIListIFDs ::= SEQUENCE {

    arugment IFDAPIListIFDsArgument,
    result   IFDAPIListIFDsResult   OPTIONAL,
    return   IFDAPIListIFDsReturnCode

}

IFDAPIListIFDsCall ::= [APPLICATION 4032] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              IFDAPIListIFDsArgument

}
IFDAPIListIFDsReturn ::= [APPLICATION 4033] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    result                IFDAPIListIFDsResult OPTIONAL,
    returnCode            IFDAPIListIFDsReturnCode

}

-- 7.4.4 IFD_API_GetIFDCapabilities
IFDAPIGetIFDCapabilitiesArgument ::= SEQUENCE {

    contextHandle IFDContextHandle,
    iFDName       IFDName

}

IFDAPIGetIFDCapabilitiesResult ::= SEQUENCE {

    iFDCapabilities IFDCapabilities

}

IFDAPIGetIFDCapabilitiesReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CONTEXT_HANDLE"
-- "IFD_UNKNOWN_IFD"

```

```

-- "IFD_UNKNOWN_ERROR"
}))

IFDAPIGetIFDCapabilities ::= SEQUENCE {

    arugment IFDAPIGetIFDCapabilitiesArgument,
    result    IFDAPIGetIFDCapabilitiesResult    OPTIONAL,
    return    IFDAPIGetIFDCapabilitiesReturnCode

}

IFDAPIGetIFDCapabilitiesCall ::= [APPLICATION 4037] SEQUENCE {

    transactionIdentifier TransactionIdentifier    OPTIONAL,
    argument                IFDAPIGetIFDCapabilitiesArgument

}

IFDAPIGetIFDCapabilitiesReturn ::= [APPLICATION 4038] SEQUENCE {

    transactionIdentifier TransactionIdentifier    OPTIONAL,
    result                IFDAPIListIFDsResult    OPTIONAL,
    returnCode            IFDAPIGetIFDCapabilitiesReturnCode

}

-- 7.4.5 IFD_API_GetStatus
IFDAPIGetStatusArgument ::= SEQUENCE {

    contextHandle IFDContextHandle,
    iFDName        IFDName        OPTIONAL

}

IFDAPIGetStatusResult ::= SEQUENCE {

    iFDStatus IFDStatus

}

IFDAPIGetStatusReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CONTEXT_HANDLE"
-- "IFD_UNKNOWN_IFD"
-- "IFD_UNKNOWN_ERROR"
}))

IFDAPIGetStatus ::= SEQUENCE {

    arugment IFDAPIGetStatusArgument,
    result    IFDAPIGetStatusResult    OPTIONAL,
    return    IFDAPIGetStatusReturnCode

}

IFDAPIGetStatusCall ::= [APPLICATION 4042] SEQUENCE {

    transactionIdentifier TransactionIdentifier    OPTIONAL,
    argument                IFDAPIGetStatusArgument

}

IFDAPIGetStatusReturn ::= [APPLICATION 4043] SEQUENCE {

    transactionIdentifier TransactionIdentifier    OPTIONAL,
    result                IFDAPIGetStatusResult    OPTIONAL,
    returnCode            IFDAPIGetStatusReturnCode

}

```

```

-- 7.4.6 IFD_API_Wait
IFDAPIWaitArgument ::= SEQUENCE {

    contextHandle    IFDContextHandle,
    timeOut          PositiveInt          OPTIONAL,
    callbackChannel  IFDCallbackChannelHandle OPTIONAL,
    iFDStatusList    IFDStatusList

}
IFDAPIWaitResult ::= SEQUENCE {

    sessionIdentifier VisibleString OPTIONAL,
    iFDStatusList     IFDStatusList

}

IFDAPIWaitReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CONTEXT_HANDLE"
-- "IFD_UNKNOWN_IFD"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIWait ::= SEQUENCE {

    arugment IFDAPIWaitArgument,
    result   IFDAPIWaitResult   OPTIONAL,
    return   IFDAPIWaitReturnCode

}

IFDAPIWaitCall ::= [APPLICATION 4047] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              IFDAPIWaitArgument

}
IFDAPIWaitReturn ::= [APPLICATION 4048] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    result                IFDAPIWaitResult   OPTIONAL,
    returnCode            IFDAPIWaitReturnCode

}

-- 7.4.7 IFD_API_Cancel
IFDAPICancelArgument ::= SEQUENCE {

    contextHandle    IFDContextHandle,
    sessionIdentifier IFDSessionIdentifier OPTIONAL,
    iFDName          IFDName

}

IFDAPICancelReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_CANCEL_NOT_POSSIBLE"
-- "IFD_INVALID_CONTEXT_HANDLE"
-- "IFD_UNKNOWN_IFD"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPICancel ::= SEQUENCE {

```

```

        arugment IFDAPICancelArgument,
        result    NULL                                OPTIONAL,
        return    IFDAPICancelReturnCode
    }

IFDAPICancelCall ::= [APPLICATION 4052] SEQUENCE {

        transactionIdentifier TransactionIdentifier OPTIONAL,
        argument              IFDAPICancelArgument
    }
IFDAPICancelReturn ::= [APPLICATION 4053] SEQUENCE {

        transactionIdentifier TransactionIdentifier OPTIONAL,
        returnCode            IFDAPICancelReturnCode
    }

-- 7.4.8 IFD_API_ControlIFD
IFDAPIControlIFDArgument ::= SEQUENCE {

        contextHandle IFDContextHandle,
        iFDName        IFDName,
        command         OCTET STRING
    }
IFDAPIControlIFDResult ::= SEQUENCE {

        response OCTET STRING
    }

IFDAPIControlIFDReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CONTEXT_HANDLE"
-- "IFD_UNKNOWN_IFD"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIControlIFD ::= SEQUENCE {

        arugment IFDAPIControlIFDArgument,
        result    IFDAPIControlIFDResult    OPTIONAL,
        return    IFDAPIControlIFDReturnCode
    }

IFDAPIControlIFDCall ::= [APPLICATION 4057] SEQUENCE {

        transactionIdentifier TransactionIdentifier    OPTIONAL,
        argument              IFDAPIControlIFDArgument
    }
IFDAPIControlIFDReturn ::= [APPLICATION 4058] SEQUENCE {

        transactionIdentifier TransactionIdentifier    OPTIONAL,
        result                IFDAPIControlIFDResult    OPTIONAL,
        returnCode            IFDAPIControlIFDReturnCode
    }

-- 7.4.9 IFD_API_Connect
IFDAPIConnectArgument ::= SEQUENCE {
contextHandle IFDContextHandle,

```

```

        iFDName      IFDName,
        slot         NonNegativeInt,
        exclusive    BOOLEAN          OPTIONAL
    }
IFDAPIConnectResult ::= SEQUENCE {

        slotHandle IFDSlotHandle

    }

IFDAPIConnectReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CONTEXT_HANDLE"
-- "IFD_UNKNOWN_IFD"
-- "IFD_UNKNOWN_SLOT"
-- "IFD_SHARING_VIOLATION"
-- "IFD_NO_CARD"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIConnect ::= SEQUENCE {

        arugment IFDAPIConnectArgument,
        result   IFDAPIConnectResult   OPTIONAL,
        return   IFDAPIConnectReturnCode

    }

IFDAPIConnectCall ::= [APPLICATION 4062] SEQUENCE {

        transactionIdentifier TransactionIdentifier OPTIONAL,
        argument              IFDAPIConnectArgument

    }
IFDAPIConnectReturn ::= [APPLICATION 4063] SEQUENCE {

        transactionIdentifier TransactionIdentifier OPTIONAL,
        result               IFDAPIConnectResult   OPTIONAL,
        returnCode           IFDAPIConnectReturnCode

    }

-- 7.4.10 IFD_API_Disconnect
IFDAPIDisconnectArgument ::= SEQUENCE {

        slotHandle IFDSlotHandle,
        action      IFDAAction          OPTIONAL

    }

IFDAPIDisconnectReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_SLOT_HANDLE"
-- "IFD_UNKNOWN_ACTION"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIDisconnect ::= SEQUENCE {

        arugment IFDAPIDisconnectArgument,
        result   NULL                                OPTIONAL,
        return   IFDAPIDisconnectReturnCode

    }

```

```

IFDAPIDisconnectCall ::= [APPLICATION 4067] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument               IFDAPIDisconnectArgument

}

IFDAPIDisconnectReturn ::= [APPLICATION 4068] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    returnCode            IFDAPIDisconnectReturnCode

}

-- 7.4.11 IFD_API_BeginTransaction
IFDAPIBeginTransactionArgument ::= SEQUENCE {

    slotHandle IFDSlotHandle

}

IFDAPIBeginTransactionReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_SLOT_HANDLE"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIBeginTransaction ::= SEQUENCE {

    arugment IFDAPIBeginTransactionArgument,
    result    NULL                                OPTIONAL,
    return    IFDAPIBeginTransactionReturnCode

}

IFDAPIBeginTransactionCall ::= [APPLICATION 4072] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument               IFDAPIBeginTransactionArgument

}

IFDAPIBeginTransactionReturn ::= [APPLICATION 4073] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    returnCode            IFDAPIBeginTransactionReturnCode

}

-- 7.4.12 IFD_API_EndTransaction
IFDAPIEndTransactionArgument ::= SEQUENCE {

    slotHandle IFDSlotHandle

}

IFDAPIEndTransactionReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_SLOT_HANDLE"
-- "IFD_NO_TRANSACTION_STARTED"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIEndTransaction ::= SEQUENCE {

```

```

        arugment IFDAPIEndTransactionArgument,
        result    NULL                                OPTIONAL,
        return    IFDAPIEndTransactionReturnCode
    }

IFDAPIEndTransactionCall ::= [APPLICATION 4077] SEQUENCE {

        transactionIdentifier TransactionIdentifier    OPTIONAL,
        argument              IFDAPIEndTransactionArgument
    }
IFDAPIEndTransactionReturn ::= [APPLICATION 4078] SEQUENCE {

        transactionIdentifier TransactionIdentifier    OPTIONAL,
        returnCode            IFDAPIEndTransactionReturnCode
    }

-- 7.4.13 IFD_API_Transmit
IFDAPITransmitArgument ::= SEQUENCE {

        slotHandle IFDSlotHandle,
        inputAPDU  OCTET STRING
    }
IFDAPITransmitResult ::= SEQUENCE {

        outputAPDU OCTET STRING
    }

IFDAPITransmitReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_SLOT_HANDLE"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPITransmit ::= SEQUENCE {

        arugment IFDAPITransmitArgument,
        result    IFDAPITransmitResult    OPTIONAL,
        return    IFDAPITransmitReturnCode
    }

IFDAPITransmitCall ::= [APPLICATION 4082] SEQUENCE {

        transactionIdentifier TransactionIdentifier    OPTIONAL,
        argument              IFDAPITransmitArgument
    }
IFDAPITransmitReturn ::= [APPLICATION 4083] SEQUENCE {

        transactionIdentifier TransactionIdentifier    OPTIONAL,
        result                IFDAPITransmitResult    OPTIONAL,
        returnCode            IFDAPITransmitReturnCode
    }

-- 7.4.14 IFD_API_VerifyUser
IFDAPIVerifyUserArgument ::= SEQUENCE {

        slotHandle      IFDSlotHandle,
        inputUnit        IFDInputUnit,
        displayIndex     NonNegativeInt    OPTIONAL,
        altVUMessages    IFDAltVUMessages OPTIONAL,

```

```

        timeoutUntilFirstKey PositiveInt      OPTIONAL,
        timeoutAfterFirstKey PositiveInt      OPTIONAL,
        template              OCTET STRING
    }
IFDAPIVerifyUserResult ::= SEQUENCE {

    response OCTET STRING
}

IFDAPIVerifyUserReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_SLOT_HANDLE"
-- "IFD_UNKNOWN_INPUT_UNIT"
-- "IFD_CANCELLATION_BY_USER"
-- "IFD_UNKNOWN_ERROR"
-- "IFD_UNKNOWN_PIN_FORMAT"
-- "IFD_UNKNOWN_BIOMETRIC_SUBTYPE"
})

IFDAPIVerifyUser ::= SEQUENCE {

    arugment IFDAPIVerifyUserArgument,
    result   IFDAPIVerifyUserResult   OPTIONAL,
    return   IFDAPIVerifyUserReturnCode
}

IFDAPIVerifyUserCall ::= [APPLICATION 4087] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              IFDAPIVerifyUserArgument
}

IFDAPIVerifyUserReturn ::= [APPLICATION 4088] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    result               IFDAPIVerifyUserResult OPTIONAL,
    returnCode           IFDAPIVerifyUserReturnCode
}

-- 7.4.15 IFD_API_ModifyVerificationData
IFDAPIModifyVerificationDataArgument ::= SEQUENCE {

    slotHandle      IFDSlotHandle,
    inputUnit       IFDInputUnit,
    displayIndex    NonNegativeInt OPTIONAL,
    altVUMessages   IFDAltVUMessages OPTIONAL,
    oldReferenceData OCTET STRING   OPTIONAL,
    timeoutUntilFirstKey PositiveInt OPTIONAL,
    timeoutAfterFirstKey PositiveInt OPTIONAL,
    repeatInput     BOOLEAN         OPTIONAL,
    template        OCTET STRING
}

IFDAPIModifyVerificationDataReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_SLOT_HANDLE"
-- "IFD_UNKNOWN_INPUT_UNIT"
-- "IFD_CANCELLATION_BY_USER"
-- "IFD_REPEATED_DATA_MISMATCH"

```



```

-- "IFD_UNKNOWN_PIN_FORMAT"
-- "IFD_UNKNOWN_BIOMETRIC_SUBTYPE"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIModifyVerificationData ::= SEQUENCE {

    arugment IFDAPIModifyVerificationDataArgument,
    result    NULL                                OPTIONAL,
    return    IFDAPIModifyVerificationDataReturnCode

}

IFDAPIModifyVerificationDataCall ::= [APPLICATION 4092] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              IFDAPIModifyVerificationDataArgument

}
IFDAPIVerifyUserModifyVerificationDataReturn ::= [APPLICATION 4093] SEQUENCE {
    transactionIdentifier TransactionIdentifier OPTIONAL,
    returnCode            IFDAPIModifyVerificationDataReturnCode
}

-- 7.4.16 IFD_API_Output
IFDAPIOutputArgument ::= SEQUENCE {

    contextHandle IFDContextHandle,
    iFDName       IFDName,
    outputInfo    IFDOutputInfo

}

IFDAPIOutput ::= SEQUENCE {

    arugment IFDAPIOutputArgument,
    result    NULL                                OPTIONAL,
    return    IFDAPIOutputReturnCode

}

IFDAPIOutputReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CONTEXT_HANDLE"
-- "IFD_UNKNOWN_IFD"
-- "IFD_UNKNOWN_DISPLAY_INDEX"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPIOutputCall ::= [APPLICATION 4097] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              IFDAPIOutputArgument

}
IFDAPIOutputReturn ::= [APPLICATION 4098] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    returnCode            IFDAPIOutputReturnCode

}

-- 7.4.17 IFD_API_SignalEvent
IFDAPISignalEventArgument ::= SEQUENCE {

```

```

        contextHandle IFDContextHandle,
        sessionID     IFDSessionIdentifier OPTIONAL,
        iFDEvent       IFDStatusList
    }

IFDAPISignalEvent ::= SEQUENCE {

        arugment IFDAPISignalEventArgument,
        result    NULL                                OPTIONAL,
        return     IFDAPISignalEventReturnCode
    }

IFDAPISignalEventReturnCode ::= VisibleString (CONSTRAINED BY {
-- "IFD_OK"
-- "IFD_TIMEOUT_ERROR"
-- "IFD_INVALID_CONTEXT_HANDLE"
-- "IFD_UNKNOWN_IFD"
-- "IFD_UNKNOWN_ERROR"
})

IFDAPISignalEventCall ::= [APPLICATION 4102] SEQUENCE {

        transactionIdentifier TransactionIdentifier    OPTIONAL,
        argument              IFDAPISignalEventArgument
    }
IFDAPISignalEventReturn ::= [APPLICATION 4103] SEQUENCE {

        transactionIdentifier TransactionIdentifier    OPTIONAL,
        returnCode            IFDAPISignalEventReturnCode
    }

END

```

Annex E (normative)

ISO24727-4-TCAPI module

```

ISO24727-4-TCAPI { iso(1) standard(0) iso24727(24727) part4(4) tcapi (73) }
-- Version 1.4, 24-May-2010
--
-- IF-PROFILE value '01'
--
-- *According to ISO/IEC 24727-2, the optional IF-PROFILE field in the CCD is
-- used to indicate that a card provides an implementation of ISO/IEC 24727-3.

-- © ISO/IEC 2008-2010
-- All rights reserved. Unless otherwise specified, no part of this publication
-- may be reproduced or utilized in any form or by any means, electronic or
-- mechanical, including photocopying and microfilm, without permission in
-- writing from either ISO at the address below or ISO's member body in the
-- country of the requester.
--
--     ISO copyright office
--     Case postale 56 î 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

DEFINITIONS AUTOMATIC TAGS EXTENSIBILITY IMPLIED ::=
BEGIN
-- EXPORTS (all)
IMPORTS URIType, IFDSessionIdentifier, TransactionIdentifier,
        GenericIdentifierType
        FROM ISO24727-COMMON { iso(1) standard(0) iso24727(24727) };

-- Major and Minor Revision values for this ASN.1 Module
revMajISO24727-4-TCAPI INTEGER ::= 1
revMinISO24727-4-TCAPI INTEGER ::= 4

ChannelHandleType ::= SEQUENCE {

    protocolTerminationPoint URIType                OPTIONAL,
    sessionIdentifier         GenericIdentifierType   OPTIONAL,
    binding                   URIType                 OPTIONAL
}

-- 7.3.1 TC_API_Open
TCAPIOpenArgument ::= SEQUENCE {

    remoteAddress OCTET STRING,
    channelParams OCTET STRING
}
TCAPIOpenResult ::= SEQUENCE {

    channelHandle ChannelHandleType
}

```

```

TCAPIOpenReturnCode ::= VisibleString (CONSTRAINED BY {
--  "API_OK"
--  "API_TIMEOUT_ERROR"
--  "API_NODE_NOT_REACHABLE"
--  "API_UNKNOWN_ERROR"
})

TCAPIOpen ::= SEQUENCE {

    argument TCAPIOpenArgument,
    result    TCAPIOpenResult    OPTIONAL,
    return    TCAPIOpenReturnCode

}

TCAPIOpenCall ::= [APPLICATION 4003] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              TCAPIOpenArgument

}

TCAPIOpenReturn ::= [APPLICATION 4004] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    result                TCAPIOpenResult    OPTIONAL,
    returnCode            TCAPIOpenReturnCode

}

-- 7.3.2 TC_API_Close
TCAPICloseArgument ::= SEQUENCE {

    channelHandle ChannelHandleType

}

TCAPICloseReturnCode ::= VisibleString (CONSTRAINED BY {
--  "API_OK"
--  "API_TIMEOUT_ERROR"
--  "API_UNKNOWN_ERROR"
--  "API_UNKNOWN_HANDLE"
})

TCAPIClose ::= SEQUENCE {

    arugment TCAPICloseArgument,
    result    NULL                OPTIONAL,
    return    TCAPICloseReturnCode

}

TCAPICloseCall ::= [APPLICATION 4007] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              TCAPICloseArgument

}

TCAPICloseReturn ::= [APPLICATION 4008] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    returnCode            TCAPICloseReturnCode

}

-- 7.3.3 TC_API_Read
TCAPIReadArgument ::= SEQUENCE {

```

```

        channelHandle ChannelHandleType
    }
    TCAPIReadResult ::= SEQUENCE {

        message OCTET STRING
    }

    TCAPIReadReturnCode ::= VisibleString (CONSTRAINED BY {
-- "API_OK"
-- "API_TIMEOUT_ERROR"
-- "API_UNKNOWN_ERROR"
-- "API_WARNING_BUFFER_LENGTH_EXCEEDED"
-- "API_UNKNOWN_HANDLE"
    })

    TCAPIRead ::= SEQUENCE {

        arugment TCAPIReadArgument,
        result   TCAPIReadResult   OPTIONAL,
        return   TCAPIReadReturnCode
    }

    TCAPIReadCall ::= [APPLICATION 4012] SEQUENCE {

        transactionIdentifier TransactionIdentifier OPTIONAL,
        argument              TCAPIReadArgument
    }

    TCAPIReadReturn ::= [APPLICATION 4013] SEQUENCE {

        transactionIdentifier TransactionIdentifier OPTIONAL,
        result                TCAPIReadResult   OPTIONAL,
        returnCode            TCAPIReadReturnCode
    }

-- 7.3.4 TC_API_Write
    TCAPIWriteArgument ::= SEQUENCE {

        channelHandle ChannelHandleType,
        message        OCTET STRING
    }

    TCAPIWriteReturnCode ::= VisibleString (CONSTRAINED BY {
-- "API_OK"
-- "API_TIMEOUT_ERROR"
-- "API_UNKNOWN_ERROR"
-- "API_UNKNOWN_HANDLE"
    })

    TCAPIWrite ::= SEQUENCE {

        arugment TCAPIWriteArgument,
        result   NULL                                OPTIONAL,
        return   TCAPIWriteReturnCode
    }

    TCAPIWriteCall ::= [APPLICATION 4017] SEQUENCE {

        transactionIdentifier TransactionIdentifier OPTIONAL,
        argument              TCAPIWriteArgument
    }

```

```

TCAPIWriteReturn ::= [APPLICATION 4018] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    returnCode             TCAPIWriteReturnCode

}

-- 7.3.5 TC_API_Reset
TCAPIResetArgument ::= SEQUENCE {

    channelHandle ChannelHandleType

}

TCAPIResetReturnCode ::= VisibleString (CONSTRAINED BY {
-- "API_OK"
-- "API_TIMEOUT_ERROR"
-- "API_UNKNOWN_ERROR"
-- "API_UNKNOWN_HANDLE"
})

TCAPIReset ::= SEQUENCE {

    arugment TCAPIResetArgument,
    result   NULL                OPTIONAL,
    return   TCAPIResetReturnCode

}

TCAPIResetCall ::= [APPLICATION 4012] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    argument              TCAPIResetArgument

}

TCAPIResetReturn ::= [APPLICATION 4013] SEQUENCE {

    transactionIdentifier TransactionIdentifier OPTIONAL,
    returnCode            TCAPIResetReturnCode

}

-- 7.3.6 TC_API_GetStatus
TCAPIGetStatusArgument ::= SEQUENCE {

    channelHandle ChannelHandleType

}

TCAPIGetStatusResult ::= SEQUENCE {

    statusString OCTET STRING

}

TCAPIGetStatusReturnCode ::= VisibleString (CONSTRAINED BY {
-- "API_OK"
-- "API_TIMEOUT_ERROR"
-- "API_UNKNOWN_ERROR"
-- "API_UNKNOWN_HANDLE"
})

TCAPIGetStatus ::= SEQUENCE {

    arugment TCAPIGetStatusArgument,
    result   TCAPIGetStatusResult    OPTIONAL,
    return   TCAPIGetStatusReturnCode

}

```

```
TCAPIGetStatusCall ::= [APPLICATION 4017] SEQUENCE {  
  
    transactionIdentifier TransactionIdentifier OPTIONAL,  
    argument               TCAPIGetStatusArgument  
}  
TCAPIGetStatusReturn ::= [APPLICATION 4018] SEQUENCE {  
  
    transactionIdentifier TransactionIdentifier OPTIONAL,  
    result                TCAPIGetStatusResult  OPTIONAL,  
    returnCode            TCAPIGetStatusReturnCode  
}  
  
END
```