INTERNATIONAL STANDARD

IEC 62055-41

First edition
2007-05

Electricity metering – Payment systems –

Part 41: Standard transfer specification (STS) – Application layer protocol for one-way token carrier systems

Reference number
IEC 62055-41:2007(E)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

INTERNATIONAL STANDARD

IEC 62055-41

First edition
2007-05

Electricity metering – Payment systems –

Part 41: Standard transfer specification (STS) – Application layer protocol for one-way token carrier systems

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XD

For price, see current catalogue

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ELECTRICITY METERING – PAYMENT SYSTEMS –

Part 41: Standard transfer specification (STS) – Application layer protocol for one-way token carrier systems

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.

2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.

3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.

5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.

6) All users should ensure that they have the latest edition of this publication.

7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.

8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

The IEC draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning the SpecialReservedTokenIdentifier given in 6.3.5.2.

The IEC takes no position concerning the evidence, validity and scope of these patent rights. The holder of these patent rights have assured the IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patents are registered with the IEC. Information may be obtained from:

Address: Actaris Measurement and Systems, P.O. Box 4059,TygerValley 7536, Republic of South Africa

Tel: +27 21 914 3640

Fax: +27 21 914 3630

Website: http://www.actaris.com

Address: Merlin Gerin SA (Pty) Ltd t/a Conlog, P.O. Box 2332, Durban 4000, Republic of South Africa

Tel: +27 31 2681141

Fax: +27 31 2087790

Website: http://www.conlog.co.za

Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62055-41 has been prepared by IEC technical committee 13: Electrical energy measurement, tariff and load control.

This standard cancels and replaces IEC/PAS 62055-41 published in 2003. This first edition constitutes a technical revision.

The text of this standard is based on the following documents:

CDV	Report on voting
13/1405/CDV	13/1409/RVC

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

INTRODUCTION

The IEC 62055 series covers payment systems, encompassing the customer information systems, point of sale systems, token carriers, payment meters and the respective interfaces that exist between these entities. At the time of preparation of this standard, IEC 62055 comprised the following parts, under the general title, *Electricity metering – Payment systems:*

Part 21: Framework for standardization

Part 31: Particular requirements – Static payment meters for active energy (classes 1 and 2)

Part 41: Standard transfer specification – Application layer protocol for one-way token carrier systems

Part 51: Standard transfer specification – Physical layer protocol for one-way numeric and magnetic card token carriers

Part 52: Standard transfer specification – Physical layer protocol for a two-way virtual token carrier for direct local connection

The Part 4x series specifies application layer protocols and the Part 5x series specifies physical layer protocols.

The standard transfer specification (STS) is a secure message protocol that allows information to be carried between point-of-sale (POS) equipment and payment meters and it caters for several message types such as credit, configuration control, display and test instructions. It further specifies devices and codes of practice that allows for the secure management (generation, storage, retrieval and transportation) of cryptographic keys used within the system.

The national electricity utility in South Africa (Eskom) first developed and published the STS in 1993 and transferred ownership to the STS Association in 1998 for management and further development. It is currently the only open system for one-way payment meters and to date there are more than 4 million STS payment meters in the field, being used by approximately 400 utilities in 28 countries. The STS has been stable for 10 years, is the *de facto* industry standard at national and international level and hence has been developed as an IEC standard with the appropriate reformatting to comply with WG15 work. The primary application of the STS has been for use with payment meters without a tariff employing energy-based tokens, but it could be applied to currency-based token systems.

Prior to the development of the STS a variety of proprietary payment meters and POS equipment had been developed, which were, however, not compatible with each other. This gave rise to a definite need among the major users to move towards standardized solutions in addressing operational problems experienced where various types of payment meter and POS equipment had to be operated simultaneously. A standard transfer specification was developed that would allow for the application and inter-operability of payment meters and POS equipment from multiple manufacturers in a payment metering installation.

Two encryption algorithms are supported in this standard. The STA is used in existing systems, while the DEA may be considered for future systems.

The token carrier, which is not specified in this part of IEC 62055, is the physical device or medium used to transport the information from the POS equipment to the payment meter. Three types of token carriers are currently specified in IEC 62055-51 and IEC 62055-52: the magnetic card, the numeric token carrier and a virtual token carrier, which have been approved by the STS Association. New token carriers can be proposed as new work items through the National Committees or through the STS Association.

Although the main implementation of the STS is in the electricity supply industry, it inherently provides for the management of other utility services like water and gas. Future revisions of the STS may allow for other token carrier technologies like smart cards and memory keys with two-way functionality and to cater for a real-time clock and complex tariffs in the payment meter.

Not all the requirements specified in this standard are compulsory for implementation in a particular system configuration, and, as a guideline, a selection of optional configuration parameters are listed in Clause C.11.

The STS Association has established D-type liaison with working group 15 of IEC TC 13 for the development of standards within the scope of the STS and is thus responsible for the maintenance of any such IEC standards that might be developed as a result of this liaison.

- The STS Association is also registered with the IEC as a Registration Authority for providing maintenance services in support of the STS (see Clause C.1 for more information).

ELECTRICITY METERING – PAYMENT SYSTEMS –

Part 41: Standard transfer specification (STS) – Application layer protocol for one-way token carrier systems

1 Scope

This part of IEC 62055 specifies the application layer protocol of the STS for transferring units of credit and other management information from a point-of-sale (POS) system to an STS-compliant payment meter in a one-way token carrier system. It is primarily intended for application with electricity payment meters without a tariff employing energy-based tokens, but may also have application with currency-based token systems and for services other than electricity.

It specifies

- a POSToTokenCarrierInterface structured with an application layer protocol and a physical layer protocol using the OSI model as reference;
- tokens for the application layer protocol to transfer the various messages from the POS to the payment meter;
- security functions and processes in the application layer protocol such as the Standard Transfer Algorithm and the Data Encryption Algorithm, including the generation and distribution of the associated cryptographic keys;
- security functions and processes in the application layer protocol at the payment meter such as decryption algorithms, token authentication, validation and cancellation;
- specific requirements for the MeterApplicationProcess in response to tokens received;
- a scheme for dealing with payment meter functionality in the MeterApplicationProcess and associated companion specifications;
- generic requirements for an STS-compliant KeyManagementSystem;
- guidelines for a KeyManagementSystem;
- entities and identifiers used in an STS system;
- a code of practice and maintenance support services from the STS Association.

It is intended for use by manufacturers of payment meters that have to accept tokens that comply with the STS and also by manufacturers of POS systems that have to produce STS-compliant tokens and is to be read in conjunction with IEC 62055-5x series.

NOTE 1 Although developed for payment systems for electricity, the standard also makes provision for tokens used in other utility services, such as water and gas.

NOTE 2 STS-compliant products are required to comply with selective parts of this International Standard only, which should be the subject of the purchase contract (see also C.11).

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 60050-300, *International Electrotechnical Vocabulary (IEV) – Electrical and electronic measurements and measuring instruments – Part 311: General terms relating to measurements – Part 312: General terms relating to electrical measurements – Part 313: Types of electrical measuring instruments – Part 314: Specific terms according to the type of instrument*

IEC 62051:1999, *Electricity metering – Glossary of terms*

IEC 62055-21:2005, *Electricity metering – Payment systems – Part 21: Framework for standardization*

IEC 62055-31:2005, *Electricity metering – Payment systems – Part 31: Particular requirements – Static payment meters for active energy (classes 1 and 2)*

IEC 62055-51, *Electricity metering – Payment systems – Part 51: Standard transfer specification – Physical layer protocol for one-way numeric and magnetic card token carriers*

IEC 62055-52, *Electricity metering – Payment systems – Part 52: Standard transfer specification – Physical layer protocol for a two-way virtual token carrier for direct local connection*[1]

ISO/IEC 7812-1:2006, *Identification cards – Identification of issuers – Part 1: Numbering system*

ISO/IEC 7812-2:2000, *Identification cards – Identification of issuers – Part 2: Application and registration procedures*

ANSI X3.92-1981, *American National Standard Data Encryption Algorithm, American National Standards Institute*

FIPS PUB 46-3:1999, *Federal Information Processing Standards Publication – Data Encryption Standard*

3 Terms, definitions and abbreviations

3.1 Terms and definitions

3.1.1 General

For the purposes of this document, the terms and definitions given in IEC 60050-300, IEC 62051, IEC 62055-31 and the following terms apply.

Where there is a difference between the definitions in this standard and those contained in other referenced IEC standards, then those defined in this standard shall take precedence.

The term “meter” is used interchangeably with “payment meter”, “prepayment meter” and “decoder”, where the decoder is a sub-part of an electricity payment meter or a multi-part payment meter.

The term “POS” is used synonymously with “CIS”, “MIS” and “HHU” in the sense that tokens may also be generated by, and transferred between these entities and the payment meter.

[1] To be published.

The term “utility” is used to signify the supplier of the service in a general sense. It should be noted that, in the liberalized markets, the actual contracting party acting as the “supplier” of the service to the consumer may not be the traditional utility as such, but may be a third service provider party.

3.1.2
companion specification
specification managed by the STS Association, which defines a specific instance of a MeterFunctionObject (see 5.5 and Clause C.8)

3.1.3
decoder
part of the TokenCarrierToMeterInterface of a payment meter that performs the functions of the application layer protocol and which allows token-based transactions to take place between a POS and the payment meter

3.1.4
meter serial number
number that is associated with the metrological part of the payment meter

NOTE In a single-part payment meter the DRN and meter serial number may be synonymous, while in a multi-part payment meter they may be different.

3.1.5
token
subset of data elements, containing an instruction and information that is present in the APDU of the Application Layer of the POSToTokenCarrierInterface, and which is also transferred to the payment meter by means of a token carrier (the converse is also true in the case of a token being sent from the payment meter to the POS)

3.1.6
token carrier
medium that is used in the Physical Layer of the POSToTokenCarrierInterface, onto which a token is modulated or encoded, and which serves to carry a token from the point where it is generated to the remote payment meter, where it is received

3.1.7
one-way token carrier system
payment metering system, which employs token carriers that transfer information in one direction only – from the POS to the payment meter

3.1.8
token-based transaction
processing of any token by the payment meter that has material effect on the amount, value or quality of service to be delivered to the consumer under control of the payment meter (in terms of current practice this means tokens of Class 0 and Class 2)

3.2 Abbreviations

ANSI	American National Standards Institute
APDU	ApplicationProtocolDataUnit
CA	CertificationAuthority
CC	CountryCode
CIS	Customer Information System
CM	CryptographicModule

CMAC	CryptographicModuleAuthenticationCode
CMID	CryptographicModuleIdentifier
COP	Code of practice
CRC	CyclicRedundancyCode
DAC	DeviceAuthenticationCode
DCTK	DecoderCommonTransferKey
DD	Discretionary Data
DDTK	DecoderDefaultTransferKey
DEA	Data Encryption Algorithm
DES	Data Encryption Standard
DITK	DecoderInitializationTransferKey
DK	DecoderKey
DKGA	DecoderKeyGenerationAlgorithm
DKR	DecoderKeyRegister
DOE	DateOfExpiry
DRN	DecoderReferenceNumber [known as a "meter number" in systems in use prior to the development of this standard]
DSN	DecoderSerialNumber
DUTK	DecoderUniqueTransferKey
EA	EncryptionAlgorithm
ECB	Electronic Code Book
ETX	ASCII End of Text character
FAC	FirmwareAuthenticationCode
FIPS	Federal Information Processing Standards
FOIN	FunctionObjectIdentificationNumber
FS	FieldSeparator
GPRS	General Packet Radio Service
GSM	Global System For Mobile Communications
HHU	HandHeldUnit
IAIN	IndividualAccountIdentificationNumber
ID	Identification; Identifier
IIN	IssuerIdentificationNumber
ISDN	Integrated Services Digital Network
ISO	International Standards Organization
ISO BIN	Replaced by IIN
KCT	KeyChangeToken
KEK	KeyExchangeKey
KEN	KeyExpiryNumber
KLF	KeyLoadFile
KMC	KeyManagementCentre
KMI	KeyManagementInfrastructure

KMS	KeyManagementSystem
KRN	KeyRevisionNumber
KT	KeyType
LAN	Local Area Network
LRC	LongitudinalRedundancyCheck
MFO	MeterFunctionObject
Mfr	Manufacturer
MII	MajorIndustryIdentifier
MIS	Management Information System
MPL	MaximumPowerLimit
MPPUL	MaximumPhasePowerUnbalanceLimit
NIST	National Institute of Standards and Technology
NKHO	NewKeyHighOrder bits
NKLO	NewKeyLowOrder bits
NWIP	New Work Item Proposal
OSI	Open Systems Interconnection
PAN	PrimaryAccountNumber
PLC	Power Line Carrier
POS	PointOfSale
PRN	Printer
PSTN	Public Switched Telephone Network
RND	RandomNumber
RO	Rollover
SG	SupplyGroup
SGC	SupplyGroupCode
STA	Standard Transfer Algorithm
STS	Standard Transfer Specification
STSA	Standard Transfer Specification Association
STX	ASCII Start of Text character
TCDU	TokenCarrierDataUnit
TCT	TokenCarrierType
TDEA	Triple Data Encryption Algorithm
TI	TariffIndex
TID	TokenIdentifier
UC	UtilityCode
VCDK	VendingCommonDESKey
VDDK	VendingDefaultDESKey
VK	VendingKey
VUDK	VendingUniqueDESKey
WAN	Wide Area Network

XOR	Exclusive Or (logical)

3.3 Notation and terminology

Throughout this standard the following rules are observed regarding the naming of terms.

- Entity names, data element names, function names and process names are treated as generic object classes and are given names in terms of phrases in which the words are capitalized and joined without spaces. Examples are: SupplyGroupCode as a data element name, EncryptionAlgorithm07 as a function name and TransferCredit as a process name (see note).
- Direct (specific) reference to a named class of object uses the capitalized form, while general (non-specific) reference uses the conventional text i.e. lower case form with spaces. An example of a direct reference is: “The SupplyGroupCode is linked to a group of meters”, while an example of a general reference is: “A supply group code links to a vending key”.
- Other terms use the generally accepted abbreviated forms like PSTN for Public Switched Telephone Network.

NOTE The notation used for naming of objects has been aligned with the so-called “camel-notation” used in the common information model (CIM) standards prepared by IEC TC57, in order to facilitate future harmonization and integration of payment system standards with the CIM standards.

4 Numbering conventions

In this standard, the representation of numbers in binary strings uses the convention that the least significant bit is to the right and the most significant bit is to the left.

Numbering of bit positions start with bit position 0, which corresponds to the least significant bit of a binary number.

Numbers are generally in decimal format, unless otherwise indicated. Any digit without an indicator signifies decimal format.

Binary digit values range from 0-1.

Decimal digit values range from 0-9.

Hexadecimal digit values range from 0-9, A-F and are indicated by “hex”.

5 Reference model for the standard transfer specification

5.1 Generic payment meter functional reference diagram

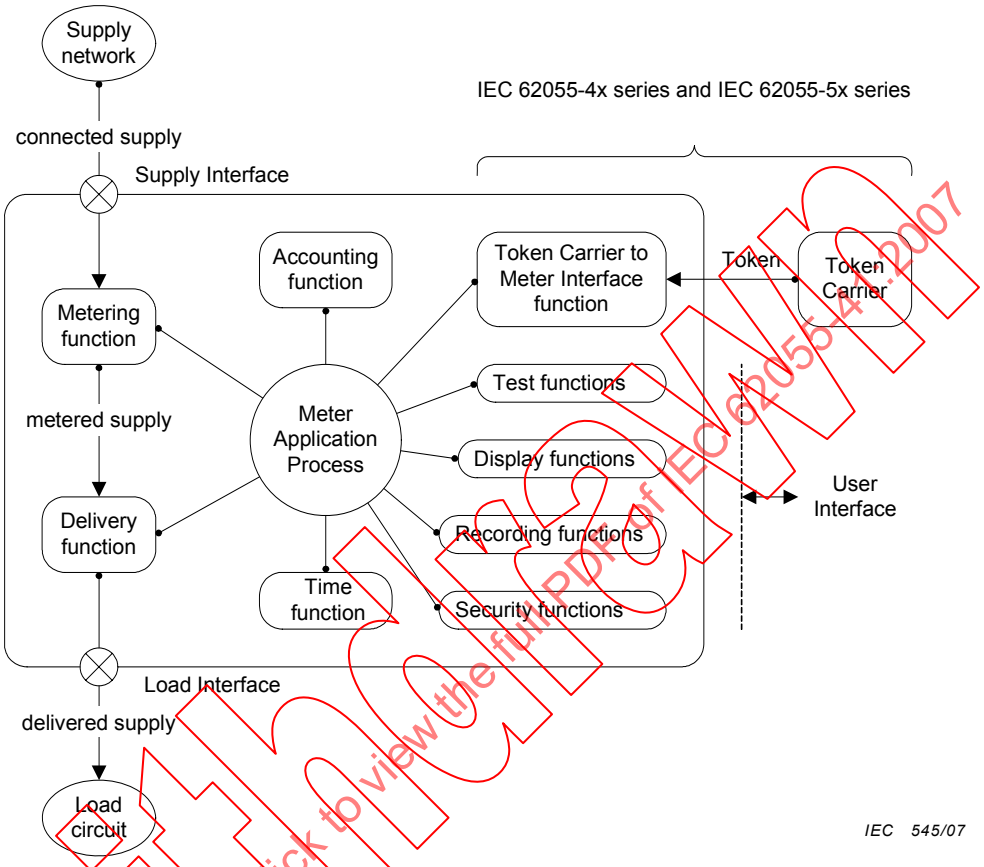


Figure 1 – Functional block diagram of a generic single-part payment meter

The IEC 62055-4x series primarily deals with the application layer protocol and IEC 62055-5x series with the physical layer protocol of the TokenCarrierToMeterInterface. The TokenCarrier is included in the Physical Layer. In this standard the Decoder (see Clause 3) is defined as that part of the payment meter where the Application Layer functions of the TokenCarrierToMeterInterface are located and it is thus allocated a DRN (see 6.1.2.3).

NOTE MeterFunctionObjects are further discussed in 5.5.

In a single-part payment meter all the essential functions are located in a single enclosure as depicted in Figure 1 above, in which case the decoder is integral with the metering function and the DRN could thus optionally be synonymous with the meter serial number.

In a multi-part payment meter it is possible for the TokenCarrierToMeterInterface to be located in a separate enclosure from that of the metering function, for example, which may well be a stand-alone meter in its own right and having its own meter serial number. In this case, the DRN would not be the same as the meter serial number but would be distinctly different and would thus be marked on the enclosure containing the decoder.

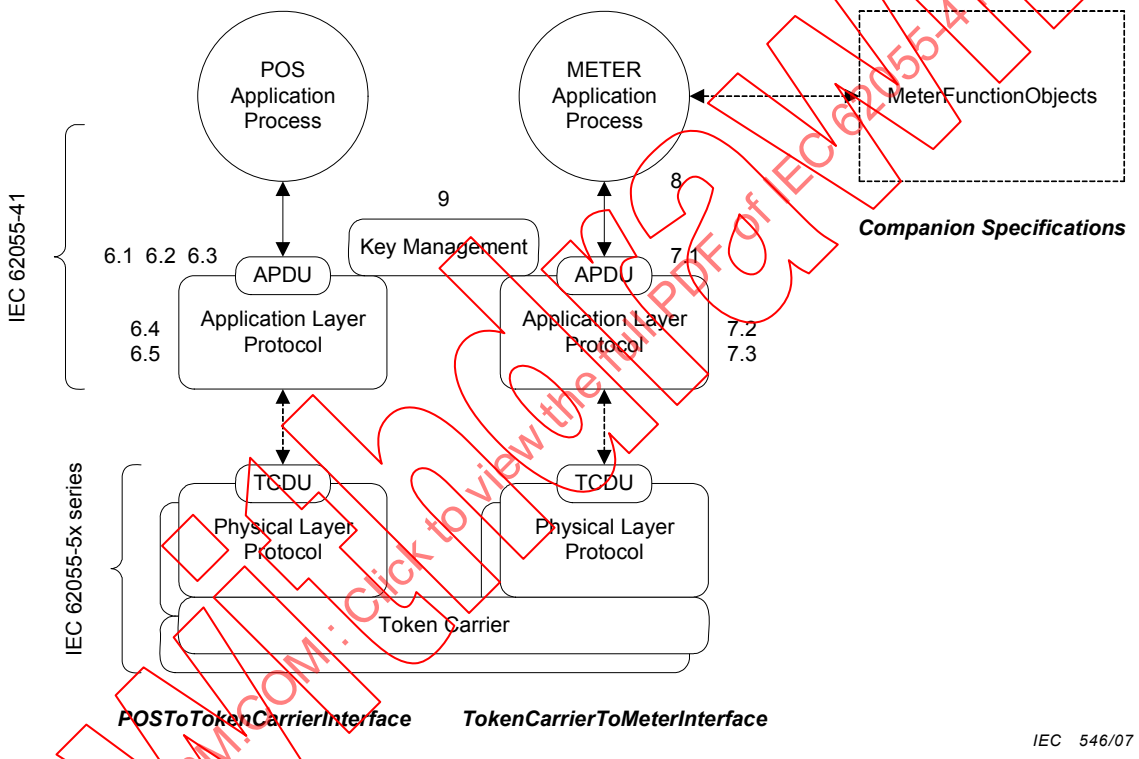
In all cases, there shall only be one Application Layer implementation and thus there shall be only one DRN associated with a payment meter, whether it is single- or multi-part, even

though there may also be more than one Physical Layer implementation in the same payment meter

It is also possible that the Application Layer functions and the Physical Layer functions are located in separate enclosures, in which case, the marking (see 8.3) of the DRN and the EA code is applied to that part that contains the physical TokenCarrier connection point. This may be a cable or modem connector for a virtual token carrier, a keypad for a numeric token carrier or a magnetic card reader for magnetic card token carrier for example (see also 5.2 for more examples of token carriers).

For a more complete description of payment meter function classes see IEC 62055-21.

5.2 STS protocol reference model



Key

APDU ApplicationProtocolDataUnit; data interface to the application layer protocol

TCDU TokenCarrierDataUnit; data interface to the physical layer protocol

Relevant clause number references in this standard are indicated adjacent to each box

Figure 2 – STS modelled as a 2-layer collapsed OSI protocol stack

The STS is a secure data transfer protocol between a POS and a payment meter using a token carrier as the transfer medium. The application layer protocol deals with tokens and encryption processes and functions, while the physical layer protocol deals with the actual encoding of token data onto a token carrier (see Figure 2).

Examples of physically transportable token carrier devices are: numeric, magnetic cards, memory cards and memory keys. Examples of virtual token carriers are: PSTN modem, ISDN modem, GSM modem, GPRS modem, radio modem, PLC modem, Infra-red, LAN and WAN connections and direct local connection. These are defined in the IEC 62055-5x series.

It must be noted that although the model primarily depicts a POS to token carrier to payment meter protocol, the same protocol is equally applicable to any other device that requires communicating with the payment meter, for example, CIS, MIS or portable HHU.

Although a collapsed 2-layered OSI architecture is followed in this standard, it does not preclude future expansion to include more layers should the need arise or for the implementer to interpose additional layers between the two shown in this model.

The APDU is the data interface to the application layer protocol, specified in this standard and the TCDU is the data interface to the physical layer protocol, specified in the IEC 62055-5x series.

The STS in this standard defines a one-way data transfer protocol (i.e. from POS to payment meter), although the reference model allows equally for a two-way transfer protocol, which may be a requirement in a future revision of this standard.

5.3 Dataflow from the POSApplicationProcess to the TokenCarrier

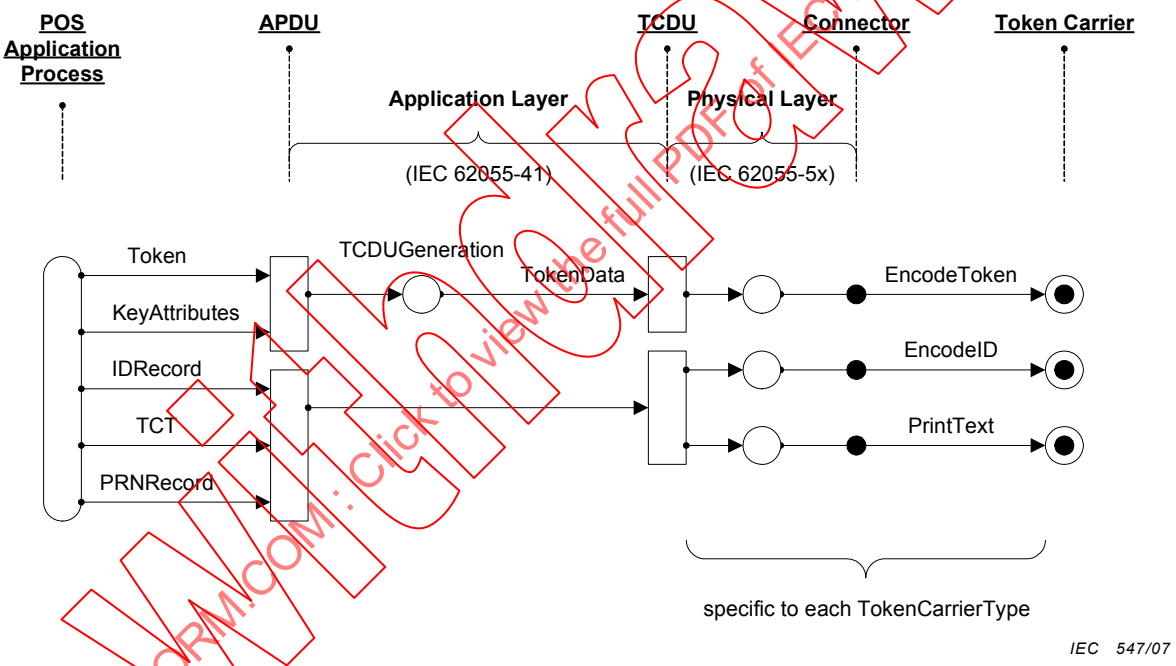


Figure 3 – Dataflow from the POSApplicationProcess to the TokenCarrier

The flow of data from the POSApplicationProcess to the TokenCarrier is shown in Figure 3.

The POSApplicationProcess presents the token to the APDU together with the KeyAttributes of the DecoderKey that is to be used for encrypting the token. The application layer protocol generates the DecoderKey, encrypts the token and presents the resultant TokenData in the TCDU. The physical layer protocol encodes the TokenData onto the TokenCarrier. Optionally, payment meter identification data may also be encoded onto the TokenCarrier (see 5.2.4 in IEC 62055-51 for example) as well as printed text onto the outside surface (see 5.1.5 in IEC 62055-51 for example). This part of the process essentially ends with the encoding of data onto the TokenCarrier, after which the TokenCarrier is transported to the payment meter (usually by the customer), where it is entered into the payment meter via the TokenCarrierInterface.

5.4 Dataflow from the TokenCarrier to the MeterApplicationProcess

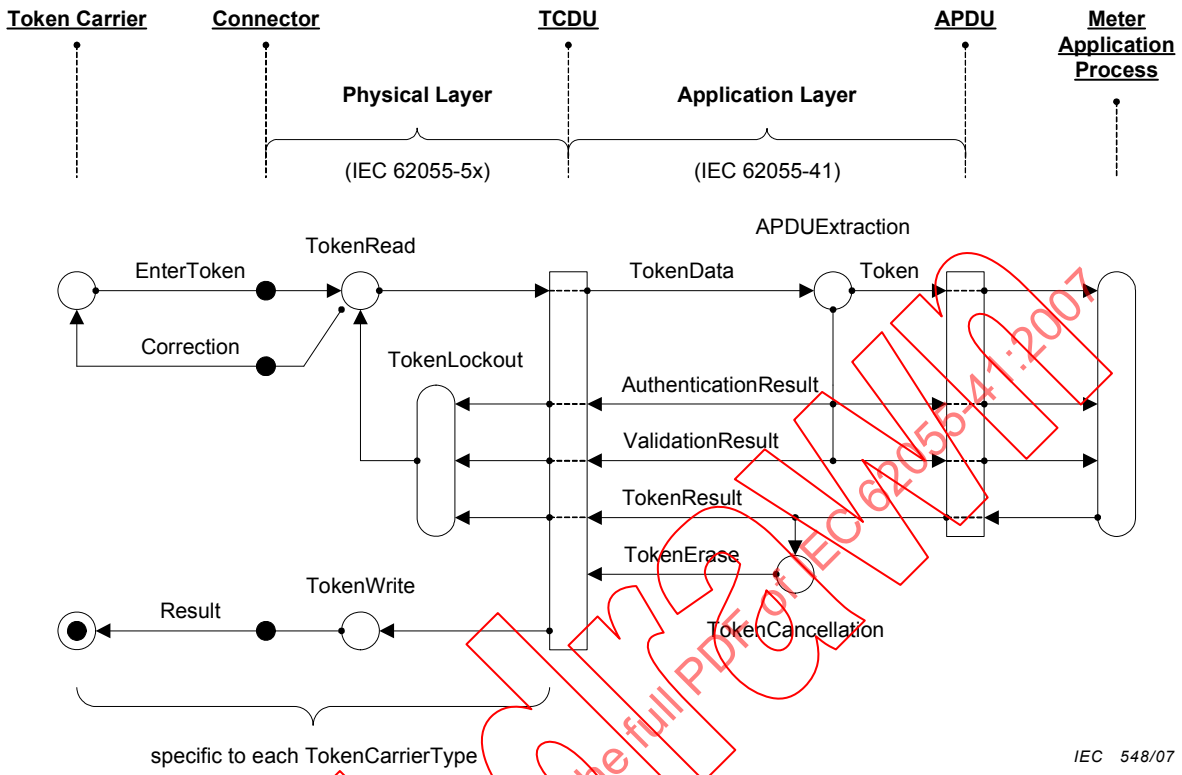


Figure 4 – Dataflow from the TokenCarrier to the MeterApplicationProcess

The flow of data from the TokenCarrier to the MeterApplicationProcess is shown in Figure 4.

The token entry process from the TokenCarrier varies according to the TCT. The nature of the connector will similarly vary according to the TCT, an example of which may be a keypad or a magnetic card reader device supporting one-way token carriers as specified in IEC 62055-51.

NOTE Other types of connectors would be required to support other types of token carriers, such as a memory key reader device or a plug-in connector from a hand-held unit acting as a virtual token carrier. Such token carriers might be specified in additional parts of IEC 62055-5x in the future.

The physical layer protocol reads the token data being entered and provides immediate corrective feedback to the user (see 6.3 in IEC 62055-51 for example). The entered token data is presented in the TCDU, from where the application layer protocol extracts the token by appropriate decryption, validation and authentication, the results of which are presented to the MeterApplicationProcess in the APDU. After processing and executing the instruction from the token, the MeterApplicationProcess indicates the result in the APDU for the application layer protocol to take further action. This normally causes the cancellation of the TID and the giving of the instruction, via the TCDU, to the physical layer protocol to complete the token entry process by erasure of the token data (if appropriate) or by writing of other relevant data back onto the TokenCarrier as may be appropriate.

For certain TokenCarrier types (for example, a high speed virtual token carrier) the physical layer protocol may employ a token entry lockout function to protect the payment meter from fraud attempts. Typically, such a lockout function would slow down the effective rate, at which tokens may be entered via the particular token carrier interface (see 6.6.6 of IEC 62055-52 for example).

5.5 MeterFunctionObjects/companion specifications

With reference to Figure 1, it can be seen that the TokenCarrierToMeterInterface, which also includes the TokenCarrier, is dealt with in the IEC 62055-4x and IEC 62055-5x series. The remaining MeterFunctionObjects shown in the diagram are defined in companion specifications and are not normative to this standard.

Companion specifications (see Figure 2) are under the administrative control (see Clause C.8) of the STS Association and serve the purpose of defining the functionality of a payment meter in a standardized way, using an object-oriented approach.

5.6 ISO transaction reference numbers

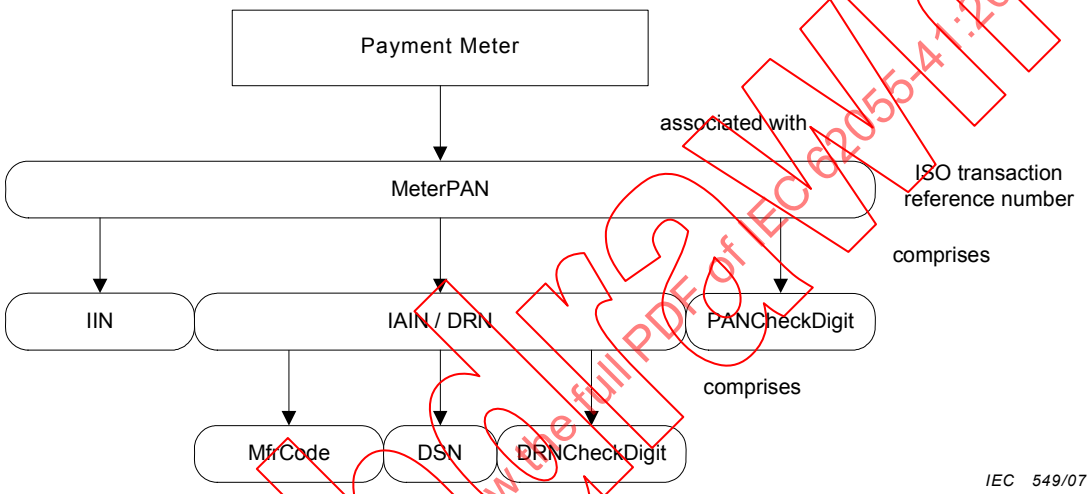


Figure 5 – Composition of ISO transaction reference number

The ISO transaction reference number comprises the data elements and their relationships as shown in Figure 5.

A token-based transaction (see Clause 3) constitutes a financial activity that needs to be dealt with in accordance with standard financial practices.

The PrimaryAccountNumber (PAN) as defined by ISO/IEC 7812-1 serves to tag transaction records, messages, requests, authorizations and notifications, in which both transacting parties are uniquely identifiable.

A payment meter is thus uniquely associated with a MeterPAN, being a composite number comprising of IIN and IAIN/DRN, which in turn comprises MfrCode and DSN (see 6.1.2).

6 POSToTokenCarrierInterface application layer protocol

6.1 APDU: ApplicationProtocolDataUnit

6.1.1 Data elements in the APDU

The APDU is the data interface between the POSApplicationProcess and the application layer protocol and comprises the data elements given in Table 1.

Table 1 – Data elements in the APDU

Element	Context	Format	Reference
MeterPAN	ISO compliant identification MeterPrimaryAccountNumber for the payment meter	18 digits	6.1.2
TCT	Directs which TokenCarrierType should be used in the physical layer protocol to carry the token to the payment meter	2 digits	6.1.3
DKGA	Directs which DecoderKeyGenerationAlgorithm is to be used for generating the DecoderKey	2 digits	6.1.4
EA	Directs which encryption algorithm is to be used for encrypting the token data	2 digits	6.1.5
SGC	Directs which SupplyGroupCode the payment meter is allocated to	6 digits	6.1.6
TI	Directs which TariffIndex the payment meter is linked to	2 digits	6.1.7
KRN	Directs which KeyRevisionNumber the DecoderKey is on	1 digits	6.1.8
KT	Directs which KeyType the DecoderKey is on	1 digits	6.1.9
KeyExpiryNumber	A number associated with the VendingKey and a DecoderKey that determines the time period, during which the key will remain valid	8 bits	6.1.10
Token	The actual token data that is to be transferred to the payment meter prior to encryption and processing	66 bits	6.2.1
IDRecord	Optional identification data intended to be encoded onto a payment meter ID card or onto a token carrier together with the token	35 digits	Table 2
PRNRecord	Optional print data intended to be printed at the same time as the coding of the token onto the TokenCarrier. Certain token carriers such as paper-based magnetic card devices allow printing to be done onto the card surface itself and this operation may be integrated with the magnetic card encoding device. The content and format is not specified and is left to each system to define according to its particular requirements	Undefined text	x

The optional IDRecord comprises the data elements given in Table 2.

Table 2 – Data elements in the IDRecord

Element	Context	Format	Reference
MeterPAN	ISO compliant identification MeterPrimaryAccountNumber for the payment meter	18 digits	6.1.2
DOE	Optional expiry date for the identification data as encoded onto a payment meter ID card or token carrier (as an example, see IEC 62055-51)	4 digits	6.1.11
TCT	Indicates which TokenCarrierType is associated with this MeterPAN	2 digits	6.1.3
EA	Indicates which encryption algorithm is associated with this MeterPAN	2 digits	6.1.5
SGC	Indicates which SupplyGroupCode is associated with this MeterPAN	6 digits	6.1.6
TI	Indicates which TariffIndex is associated with this MeterPAN	2 digits	6.1.7
KRN	Indicates which KeyRevisionNumber is associated with this MeterPAN	1 digits	6.1.8

6.1.2 MeterPAN: MeterPrimaryAccountNumber

6.1.2.1 Data elements in the MeterPAN

The MeterPAN is a unique identification number for each STS-compliant payment meter. It comprises the 3 parts given in Table 3 and is in accordance with the definition for the PAN (PrimaryAccountNumber) of ISO/IEC 7812-1.

Table 3 – Data elements in the MeterPAN

Element	Context	Format	Reference
IIN	IssuerIdentificationNumber	6 digits	6.1.2.2
IAIN / DRN	IndividualAccountIdentificationNumber/ DecoderReferenceNumber	11 digits	6.1.2.3
PANCheckDigit	Formula to check the integrity of the IIN and the IAIN	1 digit	6.1.2.4

NOTE The IIN is the 6 most significant digits of the 18-digit MeterPAN and the PANCheckDigit is the least significant digit.

See also Annex C for the code of practice on managing this data element.

6.1.2.2 IIN: IssuerIdentificationNumber

The IIN is a unique 6-digit number that defines a domain, under which further IAIN values (i.e. DRN values) may be issued for use within this defined domain.

The original intent and purpose of the IIN was to be able to tag financial transactions in order to uniquely identify them and to route them to the appropriate transacting financial accounts. It was thus intended that the IIN be issued by the ISO under the registration scheme given in ISO/IEC 7812-1 and ISO/IEC 7812-2. However, this has proven to be impractical and the value 600727 for IIN has since become the *de facto* standard, which is being used world wide for all STS-compliant electricity payment metering systems. The continuation of this practice is encouraged.

Thus, for the purpose of this standard where it applies to electricity STS-compliant systems, the IIN shall be 600727.

See also C.3.2 on managing this data element.

6.1.2.3 IAIN: IndividualAccountIdentificationNumber/ DRN: DecoderReferenceNumber

6.1.2.3.1 Data elements in the IAIN/DRN

A unique DRN shall be allocated to the device that performs the application layer protocol in an STS-compliant payment meter.

NOTE In many systems, the decoder part is integral with the metering part and hence the DRN might be synonymous with the meter serial number.

This is an 11-digit number comprising of the data elements given in Table 4.

Table 4 – Data elements in the IAIN / DRN

Element	Context	Format	Reference
MfrCode	A 2-digit number to uniquely identify a payment meter manufacturer	2 digits	6.1.2.3.2
DSN	An 8-digit serial number allocated by the manufacturer	8 digits	6.1.2.3.3
DRNCheckDigit	Check digit; formula to check the integrity of the MfrCode and the DSN	1 digit	6.1.2.3.4

NOTE The MfrCode is the 2 most significant digits of the 11-digit DRN and the DRNCheckDigit is the least significant digit.

MfrCode values less than 10 shall be right justified and left padded with 0. For example 01, 02-09.

The DSN shall be right justified and left padded with 0 to a full 8-digit string.

6.1.2.3.2 MfrCode: ManufacturerCode

The MfrCode is a 2-digit number that shall be used to uniquely identify the manufacturer of the payment meter.

The STS Association provides a service for the allocation of MfrCode values to uniquely identify manufacturers in order to ensure interoperability of STS-compliant equipment.

See also C.3.3 on managing this data element.

6.1.2.3.3 DSN: DecoderSerialNumber

The DSN is a unique 8-digit serial number that is generated internally by the manufacturer. Each manufacturer is responsible for the uniqueness of the DSN with respect to his MfrCode.

See also C.3.4 on managing this data element.

6.1.2.3.4 DRNCheckDigit

The DRNCheckDigit is a single digit used to validate the integrity of the MfrCode and DSN values when being entered by hand or being read by machine. This is a modulus 10 check digit, calculated using the Luhn formula, as illustrated in Annex B of ISO/IEC 7812-1. It is calculated on the 10 preceding digits of the DRN generated through the concatenation of the MfrCode and the DSN values.

6.1.2.4 PANCheckDigit

The PANCheckDigit is a single digit used to validate the integrity of the IIN and the IAIN values when being entered by hand or being read by machine. The method used to calculate the PANCheckDigit value is given in 4.4 of ISO/IEC 7812-1 and is calculated on the preceding 17 digits of the MeterPAN generated through the concatenation of the IIN and the IAIN values.

6.1.3 TCT: TokenCarrierType

This is a 2-digit number used to uniquely identify the type of token carrier onto which the token should be encoded for transferring to the payment meter. The values for token carrier types are given in Table 5.

Table 5 – Token carrier types

Code	TokenCarrier	Comments
00	Reserved	For future assignment
01	Magnetic card	As defined in IEC 62055-51
02	Numeric	As defined in IEC 62055-51
03-06	Reserved	Legacy systems using proprietary token carrier technologies
07	Virtual Token Carrier (VTC07)	As defined in IEC 62055-52
08-99	Reserved	For future assignment

Values less than 10 shall be right justified and left padded with 0 (for example 01, 02-09).

6.1.4 DKGA: DecoderKeyGenerationAlgorithm

This is a 2-digit number used to uniquely identify which algorithm is to be used for generating the DecoderKey. The DKGA code values are given in Table 6.

Table 6 – DKGA codes

Code	DKG algorithm	Comments	Reference
00	Reserved	For future assignment	x
01	DKGA01	Limited number of early legacy STS-compliant payment meters. Superseded by DKGA02	6.5.3.3
02	DKGA02	System using 64-bit DES VendingKey diversification	6.5.3.4
03	DKGA03	System using dual 64-bit DES VendingKey diversification	6.5.3.5
04-99	Reserved	For future assignment	x

NOTE 1 DKGA02 is the algorithm to be used for current systems, subject to the criteria for DKGA01.

NOTE 2 DKGA03 is the algorithm to be used for future systems requiring a higher level of security regarding protection of the VendingKey by “brute-force” attack.

NOTE 3 The introduction of DKGA03 should preferably coincide with the change from STA to DEA (EA code 07 to EA code 09). See also 6.1.5.

Values less than 10 shall be right justified and left padded with 0 (for example 01, 02-09).

6.1.5 EA: EncryptionAlgorithm

This is a 2-digit number used to uniquely identify which algorithm is to be used for encrypting the token data. The EA code values are given in Table 7.

Table 7 – EA codes

Code	EncryptionAlgorithm	Comments	Reference
00	Reserved	For future assignment	x
01-06	Reserved	Legacy proprietary systems	x
07	STA	Systems using the standard transfer algorithm as defined in this standard	6.5.4.1
08	Reserved	Legacy proprietary systems	x
09	DEA	Systems using the data encryption algorithm as defined in ANSI X3.92	6.5.5
10	Reserved	Legacy proprietary systems	x
11-99	Reserved	For future assignment	x
NOTE It is recommended that the choice of EA code 09 be coordinated with the choice of DKGA03 in order to minimize the effect on existing systems in the installed base (see 6.1.4).			

Values less than 10 shall be right justified and left padded with 0. For example 01, 02-09.

6.1.6 SGC: SupplyGroupCode

This is a unique 6-digit number allocated to a utility, which is registered within the KMS. It is used to uniquely identify a subgroup of payment meters within the supply or distribution domain of the utility. Each SupplyGroup has a VendingKey associated with it and hence each payment meter in the SupplyGroup has a derived DecoderKey associated with it. Token sales authorization is thus controlled by selective distribution of such VendingKey and SGC to authorized token vendor agents operating POS services on behalf of utilities.

SGC management and VendingKey management is completely under the control of the KMS and is subject to such code of practice.

Values less than 6 decimal digits shall be right justified and left padded with 0. For example 000001, 000002.. 000009.

The SGC inherits its type from the KT attribute of the VendingKey (see 6.5.2.2.1), to which it is associated as shown in Table 8.

Table 8 – SGC types and key types

KT	SGC type	VendingKey type (see 6.5.2.2.1)	DecoderKey type (see 6.5.2.3.1)
0	Initialization	Not specified	DITK
1	Default	VDDK	DDTK
2	Unique	VUDK	DUTK
3	Common	VCDK	DCTK

See also C.2.2 for the code of practice on managing this data element.

6.1.7 TI: TariffIndex

A 2-digit number associated with a particular tariff that is allocated to a particular customer. The maintenance of and the content of the tariff tables are the responsibility of the utility.

Values less than 10 shall be right justified and left padded with 0 (for example 01, 02.. 09).

The TI is also encoded into the DecoderKey, which means that when a customer is moved from one TI to another, then his DecoderKey will also have to change (see 6.5.2.1).

See also Clause C.9 for the code of practice on managing this data element.

6.1.8 KRN: KeyRevisionNumber

This is a 1-digit number in the range 1 to 9, which is associated with a version of the VendingKey and with the corresponding DecoderKey.

See 6.5.2.5 for a detailed definition of this data element.

6.1.9 KT: KeyType

This is a 1-digit number in the range 0 to 3 associated with a property of the VendingKey and thus also with the corresponding DecoderKey, which is derived from the VendingKey.

See 6.5.2 for a detailed definition of this data element.

6.1.10 KEN: KeyExpiryNumber

A KEN is associated with each VendingKey by the KMS and defines the time when a VendingKey and any corresponding DecoderKey will expire, after which it becomes invalid for further use subject to certain concessions.

The KEN corresponds to the most significant 8 bits of the 24-bit TID. Any token identifier whose most significant 8 bits are greater than a given key's KEN cannot be encrypted or decrypted with that key.

See 6.5.2.6 for a detailed definition of this data element.

See also C.2.4 for the code of practice on managing this data element.

6.1.11 DOE: DateOfExpiry

The use of this date is optional and is associated with a validity period for identity-related data that gets encoded onto an identity-carrying device. For example, a payment meter ID card or a second record encoded onto the TokenCarrier together with the token data. In some implementations it is found to be useful to let the customer bring back a used token carrier to serve as his decoder identification to the POS when purchasing his next token (see, for example, 5.1.4 and 5.2.4.9 of IEC 62055-51).

This date may also be used, for example, in cases where a consumer has been granted a concessionary tariff for a limited period. The date encoded is the last month for which the card is valid.

DOE is in the format YYMM and shall always contain 4 digits.

Where YY or MM is less than 10, it shall be right justified and left padded with 0 (for example 01, 02, 09, etc).

When the DOE in the IDRecord is not used, then YYMM = 0000.

DOE code values for the year and month are given in Tables 9 and 10.

Table 9 – DOE codes for the year

YY	Represents
00	2000 or DOE is not used (see also Table 10)
01 – 99	2001 – 2099

Table 10 – DOE codes for the month

MM	Represents
00	DOE is not used (see also Table 9)
01 – 12	Jan – Dec
13 – 99	Invalid

6.2 Tokens

6.2.1 Token definition format

The TokenData element in the APDU is a 66-bit binary number comprising several fields of smaller data elements, in accordance with which various processes are initiated in the MeterApplicationProcess and various bits of information are transferred to the payment meter registers.

The definition format for the tokens in 6.2.2 to 6.2.14 is given in Table 11.

Table 11 –Token definition format

Name of data element	Example: Class, SubClass, RND, TID, Amount, CRC, etc.
Number of bits	Example: 2 bits, 4 bits, 24 bits, 16 bits, etc.
Range of values	Example: 1, 2, 5-15, etc.

6.2.2 Class 0: TransferCredit

Class	SubClass	RND	TID	Amount	CRC
2 bits	4 bits	4 bits	24 bits	16 bits	16 bits
0	0 = electricity Reserved: 1 = water 2 = gas 3 = time 4 = currency 5-15 = future assignment				

Action: Transfer credit to the payment meter to the value as defined in the Amount field and for the service type as defined in the SubClass field.

NOTE The SubClass values 1-4 are reserved by the STS Association for applications other than electricity and values 5-15 are reserved for future assignment.

6.2.3 Class 1: InitiateMeterTest/Display

Class	SubClass	Control	MfrCode	CRC
2 bits	4 bits	36 bits	8 bits	16 bits
1	0 = STS defined	Bit position control of test/display number	0	
1	1-10 = reserved for future assignment.	Reserved for future assignment.	Reserved for future assignment.	
1	11-15 = proprietary use	If not used, then set to zero	0-99	

Action: Initiate the test or display function in the payment meter in accordance with the bit pattern defined in the Control field.

6.2.4 Class 2: SetMaximumPowerLimit

Class	SubClass	RND	TID	MPL	CRC
2 bits	4 bits	4 bits	24 bits	16 bits	16 bits
2	0				

Action: Load the maximum power limit register in the payment meter with the value as given in the MPL field.

6.2.5 Class 2: ClearCredit

Class	SubClass	RND	TID	Register	CRC
2 bits	4 bits	4 bits	24 bits	16 bits	16 bits
2	1				

Action: Clear the corresponding credit register (as indicated in the register field) in the payment meter to zero.

6.2.6 Class 2: SetTariffRate

Class	SubClass	RND	TID	Rate	CRC
2 bits	4 bits	4 bits	24 bits	16 bits	16 bits
2	2				

Action: Load the tariff rate register in the payment meter with the value given in the rate field.

This token is reserved by the STS Association for future definition.

6.2.7 Class 2: Set1stSectionDecoderKey

Class	SubClass	KENHO	KRN	RO	Res	KT	NKHO	CRC
2 bits	4 bits	4 bits	4 bits	1 bit	1 bit	2 bits	32 bits	16 bits
2	3		1-9	0-1	x	0-3		

Action: Load the DecoderKeyRegister with the 1st half of the new DecoderKey, subject to an authentic loading of a Set2ndSectionDecoderKey token.

6.2.8 Class 2: Set2ndSectionDecoderKey

Class	SubClass	KENLO	TI	NKLO	CRC
2 bits	4 bits	4 bits	8 bits	32 bits	16 bits
2	4		0-99		

Action: Load the DecoderKeyRegister with the 2nd half of the new DecoderKey, subject to an authentic loading of a Set1stSectionDecoderKey token.

6.2.9 Class 2: ClearTamperCondition

Class	SubClass	RND	TID	Pad	CRC
2 bits	4 bits	4 bits	24 bits	16 bits	16 bits
2	5			0	

Action: Clear the tamper status register in the payment meter and cancel any resultant control processes that may be in progress.

6.2.10 Class 2: SetMaximumPhasePowerUnbalanceLimit

Class	SubClass	RND	TID	MPPUL	CRC
2 bits	4 bits	4 bits	24 bits	16 bits	16 bits
2	6				

Action: Load the maximum phase unbalance limit register in the payment meter with the value given in the MPPUL field. See also 8.12 for more detail on the action of this function in the payment meter.

6.2.11 Class 2: SetWaterMeterFactor

Class	SubClass	RND	TID	WMFactor	CRC
2 bits	4 bits	4 bits	24 bits	16 bits	16 bits
2	7				

Action: Load the water meter factor register in the payment meter with the value given in the WMFactor field.

This token is reserved by the STS Association for water application.

6.2.12 Class 2: Reserved for STS use

Class	SubClass	RND	TID	ResData	CRC
2 bits	4 bits	4 bits	24 bits	16 bits	16 bits
2	8-10				

Action: Reserved for future definition by the STS Association.

This token range is reserved by the STS Association for future assignment.

6.2.13 Class 2: Reserved for Proprietary use

Class	SubClass	RND	TID	PropData	CRC
2 bits	4 bits	4 bits	24 bits	16 bits	16 bits
2	11-15				

Action: Defined by manufacturer.

This token range is reserved for proprietary definition and use.

This standard does not provide protection against collision between manufacturer uses of this token space. Generation and control of these tokens shall therefore always be under the direct management of the relevant manufacturer and shall never be available on vending systems for general use within STS-compliant payment metering systems.

6.2.14 Class 3: Reserved for STS use

Class	SubClass	Res
2 bits	4 bits	60 bits
3	0-15	

Action: Reserved for future definition by the STS Association.

This token range is reserved by the STS Association for future assignment

6.3 Token data elements

6.3.1 Data elements used in tokens

The data elements given in Table 12 are used in tokens in various combinations.

Table 12 – Data elements used in tokens

Element	Name	Format	Reference
Amount	TransferAmount (see also 6.2.2)	16 bits	6.3.6
Class	TokenClass (see also 6.2.2 to 6.2.14)	2 bits	6.3.2
Control	InitiateMeterTest/DisplayControlField (see also 6.2.3)	36 bits	6.3.8
CRC	CyclicRedundancyCode (see 6.2.2 to 6.2.13)	16 bits	6.3.7
KENHO	KeyExpiryNumberHighOrder (see also 6.2.7)	4 bits	6.3.16
KENLO	KeyExpiryNumberLowOrder (see also 6.2.8)	4 bits	6.3.17
KRN	KeyRevisionNumber (see also 6.2.7)	4 bits	6.1.8
KT	KeyType (see also 6.2.7)	2 bits	6.1.9
MfrCode	ManufacturerCode (see also 6.2.3)	8 bits	6.1.2.3.2
MPL	MaximumPowerLimit (see also 6.2.4)	16 bits	6.3.9
MPPUL	MaximumPhasePowerUnbalanceLimit (see also 6.2.10)	16 bits	6.3.10
NKHO	NewKeyHighOrder (see also 6.2.7)	32 bits	6.3.14
NKLO	NewKeyLowOrder (see also 6.2.8)	32 bits	6.3.15
Pad	Pad value with 0 (see also 6.2.9)	16 bits	x
PropData	Proprietary data field (see also 6.2.13)	16 bits	x
Rate	[TariffRate] For future definition (see also 6.2.6)	16 bits	6.3.11
Register	RegisterToClear (see also 6.2.5)	16 bits	6.3.13
Res	Reserved for future assignment (see also 6.2.7 and 6.2.14)	1 bits	x
ResData	Reserved data field for future assignment (see also 6.2.12)	16 bits	x
RND	RandomNumber (see also 6.2.2 to 6.2.13)	4 bits	6.3.4
RO	RolloverKeyChange (see also 6.2.7)	1 bits	6.3.18
SubClass	TokenSubClass (see also 6.2.2 to 6.2.14)	4 bits	6.3.3
TI	TariffIndex (see also 6.2.8)	8 bits	6.1.7
TID	TokenIdentifier (see also 6.2.2 to 6.2.13)	24 bits	6.3.5.1
WMFactor	[WaterMeterFactor] Reserved by the STS Association for water application (see also 6.2.11)	16 bits	6.3.12

6.3.2 Class: TokenClass

Tokens are classified into 4 main functional areas as given in Table 13.

Table 13 – Token classes

TokenClass	Function
0	Credit transfer
1	Non-meter-specific management
2	Meter-specific management
3	Reserved for future assignment

Class 0 and Class 2 tokens are encrypted using the DecoderKey, while Class 1 tokens are not encrypted and can thus be used on any STS-compliant payment meter.

6.3.3 SubClass: TokenSubClass

Further subclassification of the TokenClass is given in Table 14.

Table 14 – Token subclasses

<table>
<tr><th rowspan="2">Token SubClass</th><th colspan="4">TokenClass</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td>0</td><td>TransferCredit (electricity)</td><td>InitiateMeterTest/Display</td><td>SetMaximumPowerLimit</td><td rowspan="9">Reserved by STS Association for future assignment</td></tr>
<tr><td>1</td><td>TransferCredit (water)
Reserved by STS Association for water applications</td><td rowspan="8">Reserved by STS Association for future assignment</td><td>ClearCredit</td></tr>
<tr><td>2</td><td>TransferCredit (gas)
Reserved by STS Association for gas applications</td><td>SetTariffRate</td></tr>
<tr><td>3</td><td>TransferCredit (time)
Reserved by STS Association for connection time applications</td><td>Set1stSectionDecoderKey</td></tr>
<tr><td>4</td><td>TransferCredit (currency)
Reserved by STS Association for currency applications</td><td>Set2ndSectionDecoderKey</td></tr>
<tr><td>5</td><td rowspan="3">Reserved by STS Association for future assignment</td><td>ClearTamperCondition</td></tr>
<tr><td>6</td><td>SetMaximumPhasePowerUnbalanceLimit</td></tr>
<tr><td>7</td><td>SetWaterMeterFactor
Reserved by STS Association for water applications</td></tr>
</table>

Table 14 *(continued)*

Token SubClass	TokenClass			
	0	1	2	3
8			Reserved by STS Association for future assignment	
9				
10				
11		Reserved for proprietary use	Reserved for proprietary use	
12				
13				
14				
15				

6.3.4 RND: RandomNumber

The generation of this 4-bit number will be a snapshot of the four least significant bits of at least a millisecond counter. The inclusion of a random number in the data to be transferred enhances the security of the token transfer by providing a probability of 16:1 that no two tokens containing identical data to be transferred will have the same binary pattern.

6.3.5 TID: TokenIdentifier

6.3.5.1 TID calculation

The TID field is derived from the date and time of issue and indicates the number of minutes elapsed from an STS base date and time.

The STS base date and time is 01 January 1993, 00:00:00.

For a date and time format of YYYY:MM:DD:hh:mm:ss the STS base date and time is 1993:01:01:00:00:00 corresponding to a TID of 0.

The calculation of elapsed minutes shall take leap years into account.

The rule used to determine a leap year is as follows.

- The month of February shall have an extra day in all years that are evenly divisible by 4, except for century years (those ending in -00), which receive the extra day only if they are evenly divisible by 400. Thus, 1996 was a leap year whereas 1999 was not, and 1600, 2000 and 2400 are leap years but 1700, 1800, 1900 and 2100 are not.

This field is a 24-bit binary representation of the elapsed minutes, the leftmost bit being the most significant bit.

When calculating the TID the ":ss" value shall be truncated from the actual time.

Examples of TID calculated values are given in Table 15.

Table 15 – TID calculation examples

Date of issue:	Time of issue:	Elapsed minutes:	Resultant 24-bit token ID:
1 January 1993	00:00:00	0	0000 0000 0000 0000 0000 0000
1 January 1993	00:01:45	1	0000 0000 0000 0000 0000 0001
25 March 1993	13:55:22	120 355	0000 0001 1101 0110 0010 0011
25 March 1996	13:55:22	1 698 595	0001 1001 1110 1011 0010 0011
1 November 2005	00:01:55	6 749 281	0110 0110 1111 1100 0110 0001
1 December 2015	00:01:05	12 051 361	1011 0111 1110 0011 1010 0001
24 November 2024	20:15:00	16 777 215	1111 1111 1111 1111 1111 1111

6.3.5.2 SpecialReservedTokenIdentifier

The TokenIdenifier corresponding to 00 hours 01 minutes of each day is reserved for special application tokens and may not be used for any other token.

Using the date and time format of YYYY:MM:DD:hh:mm:ss the reserved TID values correspond to xxxx:xx:xx:00:01:xx.

If a token, other than a special application token is to be generated on a time corresponding to this reserved TID, then 1 minute shall be added to the TID.

See also Clause C.4 for the code of practice for the management of this special reserved TID.

NOTE The use of special application tokens are optional (see Clause C.11), but the rule for how to use the special reserved TID is mandatory.

6.3.5.3 Multiple tokens generated within the same minute

The POS shall ensure that no legitimately purchased token can carry the same TID as that of any other legitimately purchased token for the same payment meter even if more than one token is purchased within the same minute on the same POS.

If multiple tokens need to be generated within the same minute for the same payment meter, then 1 min shall be added to the TID of each successive token in the set. At the end of the token generating process, the POS shall revert back to real time again.

This shall apply to any token that implements a TID.

This shall not apply to special application tokens that implement the SpecialReserved TokenIdentifier (see 6.3.5.2)

For example: if 3 credit tokens A, B and C are generated within the same minute at 13h23 and in sequential order A, B and C, then A shall carry the TID time stamp 13h23, B shall carry time stamp 13h24 and C shall carry 13h25.

6.3.6 Amount: TransferAmount

The associated unit for the transfer amount is defined in Table 16.

Table 16 – Units of measure for electricity

Transfer type	Units of measure
Electrical energy	Watt-hours x 100
Electrical power	Watts

The STS Association also reserves the transfer types given in Table 17 for other applications.

Table 17 – Units of measure for other applications

Transfer type	Units of measure
Water	Litres x 100
Gas	To be defined
Time	Minutes
Currency	To be defined
NOTE The STS Association defines other future transfer types for other utility services.	

The 16 bits of the transfer amount field are subdivided into 2 sections, a base-10 exponent of 2 bits and a mantissa of 14 bits. The bits are numbered from right to left, starting at 0. Bit 15 is the most significant bit of the exponent and Bit 13 is the most significant bit of the mantissa. The bit allocations within this field are illustrated in Table 18.

Table 18 – Bit allocations for the TransferAmount

Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	e	e	m	m	m	m	m	m	m	m	m	m	m	m	m	m

The equation for transfer amount conversion is as follows:

$t = 10^{e} \times m$, for $e = 0$; or

$$t = (10^{e} \times m) + \sum_{n=1}^{e}\left(2^{14} \times 10^{(n-1)}\right), \text{ for } e > 0$$

where

t is the transfer amount,

e is the base 10 exponent,

m is the mantissa, and

n is an integer in the range 1 to e inclusive.

All transfer amount conversions shall be rounded up in favour of the customer. The possible transfer amount ranges and the associated maximum errors that can arise owing to rounding up are shown in Table 19. Examples of TransferAmount values are given in Table 20.

Table 19 – Maximum error due to rounding

Exponent value	Transfer amount range	Maximum error
0	0000000 to 00016383	0,000
1	0016384 to 00180214	0,061 %
2	0180224 to 01818524	0,055 %
3	1818624 to 18201624	0,055 %

Table 20 – Examples of TransferAmount values

	Units purchased:	Resultant 16-bit transfer amount field:
1	0,1 kWh	0000 0000 0000 0001
2	25,6 kWh	0000 0001 0000 0000
3	1638,3 kWh	0011 1111 1111 1111
4	1 638,4 kWh	0100 0000 0000 0000
5	18022,3 kWh	0111 1111 1111 1111
6	18022,4 kWh	1000 0000 0000 0000
7	181862,3 kWh	1011 1111 1111 1111
8	181862,4 kWh	1100 0000 0000 0000
9	1820162,4 kWh	1111 1111 1111 1111

6.3.7 CRC: CyclicRedundancyCode

The CRC is a checksum field used to verify the integrity of the data transferred. The checksum is derived using the following CRC generator polynomial:

$$x^{16} + x^{15} + x^2 + 1$$

The total length of the data transferred via the token is 66 bits. The last 16 bits comprise the CRC checksum that is derived from the preceding 50 bits. These 50 bits are left padded with 6 binary zeros to make 56 bits. Before calculation, the CRC checksum is initialized to FFFF hex (see example in Table 21).

Table 21 – Example of a CRC calculation

Original 50 bits	0 00 4A 2D 90 0F F2 hex
Left padded to make 7 bytes	00 00 4A 2D 90 0F F2 hex
Checksum calculated	0F FA hex

6.3.8 Control: InitiateMeterTest/DisplayControlField

The initiate payment meter test data field is 36 bits long and is used to indicate the type of test to be performed. The particular test is selected by setting the relevant bit to a logical one. The permissible field values are defined in Table 22.

Table 22 – Permissible control field values

LS Bit No. = 1	Test No	Action	Condition
All bits = 1	0	Do test No. 1 to 5 plus, optionally, any other	Mandatory
1	1	Test the load switch	Mandatory
2	2	Test the payment meter information display devices	Mandatory
3	3	Display cumulative kWh energy register totals	Mandatory
4	4	Display the KRN	Mandatory
5	5	Display the TI	Mandatory
6	6	Test the token reader device	Optional
7	7	Display maximum power limit	Optional
8	8	Display tamper status	Optional
9	9	Display power consumption	Optional
10	10	Display software version	Optional
11	11	Display phase power unbalance limit	Optional
12	12	Reserved by STS Association for: Display water meter factor	Mandatory for water payment meter
13	13	Display tariff rate	Mandatory for currency-based payment meter
14-36	Reserved	Reserved by STS Association for future assignment	Reserved

NOTE The cumulative kWh energy register is defined in 5.11.4 of IEC 62055-31.

All payment meters shall support test number 0; if any of the incorporated tests are not supported, the payment meter shall perform the subset of tests that are supported.

6.3.9 MPL: MaximumPowerLimit

The maximum power limit field is a 16-bit field that indicates the maximum power that the load may draw, in watts. Calculation of this field is identical to that of the TransferAmount field (see 6.3.6). See also note in 8.6 for functional requirements of the MeterApplication Process.

6.3.10 MPPUL: MaximumPhasePowerUnbalanceLimit

The maximum phase power unbalance limit field is a 16-bit field that indicates the maximum allowable power difference between phase loads, in watts. Calculation of this field is identical to that of the TransferAmount field (see 6.3.6).

6.3.11 Rate: TariffRate

Reserved by the STS Association for future definition.

6.3.12 WMFactor: WaterMeterFactor

Reserved by the STS Association for water application.

6.3.13 Register: RegisterToClear

A unique 16-bit binary value in the range 0 to FFFF hex; to select the particular register that should be cleared with the ClearCredit token. The defined values are given in Table 23.

Table 23 – Selection of register to clear

Value	Action
0	Electricity Credit register
1	Reserved by STS Association for Water Credit register
2	Reserved by STS Association for Gas Credit register
3	Reserved by STS Association for Time Credit register
4	Reserved by STS Association for Currency Credit register
5 to FFFE hex	Reserved for future assignment
FFFF hex	Clear all Credit registers in the payment meter

6.3.14 NKHO: NewKeyHighOrder

The high-order 32 bits of the new DecoderKey that has been generated (see 6.4.4) and which is to be transferred to the payment meter by means of the token.

6.3.15 NKLO: NewKeyLowOrder

The low-order 32 bits of the new DecoderKey that has been generated (see 6.4.5) and which is to be transferred to the payment meter by means of the token.

6.3.16 KENHO: KeyExpiryNumberHighOrder

This is the high-order 4 bits of the KEN (see 6.1.10).

6.3.17 KENLO: KeyExpiryNumberLowOrder

This is the low-order 4 bits of the KEN (see 6.1.10).

6.3.18 RO: RolloverKeyChange

If the RolloverKeyChange bit is set = 1, the payment meter shall perform a rollover key change. This operation is identical to a normal key change, except that the TID memory store in the payment meter is filled with token identifiers of value 0 (zero).

6.4 TCDUGeneration functions

6.4.1 Definition of the TCDU

The TCDU may be different for each TokenCarrierType and is therefore defined separately for each physical layer protocol standard relevant to each part of the IEC 62055-5x series.

6.4.2 Transposition of the Class bits

This function is used by other TCDUGeneration functions (see 6.4.3 to 6.4.5). It inserts the 2 Class bits into the 64-bit data stream to make a 66-bit number according to the method outlined below.

The 64-bit number has its least significant bit in bit position 0 and its most significant bit in bit position 63. The 64-bit binary number string is modified to include the unencrypted token Class The 2-bit token Class value is inserted to occupy bit positions 28 and 27. The original values of bit positions 28 and 27 are relocated to bit positions 65 and 64. The most significant bit of the token Class now occupies bit position 28. The process is shown in Figure 6.

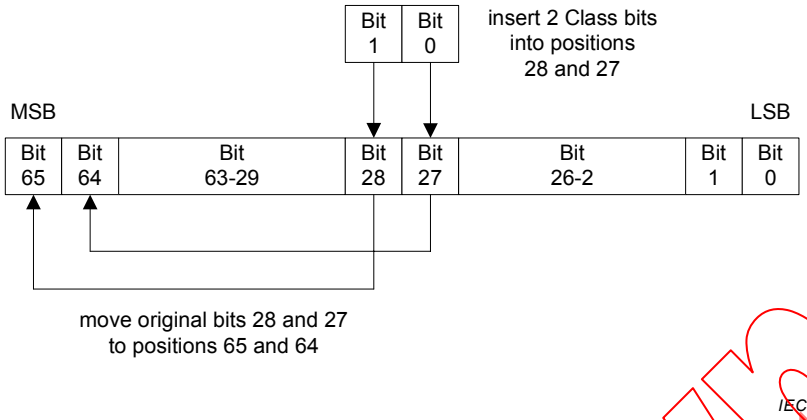


Figure 6 – Transposition of the 2 Class bits

Example: Insertion of the token Class = 01 (binary).

The 64-bit binary number grouped in nibbles: (Bits 27 and 28 highlighted in bold).

0110 0101 0100 0011 0010 0001 0000 1001 100**0** **0**111 0110 0101 0100 0011 0010 0001

Copy bits 28 and 27 into bit positions 65 and 64, creating a 66-bit number:

00 0110 0101 0100 0011 0010 0001 0000 1001 1000 0111 0110 0101 0100 0011 0010 0001

Replace bits 28 and 27 with the 2 Class bits:

00 0110 0101 0100 0011 0010 0001 0000 1001 100**0** **1**111 0110 0101 0100 0011 0010 0001

6.4.3 TCDUGeneration function for Class 0,1 and 2 tokens

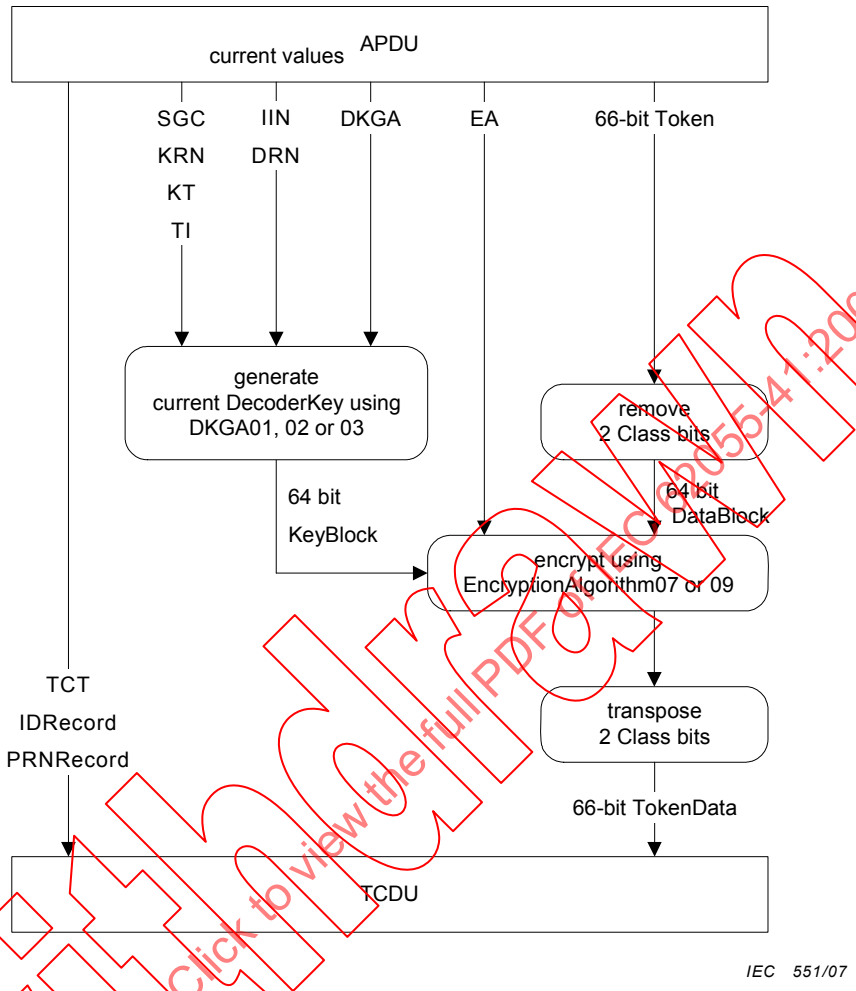


Figure 7 – TCDUGeneration function for Class 0, 1 and 2 tokens

This is the transfer function from the APDU to the TCDU (see Figure 7) and is applicable to all Class 0, Class 1 and Class 2 tokens, except for the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens (see 6.4.4 and 6.4.5).

NOTE 1 The data elements in the APDU are defined in 6.1.1.

NOTE 2 The data elements in the TCDU are defined in part of the IEC 62055-5x series physical layer protocol standard relevant to the specific TCT of interest.

The transfer function for Class 0 and Class 2 tokens is outlined as follows.

- The 2 Class bits are removed from the 66-bit token to yield a 64-bit result, which is then presented to the encryption algorithm as its DataBlock input. The specific algorithm to use is in accordance with the EA code in the APDU.
- The KeyBlock input for the encryption algorithm is obtained from the decoder key generation algorithm, which generates the current DecoderKey using the current values of SGC, KRN, KT, TI, IIN and DRN from the APDU as indicated. The specific decoder key generation algorithm to use is in accordance with the value of DKGA in the APDU.
- After encryption the 2 Class bits are again re-inserted into the 64-bit number in accordance with the method defined in 6.4.2 to yield a 66-bit result, which is populated into the TokenData field of the TCDU in accordance with the particular definition in the relevant physical layer protocol standard.

- Similarly the TCT, IDRecord and PRNRecord data elements from the APDU are transferred to the TCDU as indicated, into the appropriate fields of the TCDU in accordance with the particular definition in the relevant physical layer protocol standard.

The transfer function for Class 1 tokens is identical to the TCDUGeneration function for Class 0 and Class 2 tokens, except that the token does not get encrypted. The function is outlined as follows.

- The 2 Class bits are removed from the 66-bit token and transposed in accordance with the method defined in 6.4.2 to yield a 66-bit result, which is populated into the TokenData field of the TCDU in accordance with the particular definition in the relevant physical layer protocol standard.
- Similarly the TCT, IDRecord and PRNRecord data elements from the APDU are transferred to the TCDU as indicated, into the appropriate fields of the TCDU in accordance with the particular definition in the relevant physical layer protocol standard.

6.4.4 TCDUGeneration function for Set1stSectionDecoderKey token

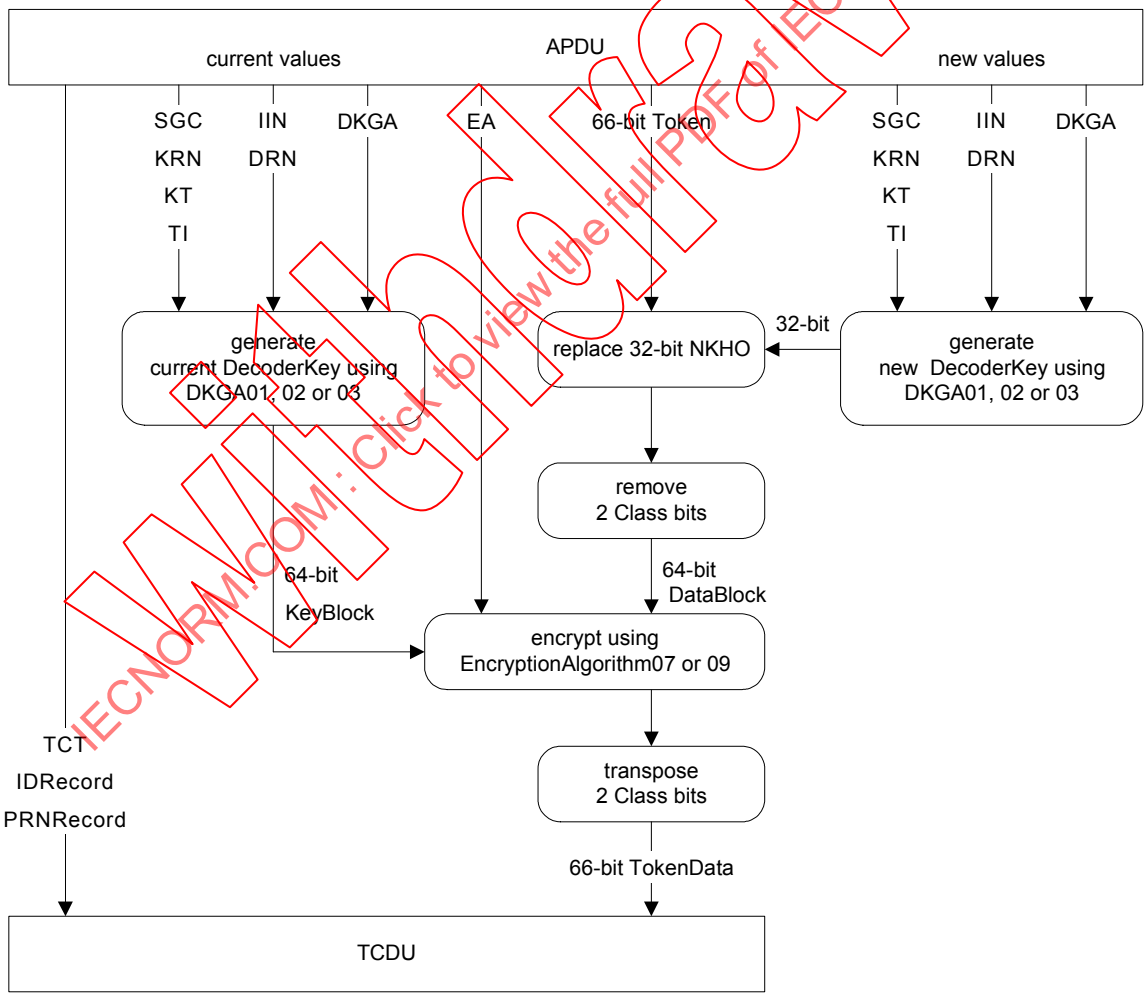


Figure 8 – TCDUGeneration function for Set1stSectionDecoderKey token

This is the transfer function from the APDU to the TCDU (see Figure 8) and is applicable only to the Set1stSectionDecoderKey token.

The Set1stSectionDecoderKey TCDUGeneration function is shown here as being separate from the Set2ndSectionDecoderKey TCDUGeneration function, but in practice the two may be merged into one in order to save on processing resource and for the sake of convenience. In such a case, the new DecoderKey generation only needs to happen once, for example, although the final result is still the same. Thus two separate TCDU instances are always produced: one for the Set1stSectionDecoderKey token and a second for the Set2ndSectionDecoderKey token.

It should be noted that the APDU shall present two sets of data for the PANBlock and CONTROLBlock: one set with the new data for the new DecoderKey and a second set with the current data for the current DecoderKey. The DKGA value is the same for both sets.

NOTE 1 The data elements in the APDU are defined in 6.1.1.

NOTE 2 The data elements in the TCDU are defined in each part of the IEC 62055-5x series physical layer protocol standard relevant to the specific TCT of interest.

The transfer function is outlined as follows.

- The new DecoderKey is generated using the new values of SGC, KRN, KT, TI, IIN and DRN. The specific algorithm to use is in accordance with the value of DKGA in the APDU.
- The resultant new DecoderKey value high order 32 bits are then used to replace the NKHO field of the Set1stSectionDecoderKey token (see 6.2.7) as presented by the APDU.
- The 2 Class bits are removed from the 66-bit token to yield a 64-bit result, which is then presented to the encryption algorithm as its DataBlock input. The specific encryption algorithm to use is in accordance with the EA code in the APDU.
- The KeyBlock input for the encryption algorithm is obtained from the decoder key generation algorithm, which generates the current DecoderKey using the current values of SGC, KRN, KT, TI, IIN and DRN from the APDU as indicated. The specific decoder key generation algorithm to use is in accordance with the value of DKGA in the APDU.
- After encryption, the 2 Class bits are again re-inserted into the 64-bit number in accordance with the method defined in 6.4.2 to yield a 66-bit result, which is populated into the TokenData field of the TCDU in accordance with the particular definition in the relevant physical layer protocol standard.
- Similarly, the TCT, IDRecord and PRNRecord data elements from the APDU are transferred to the TCDU as indicated, into the appropriate fields of the TCDU in accordance with the particular definition in the relevant physical layer protocol standard.

6.4.5 TCDUGeneration function for Set2ndSectionDecoderKey token

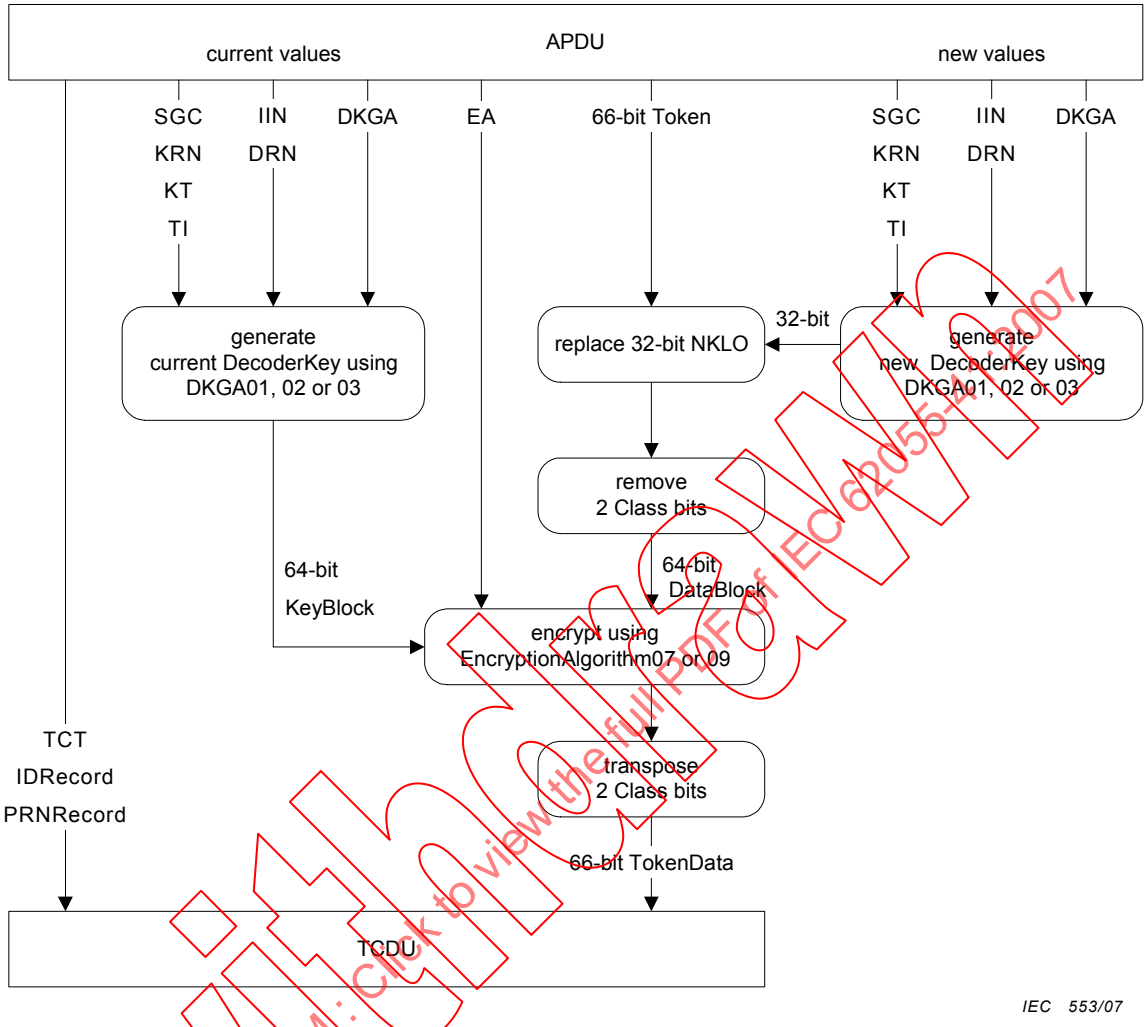


Figure 9 – TCDUGeneration function for Set2ndSectionDecoderKey token

This is the transfer function from the APDU to the TCDU (see Figure 9) and is applicable only to the Set2ndSectionDecoderKey token.

The Set2ndSectionDecoderKey TCDUGeneration function is shown here as being separate from the Set1stSectionDecoderKey TCDUGeneration function, but, in practice, the two may be merged into one in order to save on processing resource and for the sake of convenience. In such a case, the new DecoderKey generation only needs to happen once, for example, although the final result is still the same. Thus two separate TCDU instances are always produced: one for the Set1stSectionDecoderKey token and a second for the Set2ndSectionDecoderKey token.

It should be noted that the APDU shall present two sets of data for the PANBlock and CONTROLBlock: one set with the new data for the new DecoderKey and a second set with the current data for the current DecoderKey. The DKGA value is the same for both sets.

NOTE 1 The data elements in the APDU are defined in 6.1.1.

NOTE 2 The data elements in the TCDU are defined in each part of the IEC 62055-5x series physical layer protocol standard relevant to the specific TCT of interest.

The transfer function is outlined as follows.

- The new DecoderKey is generated using the new values of SGC, KRN, KT, TI, IIN and DRN. The specific decoder key generation algorithm to use is in accordance with the value of DKGA in the APDU.
- The resultant new DecoderKey value low order 32 bits are then used to replace the NKLO field of the Set2ndSectionDecoderKey token (see 6.2.8) as presented by the APDU.
- The 2 Class bits are removed from the 66-bit token to yield a 64-bit result, which is then presented to the encryption algorithm as its DataBlock input. The specific encryption algorithm to use is in accordance with the EA code in the APDU.
- The KeyBlock input for the encryption algorithm is obtained from the decoder key generation algorithm, which generates the current DecoderKey using the current values of SGC, KRN, KT, TI, IIN and DRN from the APDU as indicated. The specific decoder key generation algorithm to use is in accordance with the value of DKGA in the APDU.
- After encryption, the 2 Class bits are again re-inserted into the 64-bit number in accordance with the method defined in 6.4.2 to yield a 66-bit result, which is populated into the TokenData field of the TCDU in accordance with the particular definition in the relevant physical layer protocol standard.
- Similarly, the TCT, IDRecord and PRNRecord data elements from the APDU are transferred to the TCDU as indicated, into the appropriate fields of the TCDU in accordance with the particular definition in the relevant physical layer protocol standard.

6.5 Security functions

6.5.1 General requirements

With the exception of DITK values, VendingKey and DecoderKey values shall only be generated by a device responsible for token generation, such as a POS that is certified as STS-compliant and which is subject to an STS-certified KeyManagementSystem (see Clause 9). This subclause describes the key generation methods used by such devices and is applicable to manufacturers of these devices.

6.5.2 Key attributes and key changes

6.5.2.1 Key change requirements

With the exception of DITK values, STS key values shall only be introduced or changed in a payment meter from a device responsible for key management, such as a POS that is certified as STS-compliant and which is subject to STS key management. This clause describes the STS key change method used between such devices and payment meters, and is applicable to manufacturers of these devices and payment meters.

An STS key change provides the mechanism for changing the DecoderKey present in a decoder from its current value to a new value. This process may be initiated by several events or circumstances, including the following:

- a new or repaired payment meter that contains a manufacturer's DITK value shall be changed before leaving the manufacturing or repair premises to contain the appropriate value of manufacturer's default (DDTK) or utility's DecoderKey (DUTK or DCTK) depending on the SupplyGroup to which the payment meter has been allocated;
- a SupplyGroup's VendingKey has either expired or been compromised, and is replaced by a new VendingKey revision and, as a result, each DecoderKey within the SupplyGroup must be changed from its current DecoderKey value to the DecoderKey value that corresponds to the new VendingKey value;

- a payment meter is re-allocated from one SupplyGroup to another SupplyGroup and, as a result, its DecoderKey shall be changed from its current value generated from the previous SupplyGroup VendingKey to the new value generated from its new SupplyGroup VendingKey; or
- the TI for a payment meter is changed and, as a result, its DecoderKey shall be changed from its current value (that corresponds to the previous TI) to the new value (that corresponds to the new TI).

The Set1stSectionDecoderKey and Set2ndSectionDecoderKey token pair effects an STS key change. This meter-specific management token pair transfers the following information from the POS to the payment meter, encrypted under the current DecoderKey:

- the value of the new DecoderKey;
- the KEN;
- the KRN;
- the KT;
- the TI.

An STS key change for a payment meter shall be initiated automatically whenever any one of the following attributes of the VendingKey change in value:

- the value of the VendingKey;
- the value of the SGC;
- the value of the TI;
- the value of the KEN;
- the value of the KRN;
- the value of the KT.

NOTE See 6.1.1 for detailed specifications on the data elements in the APDU.

6.5.2.2 VendingKey classification

6.5.2.2.1 Classification of vending keys

The VendingKey is a DES key value that is secretly generated, stored and distributed within the KeyManagementSystem (see Annex A). DES VendingKeys are the seed keys from which DecoderKeys are generated.

The VendingKey is classified according to its associated KT value, which is an attribute that defines the purpose for which the key can be used. Three KT values are defined for VendingKeys and correspond to three of the SupplyGroup types (see 6.1.6), namely Default, Unique and Common. The VendingKey for a given SupplyGroup is the seed key used to generate the DecoderKey values for all payment meters within the SupplyGroup.

STS VendingKeys are classified according to the KT values given in Table 24.

Table 24 – Classification of vending keys

KT	SGC type	VendingKey type	Context
0	Initialization	Not specified	Not applicable
1	Default	VDDK	VendingDefaultDESKey
2	Unique	VUDK	VendingUniqueDESKey
3	Common	VCDK	VendingCommonDESKey

At any given moment, a unique VDDK value exists for each Default SupplyGroup defined. Similarly, a unique VUDK value for each Unique SupplyGroup and a unique VCDK value for each Common SupplyGroup are defined.

6.5.2.2.2 VDDK: VendingDefaultDESKey

This type of key is used as the seed key for generation of DDTK values – it shall not be used to generate DITK, DUTK or DCTK values.

6.5.2.2.3 VUDK: VendingUniqueDESKey

This type of key is used as the seed key for generation of DUTK values – it shall not be used to generate DITK, DDTK or DCTK values.

6.5.2.2.4 VCDK: VendingCommonDESKey

This type of key is used as the seed key for generation of DCTK values – it shall not be used to generate DITK, DDTK or DUTK values.

6.5.2.3 DecoderKey classification

6.5.2.3.1 Classification of decoder keys

STS DecoderKeys are classified according to the KT values given in Table 25 and inherit their type from that of the VendingKey, from which they are derived.

Table 25 – Classification of decoder keys

KT	SGC type	DecoderKey type	Context
0	Initialization	DITK	DecoderInitialisationTransferKey
1	Default	DDTK	DecoderDefaultTransferKey
2	Unique	DUTK	DecoderUniqueTransferKey
3	Common	DCTK	DecoderCommonTransferKey

For further information regarding the rules for changing of a key from one type to another type, see Figure 10 and Table 26 in 6.5.2.4.

A payment meter shall be capable of storing at least one DecoderKey value and its associated KT value in its DecoderKeyRegister (see 7.3.2).

It shall not be possible for the DecoderKey value to be read or retrieved from a payment meter under any circumstances, whether encrypted or in the clear.

6.5.2.3.2 DITK: DecoderInitialisationTransferKey

DITK values are used to initialize the DecoderKeyRegister during production or repair at the manufacturer's premises. These keys are the property of the MeterManufacturer. As such, they are generated and managed by the manufacturer and are unknown to the utility.

No payment meter purchased by the utility shall leave a manufacturer's premises with a DITK value in the DecoderKeyRegister. The DecoderKeyRegister shall contain either a DDTK, DUTK or DCTK value supplied by the KMC. A DITK is the only key type that can be introduced into a payment meter as a plaintext value. DDTK, DUTK or DCTK values can only be introduced into a payment meter as cipher text (encrypted) values.

A DITK shall only be used for the following key management functions:

- as the parent key for another DITK; in other words, to encrypt another DITK for the purpose of introducing it into the DecoderKeyRegister;
- as the parent key for a DDTK;
- as the parent key for a DUTK;
- as the parent key for a DCTK, but only in a payment meter using an erasable magnetic card as a token carrier (for TCT value = 01).

The above functions may be performed via the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens or via a manufacturer proprietary loading mechanism that utilizes the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens. The payment meter should only accept the DDTK, DUTK or DCTK encrypted under the DITK supplied by the manufacturer in the Set1stSectionDecoderKey and Set2ndSectionDecoderKey token format.

It is the responsibility of the manufacturer to ensure that appropriate security measures are applied to any DITK so that DDTK, DUTK or DCTK values encrypted with a DITK cannot be compromised.

A DITK can also be used to decrypt other meter-specific management functions. It can be used to decrypt an STS credit transfer function; in other words, a valid STS TransferCredit token can be decrypted and applied by a payment meter that contains a DITK in its key register in order to facilitate testing of the payment meter during production or repair.

6.5.2.3.3 DDTK: DecoderDefaultTransferKey

DDTK values are used to support payment meters allocated to a default SupplyGroup. A payment meter that has not been allocated to a Common SupplyGroup or a Unique SupplyGroup at the time of manufacture or repair cannot be loaded with its corresponding DCTK or DUTK value. Instead it is allocated to a Default group unique to each manufacturer and loaded with its corresponding DDTK value. Each MeterManufacturer receives a unique VDDK, from which he generates all DDTK values for installation into payment meters during manufacture.

Subsequently, at the time of installation or operation, a payment meter that has now been re-allocated to another specific SupplyGroup can be loaded with the corresponding DUTK or DCTK value, encrypted under its parent DDTK. DDTK values are the property of the respective MeterManufacturer and are managed within the KeyManagementSystem.

A DDTK is a secret value and shall not be accepted by a payment meter as a plaintext value. A payment meter shall only load a DDTK if it is encrypted under the parent DecoderKey present in the DecoderKeyRegister.

A DDTK shall only be used for the following key management functions:

- as the parent key for another DDTK; in other words, to encrypt another DDTK for the purpose of introducing it into the DecoderKeyRegister;
- as the parent key for a DUTK;
- as the parent key for a DCTK, but only in a payment meter using an erasable magnetic card as a token carrier (for TCT value = 01).

The above functions may be performed via the Set1stSectionDecoderKey and Set 2ndSectionDecoderKey tokens, or via a manufacturer's proprietary loading mechanism that utilizes the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens. A DDTK shall not be used to decrypt a DITK for the purpose of introducing it into the DecoderKeyRegister.

A DDTK can also be used to decrypt other meter-specific management functions. It shall not be used to decrypt and accept an STS credit transfer function; in other words, a valid TransferCredit token shall not be accepted by a payment meter that contains a DDTK in its DKR, even if the TransferCredit token has been encrypted with the same DDTK value.

NOTE The emphasis is on the acceptance and not not on the decryption of the TransferCredit token.

Similarly a POS device used for encrypting tokens shall not encrypt TransferCredit tokens using DDTK values (see also 6.5.2.4).

6.5.2.3.4 DUTK: DecoderUniqueTransferKey

DUTK values are used to support payment meters allocated to a unique SupplyGroup. A payment meter that has been allocated to a unique SupplyGroup at the time of manufacture or repair can be loaded with its DUTK value that corresponds to the unique group and that has been encrypted under a parent DITK. Subsequently, at the time of installation or operation, a payment meter, which has to be re-allocated to another unique group can be loaded with the corresponding DUTK value, encrypted under a parent DUTK.

A DUTK is a secret value, and shall not be accepted by a payment meter as a plaintext value. A payment meter shall only load a DUTK if it has been encrypted under the parent DecoderKey present in the DecoderKeyRegister. DUTK values are the property of the respective utility and are managed within the KeyManagementSystem.

A purchased or repaired payment meter that leaves the manufacturer's premises may contain a DUTK value supplied by the KMC in the DecoderKeyRegister.

A DUTK shall only be used for the following key management functions:

- as the parent key for another DUTK, in other words, to encrypt another DUTK for the purpose of introducing it into the DecoderKeyRegister;
- as the parent key for a DDTK.

The above functions may be performed via the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens, or via a manufacturer's proprietary loading mechanism that utilizes the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens. A DUTK shall not be used to decrypt a DITK or a DCTK for the purpose of loading it into the DecoderKeyRegister. Similarly a DUTK shall not be used to encrypt a DITK or a DCTK for the purpose of transferring it to the payment meter in the form of a token.

A DUTK can also be used to encrypt or decrypt other meter-specific management functions. It can be used to encrypt or decrypt a STS credit transfer function; in other words, a valid TransferCredit token can be encrypted or decrypted and applied by a payment meter that contains a DUTK in its DKR.

6.5.2.3.5 DCTK: DecoderCommonTransferKey

DCTK values are used to support payment meters that use erasable magnetic card token carriers (i.e. TCT value = 01) and that are allocated to common SupplyGroups. A payment meter that has been allocated to a common SupplyGroup at the time of manufacture or repair can be loaded with the DCTK value that corresponds to the common SupplyGroup and that has been encrypted under a parent DITK. Subsequently, at the time of installation or operation, a payment meter that has to be re-allocated to another common SupplyGroup can be loaded with the corresponding DCTK value that has been encrypted under a parent DCTK.

A DCTK shall only be used with payment meters that use erasable magnetic card token carriers (TCT value = 01) and shall only be accepted by such payment meters. Payment meters with any other token carrier types (TCT value > 01) shall reject tokens encrypted under DCTK values.

POS encryption devices shall not encrypt tokens using DCTK values other than for erasable magnetic card token carriers (TCT value = 01).

A DCTK is a secret value and shall not be accepted by a payment meter as a plaintext value. A payment meter shall only load a DCTK if it has been encrypted under the parent DecoderKey present in the DecoderKeyRegister. DCTK values are the property of the respective utility and are managed within the KeyManagementSystem.

A purchased or repaired payment meter with an erasable magnetic card token carrier (TCT value = 01) that leaves the manufacturer's premises may contain a DCTK value supplied by the KMC in the DecoderKeyRegister.

A DCTK shall only be used for the following key management functions:

- as the parent key for another DCTK; in other words, to encrypt another DCTK for the purpose of introducing it into the DecoderKeyRegister;
- as the parent key for a DDTK;
- as the parent key for a DUTK.

The above functions may be performed via the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens, or via a manufacturer's proprietary loading mechanism that utilizes the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens. A DCTK shall not be used to decrypt a DITK for the purpose of introducing it into the DecoderKeyRegister. Similarly a DCTK shall not be used to encrypt a DITK for the purpose of transferring it to the payment meter in the form of a token.

A DCTK can also be used to encrypt or decrypt other meter-specific management functions. It can be used to encrypt or decrypt a STS credit transfer function; in other words, a valid TransferCredit token can be encrypted or decrypted and applied by a payment meter that contains a DCTK in its DKR and that uses a magnetic card token carrier (TCT value = 01).

6.5.2.4 State diagram for DecoderKey changes

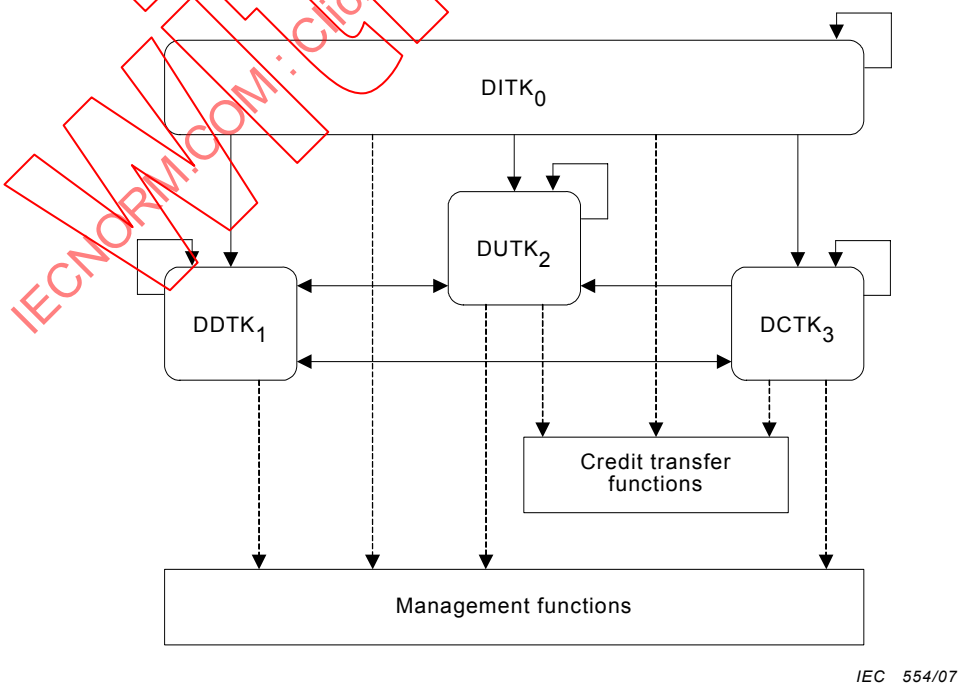


Figure 10 – DecoderKey changes – State diagram

Figure 10 illustrates the KT states that a DecoderKey may assume from time to time.

Where one key is used to encrypt another key (as in the Set1stSectionDecoderKey and Set2ndSectionDecoderKey token pair), the former is referred to as the parent key and the latter as the child key.

The solid line arrows indicate the direction in which a key may change from one type to another type. The type that it changes from is the parent key and the type that it changes to is the child key. To effect a change of the DecoderKey the new key (or child key) is encrypted with the parent key and then loaded into the payment meter by means of a Set1stSectionDecoderKey and Set2ndSectionDecoderKey token pair. The payment meter then replaces the parent key with the child key, which now becomes the new parent key.

The dotted line arrows indicate the function, for which a KT may be used, i.e. the values that it may encrypt or decrypt. For example, only a DITK, DUTK or DCTK can be used to encrypt or decrypt a credit transfer function, but all four types can be used to encrypt or decrypt meter-specific management functions.

Table 26 details the permitted key change state relationships and associated functions.

The child key rows refer to the permitted usage of decoder key types for encryption of DecoderKeys in the Set1stSectionDecoderKey and Set2ndSectionDecoderKey token management functions. Similarly, the management and credit rows detail the permitted usage of decoder key types for the encryption of the remaining meter-specific management functions and credit transfer functions respectively.

Table 26 – Permitted relationships between decoder key types

Child key	Permitted usage			
	Parent key			
	$DITK_0$	$DDTK_1$	$DUTK_2$	$DCTK_3$
$DITK_0$	Yes	No	No	No
$DDTK_1$	Yes	Yes	Yes	Yes[a]
$DUTK_2$	Yes	Yes	Yes	Yes[a]
$DCTK_3$	Yes[a]	Yes[a]	No	Yes[a]
Management function	Yes	Yes	Yes	Yes[a]
Credit function	Yes	No	Yes	Yes[a]

[a] For payment meters with TCT = 01 only.

6.5.2.5 KeyRevisionNumber (KRN)

A KRN is associated with each VendingKey and a corresponding SGC by the KMS and defines the revision or sequence of the VendingKey within the SupplyGroup to which it corresponds. It is a single decimal digit with a range of 1, 2..9. The KRN assigned to the first VendingKey for a SupplyGroup is 1. Successive VendingKeys are allocated successive revision numbers until revision number 9, at which stage the sequence begins at 1 again; in other words, at any given moment, there may be no more than 9 successive VendingKey revisions present for a given SupplyGroup. A KRN is also associated with each DecoderKey, and corresponds to that of the VendingKey from which it is generated.

The KRN is associated with each SupplyGroup by the KMS, and defines the current VendingKey revision and also the current DecoderKey revision, at which all payment meters within the SupplyGroup should be set. For any given payment meter, the SGC and KRN uniquely identify the revision of DecoderKey that it contains. This information is managed by the management system and if for any reason the KRN in the payment meter is not the same as the vending KRN for the same SGC as recorded in the management system, this condition shall be corrected by means of an appropriate change of the DecoderKey.

A payment meter is required to store the KRN that corresponds to its current DecoderKey, as passed in the Set1stSectionDecoderKey and Set2ndSectionDecoderKey token pair (see also 7.3.2).

The concept of key revision only applies to vending key types and decoder key types. A DITK shall not be associated with a KRN, and the KRN of a payment meter loaded with a DITK in the DecoderKeyRegister shall always be set to zero.

For a given SupplyGroup there shall be a maximum of two active VendingKeys in the POS namely the CurrentKey and the OldKey. The CurrentKey will be used to encrypt all tokens, apart from key change tokens; the OldKey will only be used to encrypt key change tokens.

6.5.2.6 KeyExpiryNumber (KEN)

A KEN is associated with each VendingKey by the KMS, and defines the following:

- the time-period, after which the VendingKey expires, and may no longer be used by a POS to generate DecoderKeys for the purpose of encrypting TransferCredit tokens, or meter-specific management tokens that incorporate the TID field;
- the time-period, after which any DecoderKey generated from the VendingKey expires, and may no longer be used by a payment meter to accept TransferCredit tokens, or meter-specific management tokens that incorporate the TID field. Implementation of this by a payment meter is optional.

The required value of the KEN shall be transferred to the payment meter in the KENHO and KENLO fields of the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens respectively (see 6.2.7 and 6.2.8).

The KEN is an 8-bit number (range 0 – 255) that expresses this period as a displacement relative to the STS base date token identifier time stamp (see 6.3.5.1). Each unit in the KEN corresponds to a period of duration 2^{16}-1 (65535) minutes, and there are 2^8 (256) of these periods numbered 0,1..255 before the current STS base date time stamp is replaced by the next STS base time stamp. Thus the KEN corresponds to the most significant 8 bits of the 24-bit TID. Any token identifier whose most significant 8 bits are greater than a given key's KEN shall not be encrypted or decrypted with that key.

A POS may not issue a TransferCredit token encrypted under a DecoderKey whose corresponding VendingKey has expired. This is simple to verify by comparing the most significant 8 bits of the TID with the KEN corresponding to the VendingKey; if it is greater, the VendingKey has expired and may no longer be used to generate a DecoderKey to encrypt the TransferCredit token. It also cannot be used to generate a DecoderKey to encrypt any meter-specific management tokens that utilize the TID field. This does not apply to the Set1stSectionDecoderKey and Set2ndSectionDecoderKey token pair that does not utilize the TID field. Hence, an expired DecoderKey can still be used to encrypt its replacement DecoderKey for the purpose of a DecoderKey change.

A payment meter can optionally implement key expiry and store the KEN that corresponds to its current DecoderKey, as passed in the Set1stSectionDecoderKey and Set2ndSectionDecoderKey token pair. All tokens that are entered into the payment meter, and that incorporate a token identifier field, are validated against this KEN. If the most significant 8 bits of the TID are greater than this KEN, the token shall be rejected.

Where implemented, the concept of key expiry only applies to VendingKey values of type VDDK, VUDK and VCDK, and DecoderKey values of type DDTK, DUTK and DCTK that can be generated from the corresponding vending key types. A DITK shall not be associated with a KEN, and the KEN of a payment meter loaded with a DITK in the DecoderKeyRegister shall always be set to 255.

The management of the KEN by the KMS shall comply with the relevant code of practice.

See also C.2.4 for the code of practice on managing this data element.

6.5.3 DecoderKey generation

6.5.3.1 PANBlock construction

The 64-bit PANBlock is constructed from data elements extracted from the MeterPAN in the APDU as defined in Table 27 and Table 28.

The most significant digit is in position 15 and the least significant digit in position 0.

Table 27 – Definition of the PANBlock

Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	I	I	I	I	I	D	D	D	D	D	D	D	D	D	D	D

Table 28 – Data elements in the PANBlock

Digit	Name	Format	Reference
I	IIN	Range 0 to 9 hex per digit	6.1.2.2
D	DRN	Range 0 to 9 hex per digit	6.1.2.3

Each digit is in hexadecimal format.

The PANBlock is made up of the 5 least significant digits of the IIN and the 11 digits of the DRN. The 11 digits of the DRN take up positions 10 to 0 in the PANBlock and the 5 least significant digits of the IIN take up positions 15 to 11 in the PANBlock.

If the IIN is of insufficient length to make up the 16 digits, the digits extracted are right justified within the block and padded on the left with zeroes (for example, for an IIN of 600727 and a DRN of 12345678903, the PANBlock is 0072712345678903).

For a DDTK or DUTK the actual designated DRN is used, but for a DCTK the DRN digits are set to zeros in the PANBlock (for example, for a IIN of 600727, the PANBlock is 0072700000000000).

6.5.3.2 CONTROLBlock construction

The 64-bit CONTROLBlock is constructed from the data elements in the APDU as defined in Table 29 and Table 30.

The most significant digit is in position 15 and the least significant digit in position 0.

Table 29 – Definition of the CONTROLBlock

Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	C	S	S	S	S	S	S	T	T	R	F	F	F	F	F	F

Table 30 – Data elements in the CONTROLBlock

Digit	Name	Format	Reference
C	KT digit	Range 0 to 3 hex per digit, 4 to F hex = reserved	6.1.9
S	SGC digit	Range 0 to 9 hex per digit	6.1.6
T	TariffIndex digit	Range 0 to 9 hex per digit	6.1.7
R	KRN digit	Range 0 to 9 hex per digit	6.1.8
F	Pad value digit	Always F hex per digit	x

Each digit is in hexadecimal format.

6.5.3.3 DKGA01: DecoderKeyGenerationAlgorithm01

This DecoderKeyGenerationAlgorithm01 is to be used on a small limited set of defined DRN values only. It is included in this standard to maintain backward compatibility with a limited number of legacy STS-compliant payment meters of an early generation also using the STA (EA code 07). The POSApplicationProcess gives the appropriate directive by means of the DKGA code in the APDU.

The DecoderKey is diversified from a 64-bit single DES VendingKey value.

This DecoderKeyGenerationAlgorithm01 is applicable to all payment meters that meet all of the following criteria:

- using IIN = 600727;
- and the KRN = 1;
- and the KT = 1 or 2 (default or unique);
- and the EA code 07 (STA);
- and the DRN falls within the ranges listed in Table 31.

Table 31 – Range of applicable decoder reference numbers

0109000000X	to	0109000499X
0100000000X	to	0100499999X
0300000000X	to	0311400000X
0400000000X	to	0405999999X
0601000000X	to	0603999999X
0640000000X	to	0641999999X
0666000000X	to	0669999999X
0699000001X	to	0699000999X
0700000000X	to	0702099999X
NOTE X is a check digit, the value of which varies in accordance with the value of the preceding 10 digits (see 6.1.2.3)		

This DecoderKeyGenerationAlgorithm01 is also applicable to all payment meters that meet all of the following criteria:

- using IIN = 600727;
- and the KRN = 1;
- and the KT = 3 (common);
- and the EA code 07 (STA);
- and coded with one of the SGC values listed in Table 32.

Table 32 – List of applicable supply group codes

100702
990400
990401
990402
990403
990404
990405

The process flow for the DKGA01 is shown in Figure 11.

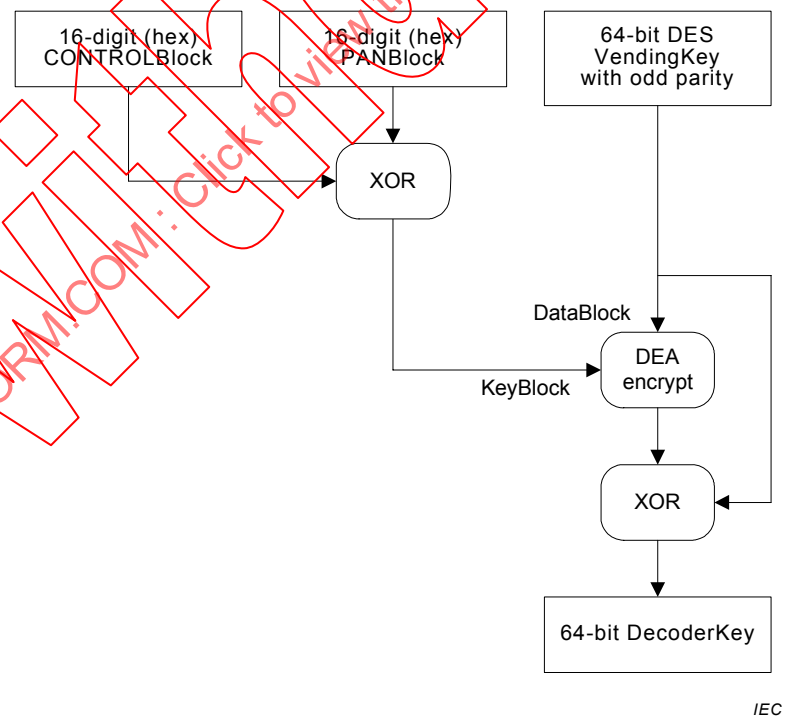


Figure 11 – DecoderKeyGenerationAlgorithm01

Construct the 64-bit PANBlock and the 64-bit CONTROLBlock as defined in 6.5.3.1 and 6.5.3.2.

The encryption algorithm is DEA in accordance with FIPS 46-3, single DES in ECB mode, using a a single 64-bit DES VendingKey with odd parity.

In this instance the 64-bit DES VendingKey is used as the conventional DataBlock input to the DEA, while the resultant XOR of the CONTROLBlock with the PANBlock is used as the conventional KeyBlock input to the DEA. In other words, the data and key input blocks are swapped with respect to the conventional configuration.

6.5.3.4 DKGA02: DecoderKeyGenerationAlgorithm02

The DecoderKeyGenerationAlgorithm02 may be used for all payment meters that do not meet the criteria for selecting DecoderKeyGenerationAlgorithm01. The POS ApplicationProcess gives the appropriate directive by means of the DKGA code in the APDU.

The DecoderKey is diversified from a 64-bit single DES VendingKey value.

The process flow for the DKGA02 is shown in Figure 12.

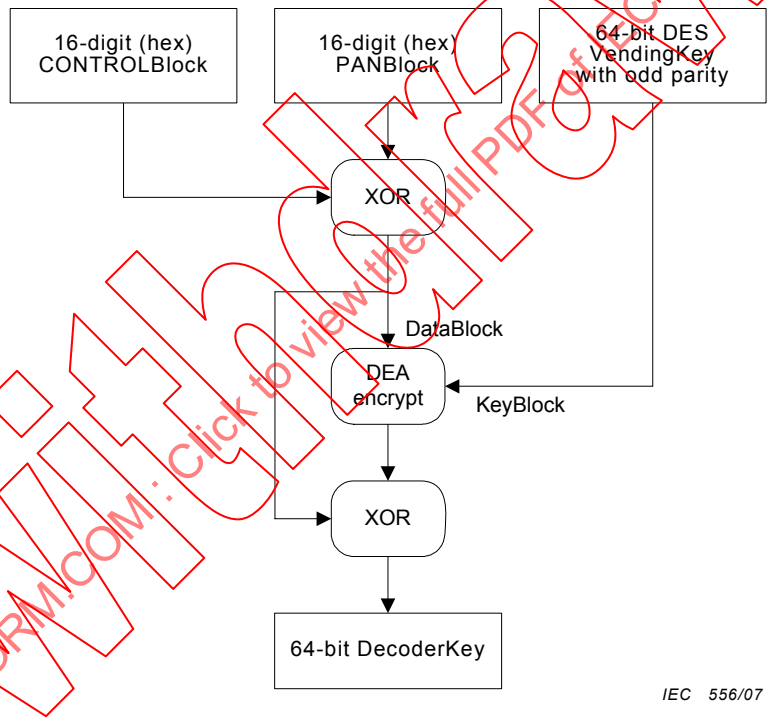


Figure 12 – DecoderKeyGenerationAlgorithm02

Construct the 64-bit PANBlock and the 64-bit CONTROLBlock as defined in 6.5.3.1 and 6.5.3.2.

Encryption is DEA in accordance with FIPS 46-3, single DES in ECB mode, using a single 64-bit DES VendingKey with odd parity.

6.5.3.5 DKGA03: DecoderKeyGenerationAlgorithm03

The DecoderKeyGenerationAlgorithm03 may be used for all payment meters that do not meet the criteria for selecting DecoderKeyGenerationAlgorithm01. The POSApplicationProcess gives the appropriate directive by means of the DKGA code in the APDU.

The DecoderKey is diversified from two 64-bit DES VendingKey values.

The process flow for the DKGA03 is shown in Figure 13.

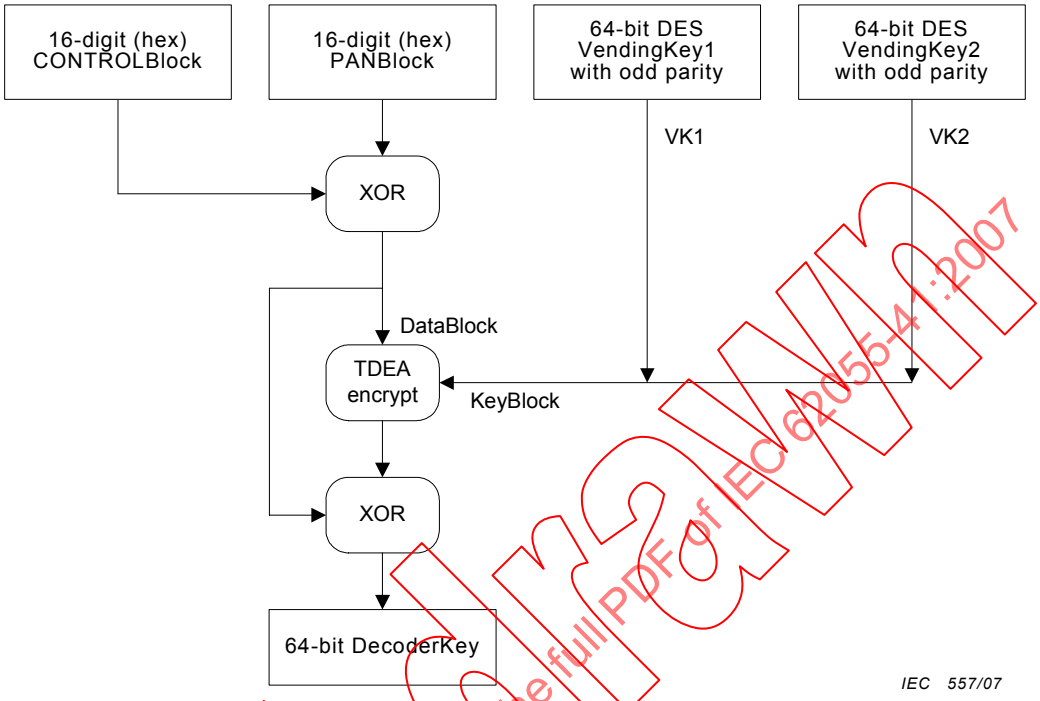


Figure 13 – DecoderKeyGenerationAlgorithm03

Construct the 64-bit PANBlock and the 64-bit CONTROLBlock as defined in 6.5.3.1 and 6.5.3.2.

Encryption is TDEA in accordance with FIPS 46-3, triple DES in ECB mode, using two 64-bit DES VendingKey values VK1 and VK2 with odd parity.

The operation is: encrypt with VK1, decrypt with VK2, encrypt with VK1.

6.5.4 STA: EncryptionAlgorithm07

6.5.4.1 Encryption process

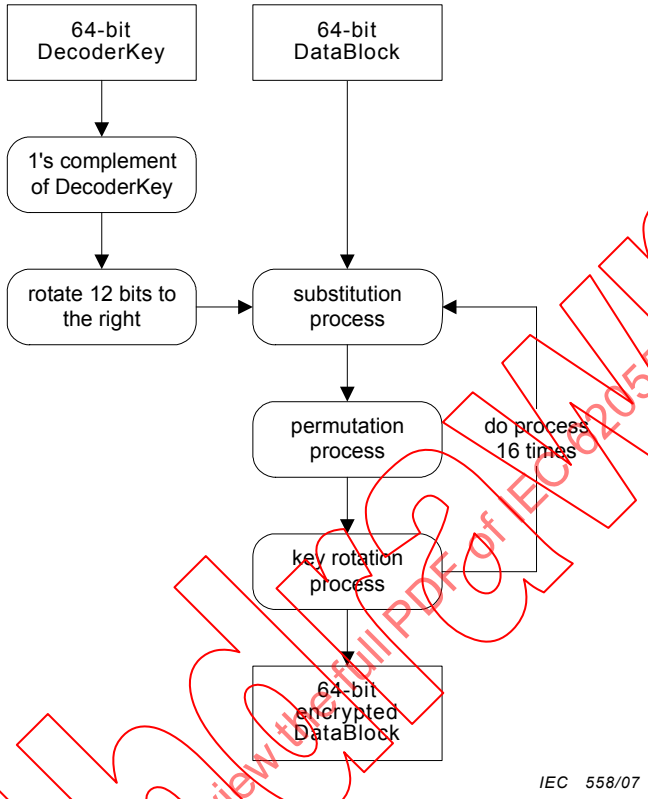


Figure 14 – STA: EncryptionAlgorithm07

The Standard Transfer Algorithm encryption process is shown in Figure 14, which comprises a key alignment process and 16 iterations of a substitution, permutation and key rotation process.

The POSApplicationProcess gives the appropriate directive by means of the EA code in the APDU.

6.5.4.2 Substitution process

The encryption substitution process is illustrated in Figure 15.

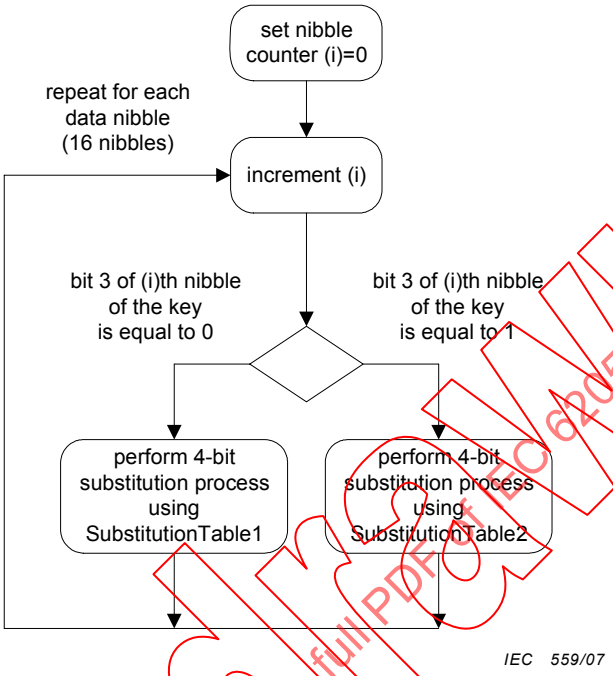


Figure 15 – STA encryption substitution process

There is a 4-bit substitution process for each of the 16 nibbles in the data stream. The substitution table used is one of two 16-value substitution tables and is dependent on the most significant bit setting of the corresponding nibble in the key. A sample substitution table is given in Table 33.

Table 33 – Sample substitution tables

SubstitutionTable1	12, 10, 8, 4, 3, 15, 0, 2, 14, 1, 5, 13, 6, 9, 7, 11
SubstitutionTable2	6, 9, 7, 4, 3, 10, 12, 14, 2, 13, 1, 15, 0, 11, 8, 5
NOTE This table contains only sample values (see Clause C.5 for access to table with actual values).	

The first entry in the substitution table corresponds to entry position 0 and the last to entry position 15.

Use the value of the data nibble as an index to an entry position in the substitution table; then replace the nibble value with the value from the substitution table found at that entry position. For example: if the value of the data nibble is 8 and we are using SubstitutionTable1, then the entry at position 8 is the value 14, thus replace the data nibble value with the value 14.

6.5.4.3 Permutation process

The encryption permutation process is illustrated in Figure 16.

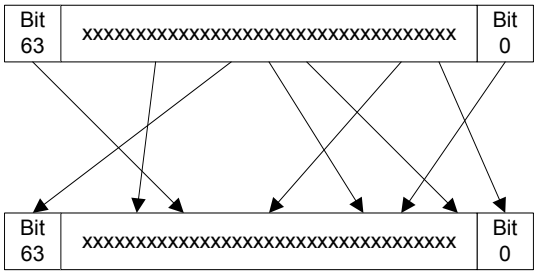


Figure 16 – STA encryption permutation process

A sample permutation table is given in Table 34.

Table 34 – Sample permutation table

PermutationTable3	29, 27, 34, 9, 16, 62, 55, 2, 40, 49, 38, 25, 33, 61, 30, 23, 1, 41, 21, 57, 42, 15, 5, 58, 19, 53, 22, 17, 48, 28, 24, 39, 3, 60, 36, 14, 11, 52, 54, 12, 31, 51, 10, 26, 0, 45, 37, 43, 44, 6, 59, 4, 7, 35, 56, 50, 13, 18, 32, 47, 46, 63, 20, 8
NOTE This table contains only sample values (see Clause C.5 for access to table with actual values).	

The first entry in the permutation table corresponds to the least significant bit position 0 in the DataBlock and the last entry to the most significant bit position 63 in the DataBlock.

Use the bit position of the source DataBlock as an index into the permutation table; then use the value found in the permutation table at that entry position as a pointer to the bit position in the destination DataBlock. For example: for the source DataBlock bit position 7 corresponds to the value 2 in the permutation table, thus the value of bit 7 from the source DataBlock is placed in bit position 2 in the destination DataBlock.

6.5.4.4 Key rotation process

The entire key is rotated one bit position to the left as illustrated in Figure 17.

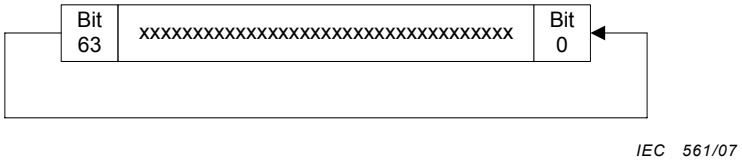


Figure 17 – STA encryption DecoderKey rotation process

6.5.4.5 Worked example to generate TokenData for a TransferCredit token using the STA

A worked example using the sample substitution and permutation tables is illustrated in Figure 18.

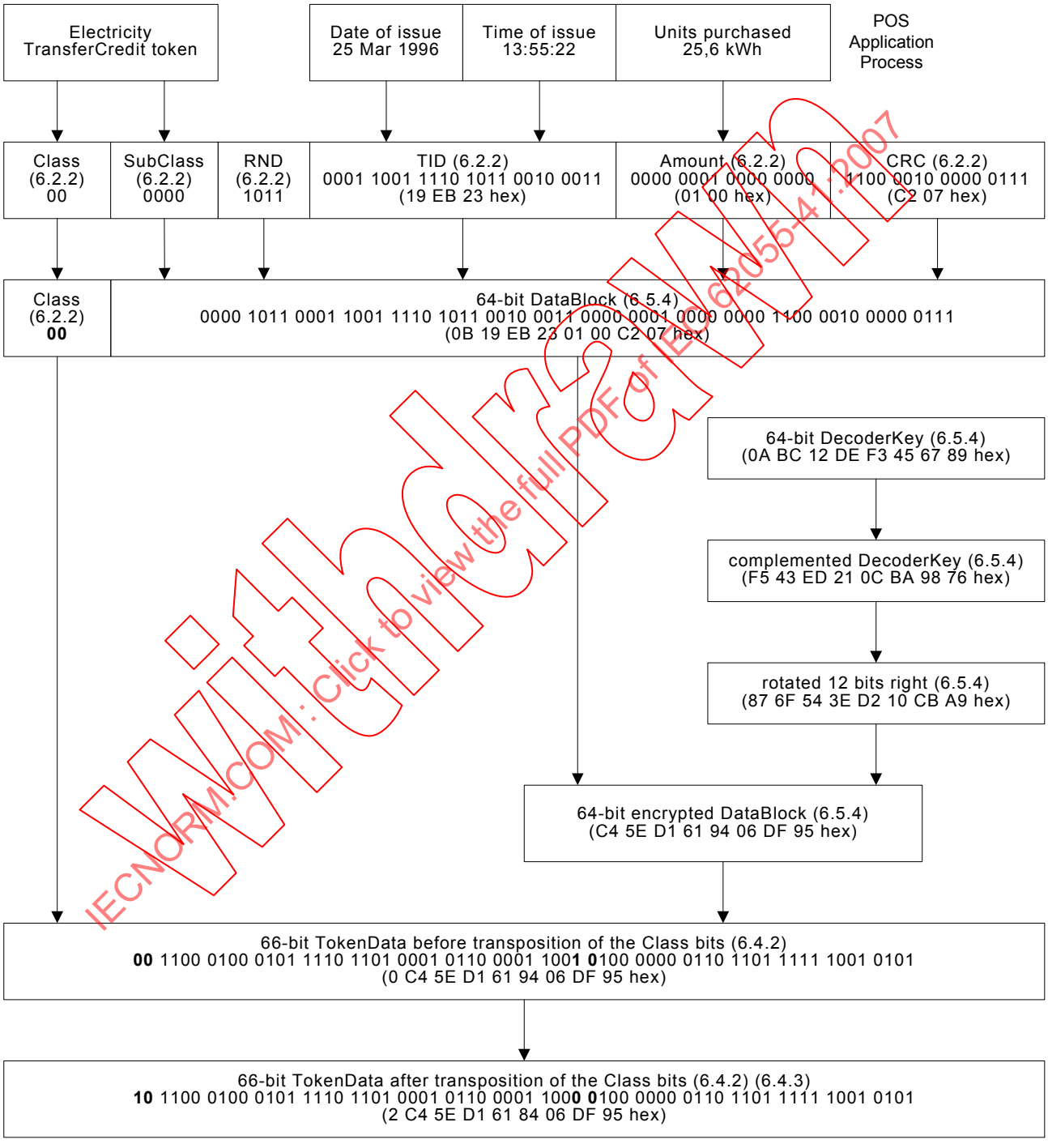


Figure 18 – STA encryption worked example for TransferCredit token

6.5.5 DEA: EncryptionAlgorithm09

The encryption process using the DEA is shown in Figure 19.

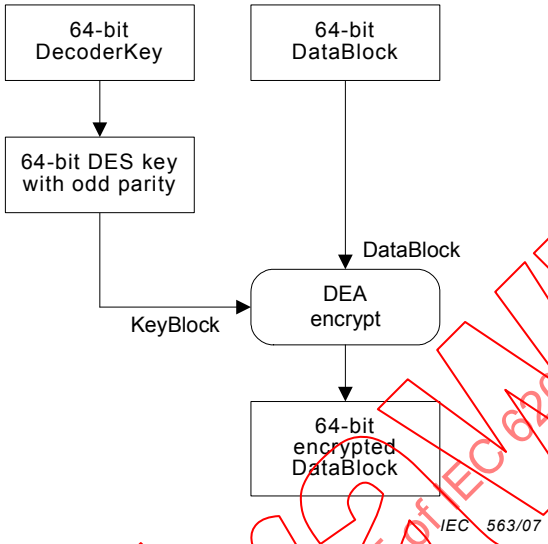


Figure 19 – DEA: EncryptionAlgorithm09

The DEA is a 64-bit block cipher in accordance with FIPS 46-3 operating in ECB mode. The POSApplicationProcess gives the appropriate directive by means of the EA code in the APDU.

The 64-bit DecoderKey is produced with DecoderKeyGenerationAlgorithm02 or with DecoderKeyGenerationAlgorithm03 (see 6.5.3.4 and 6.5.3.5).

The DecoderKey is converted into a 64-bit DES Key with odd parity in accordance with FIPS 46-3 by changing every 8th bit into a parity bit, starting with the least significant bit. Thus, bit 0, bit 8, bit 16, bit 24, bit 32, bit 40, bit 48 and bit 56 are converted into parity bits, where bit 0 is the least significant bit.

Encryption is DEA in accordance with FIPS 46-3, single DES in ECB mode, using a single 64-bit DES Key with odd parity.

7 TokenCarriertoMeterInterface application layer protocol

7.1 APDU: ApplicationProtocolDataUnit

7.1.1 Data elements in the APDU

The APDU is the data interface between the MeterApplicationProcess and the application layer protocol and comprises the data elements given in Table 35.

Table 35 – Data elements in the APDU

Element	Context	Format	Reference
Token	The TokenData from the TCDU after decryption and processing; now presented to the MeterApplicationProcess in the APDU	66 bits	7.1.2
AuthenticationResult	Status indicator to the MeterApplicationProcess to convey the result from the initial authentication checks		7.1.3
ValidationResult	Status indicator to the MeterApplicationProcess to convey the result from the initial validation checks		7.1.4
TokenResult	Status indicator from the MeterApplicationProcess to convey the result after processing the token so that the application layer protocol can take the appropriate action		7.1.5

7.1.2 Token

The TokenData from the TCDU after decryption and processing; now presented to the MeterApplicationProcess in the APDU.

The actual 66-bit token as originally entered into the APDU by the MeterApplicationProcess. The MeterApplicationProcess is now able to process it further. See 6.2.1 for the detailed definition of this data element.

7.1.3 AuthenticationResult

A status indicator to tell the MeterApplicationProcess that the initial authentication checks (see 7.3.5) passed or failed, in order that the MeterApplicationProcess can respond appropriately. Possible values are given in Table 36.

Table 36 – Possible values for the AuthenticationResult

Value	Context	Format	Reference
Authentic	The authentication test passed or failed False if any one of the below error codes is indicated Tue if none of the below error codes is indicated	Boolean	7.3.5
CRCError	The CRC value in the token is different to the CRC value as calculated from the data in the token	Boolean	7.3.5
MfrCodeError	The MfrCode value in the Class 1 token does not match the MfrCode value for the Decoder	Boolean	7.3.5

7.1.4 ValidationResult

A status indicator to tell the MeterApplicationProcess that the initial validation checks (see 7.3.6) passed or failed, in order that the MeterApplicationProcess can respond appropriately. Possible values are given in Table 37.

Table 37 – Possible values for the ValidationResult

Value	Context	Format	Reference
Valid	The Validation test passed or failed False if any one of the below error codes is indicated Tue if none of the below error codes is indicated	Boolean	7.3.6
OldError	The TID value as recorded in the token is older than the oldest value of recorded values recorded in the memory store of the payment meter	Boolean	7.3.6
UsedError	The TID value as recorded in the token is already recorded in the memory store of the payment meter	Boolean	7.3.6
KeyExpiredError	The TID value as recorded in the token is larger than the KEN stored in the payment meter memory	Boolean	7.3.6
DDTKError	The Decoder has a DDTK value in the DKR; a TransferCredit token may not be processed by the MeterApplicationProcess in accordance with the rules given in 6.5.2.3.3	Boolean	7.3.6

7.1.5 TokenResult

After the MeterApplicationProcess has executed the instruction contained in the token, the TokenResult value reflects the outcome. The application layer protocol may then take the appropriate action to complete the token reading process, which may include accepting the token (and storing of the TID), rejection of the token, erasure of token data from the TokenCarrier, etc. Possible values are given in Table 38.

Table 38 – Possible values for the TokenResult

Value	Context	Format	Reference
Accept	The token was successfully processed False if any one of the below error codes is indicated True if none of the below error codes is indicated	Boolean	8.2
1stKCT	The MeterApplicationProcess indicates that this is the Set1stSectionDecoderKey token of the pair of key change tokens being read; the token is provisionally accepted	Boolean	8.2
2ndKCT	The MeterApplicationProcess indicates that this is the Set2ndSectionDecoderKey token of the pair of key change tokens being read; the token is provisionally accepted	Boolean	8.2
OverflowError	The credit register in the payment meter would overflow if the token were to be accepted; the token is not accepted	Boolean	8.2
KeyTypeError	The key may not be changed to this type in accordance with the key change rules given in 6.5.2.4	Boolean	8.2
FormatError	One or more data elements in the token does not comply with the required format for that element	Boolean	6.2
RangeError	One or more data elements in the token have a value that is outside of the defined range of values defined in the application for that element	Boolean	6.3
FunctionError	The particular function to execute the token is not implemented	Boolean	8.2

7.2 APDUExtraction functions

7.2.1 Extraction process

The process of extracting the APDU from the TCDU is shown in Figure 20.

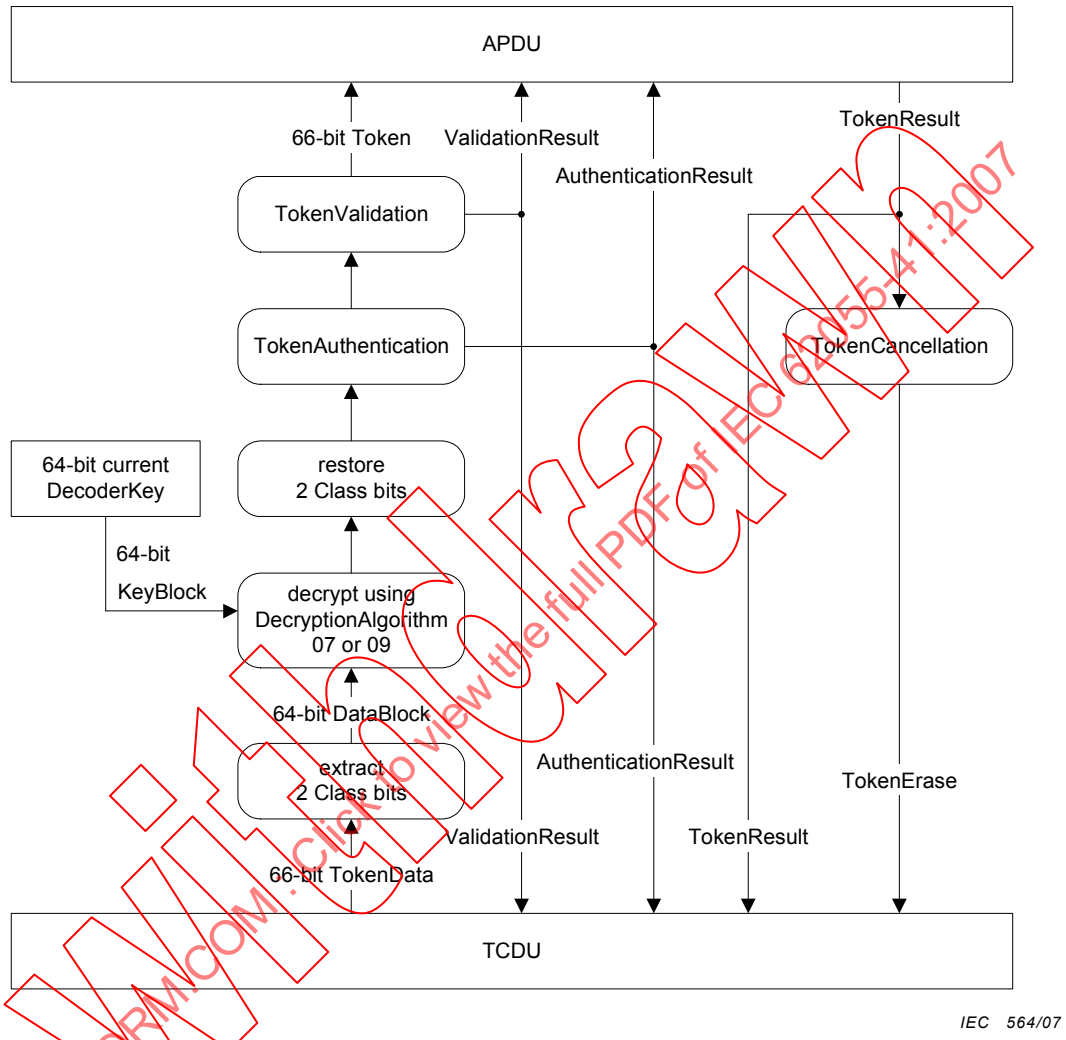


Figure 20 – APDUExtraction function

The APDUExtraction function extracts the 66-bit TokenData from the TCDU, decrypts and processes it before presenting the result in the APDU to the MeterApplicationProcess. It finally cancels and optionally causes the token data to be erased from the TokenCarrier in response to the result from the MeterApplicationProcess.

7.2.2 Extraction of the 2 Class bits

This function is used by other APDUExtraction functions (see 7.2.3 to 7.2.5). It removes the 2 Class bits from the 66-bit data stream to make a 64-bit number according to the method outlined in Figure 21 and is the inverse of 6.4.2.

The 66-bit number has its least significant bit in bit position 0 and its most significant bit in bit position 65. The 2-bit token Class value is extracted from bit positions 28 and 27. The values of bit positions 65 and 64 are relocated to bit positions 28 and 27. The most significant bit of the token Class comes from original bit position 28.

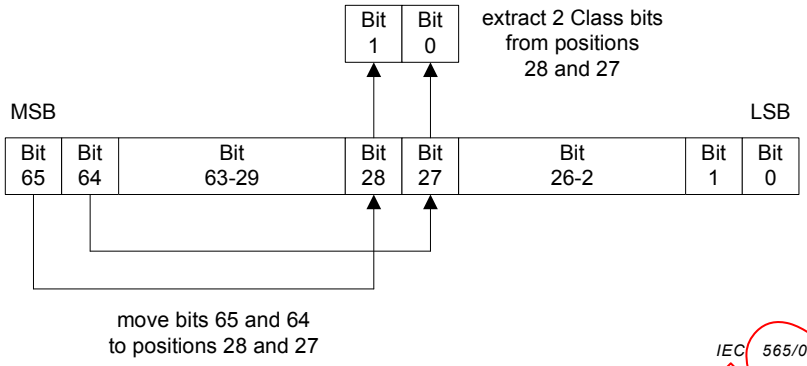


Figure 21 – Extraction of the 2 Class bits

Example: Extraction of the token Class = 01 (binary)

Extract the 2 Class bits from bit positions 28 and 27 (in bold):

00 0110 0101 0100 0011 0010 0001 0000 1001 1000 **1**111 0110 0101 0100 0011 0010 0001

Move bits 65 and 64 into bit positions 28 and 27 (in bold):

00 0110 0101 0100 0011 0010 0001 0000 1001 1000 **0**111 0110 0101 0100 0011 0010 0001

The resultant 64-bit binary number grouped in nibbles: (Bits 27 and 28 highlighted in bold.)

0110 0101 0100 0011 0010 0001 0000 1001 1000 **0**111 0110 0101 0100 0011 0010 0001

7.2.3 APDUExtraction function for Class 0 and Class 2 tokens

This is the transfer function from the TCDU to the APDU and is applicable to all Class 0 and 2 tokens, except for the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens (see 7.2.5).

NOTE 1 The data elements in the APDU are defined in 7.1.1.

NOTE 2 The data elements in the TCDU are defined in each part of the IEC 62055-5x series physical layer protocol standard relevant to the specific TCT of interest.

The transfer function for Class 0 and Class 2 tokens is outlined as follows.

- The 2 Class bits are extracted from the 66-bit TokenData using the method in 7.2.2 to yield a 64-bit result, which is then presented to the decryption algorithm as its DataBlock input. Note that it is the responsibility of the POS to keep record of which specific decryption algorithm is in use in each particular payment meter (see 6.1.5 EA). The decryption algorithm and encryption algorithm are complementary and thus share the same EA code.
- The KeyBlock input for the decryption algorithm contains the current value of the DecoderKey, which is obtained from the DecoderKeyRegister in the payment meter secure memory.
- After decryption the 2 Class bits are again re-inserted into the 64-bit number to make a 66-bit number. The most significant bit of the 2 Class bits goes into bit position 65 and the least significant Class bit goes into bit position 64.

- The 66-bit token is authenticated in accordance with 7.3.5 and the result is indicated in the AuthenticationResult field of the APDU.
- The 66-bit token is validated in accordance with 7.3.6 and the result is indicated in the ValidationResult field of the APDU and the 66-bit token is placed in the Token field of the APDU.
- The MeterApplicationProcess processes the Token from the APDU and indicates the result in the TokenResult field of the APDU (see also 8.2). It is the responsibility of the MeterApplicationProcess to deal with display messages and indicators (see also 8.3) to the user and not the application layer protocol.
- If the TokenResult indicates Accept (see 7.1.5 and 8.2), then the Token is cancelled in accordance with 7.3.7 and the instruction is given in the TokenErase field of the TCDU to erase the data from the TokenCarrier.

NOTE It is the responsibility of the physical layer protocol to decide whether the erase instruction is applicable or not in accordance with its specific implementation and TCT (see, for example, Clause 6 of IEC 62055-51).

7.2.4 APDUExtraction function for Class 1 tokens

The APDUExtraction function for Class 1 tokens is identical to that of the Class 0 and Class 2 tokens, except that the decryption step is not performed.

7.2.5 APDUExtraction function for Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens

This is the transfer function from the TCDU to the APDU and is applicable to the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens.

NOTE 1 The data elements in the APDU are defined in 7.1.1.

NOTE 2 The data elements in the TCDU are defined in each part of the IEC 62055-5x series physical layer protocol standard relevant to the specific TCT of interest.

The transfer function for Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens is outlined as follows.

- The 2 Class bits are extracted from the 66-bit TokenData using the method in 7.2.2 to yield a 64-bit result, which is then presented to the decryption algorithm as its DataBlock input. Note that it is the responsibility of the POS to keep record of which specific decryption algorithm is in use in each particular payment meter (see 6.1.5 EA). The decryption algorithm and encryption algorithm are complementary and thus share the same EA code.
- The KeyBlock input for the decryption algorithm contains the current value of the DecoderKey, which is obtained from the DecoderKeyRegister in the payment meter secure memory.
- After decryption, the 2 Class bits are again re-inserted into the 64-bit number to make a 66-bit number. The most significant bit of the 2 Class bits goes into bit position 65 and the least significant Class bit goes into bit position 64.
- The 66-bit token is authenticated in accordance with 7.3.5 and the result is indicated in the AuthenticationResult field of the APDU.
- The 66-bit token is not validated in the application layer protocol, but only in the MeterApplicationProcess. The 66-bit token is placed in the Token field of the APDU.
- The MeterApplicationProcess processes the Token from the APDU and indicates the result in the TokenResult field of the APDU (see also 8.2). It is the responsibility of the MeterApplicationProcess to deal with display messages and indicators (see also 8.3) to the user and not the application layer protocol.
- If the TokenResult indicates 1stKCT or 2ndKCT (see 7.1.5 and 8.2), then the instruction to erase the data from the TokenCarrier is not given in the TokenErase field of the TCDU.

- If the TokenResult indicates Accept (see 7.1.5 and 8.2) then the instruction to erase the data from the TokenCarrier is given in the TokenErase field of the TCDU.

The Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens may be entered in any order (see 8.9), but only the last one shall be erased.

NOTE It is the responsibility of the physical layer protocol to decide whether the erase instruction is applicable or not, in accordance with its specific implementation and TCT (see, for example, Clause 6 of IEC 62055-51).

7.3 Security functions

7.3.1 Key attributes and key changes

7.3.1.1 Key change requirements

The payment meter shall comply with the relevant requirements of 6.5.2, 7.3.1.2 and 7.3.1.3.

7.3.1.2 Key change processing without key expiry

The following defines the key change processing required if key expiry is not implemented in the payment meter.

- Compare the KT value on the token against the KT value in the payment meter:
  - if KT values are equal, change the DecoderKeyRegister content, decoder KRN and payment meter TI to the corresponding new values on the token;
  - if KT values are not equal, validate KT rules (see 6.5.2.4):
    - if key change is allowed, change the DecoderKeyRegister content, decoder KRN, decoder KT and payment meter TI to the corresponding new values on the token;
    - if key change is not allowed, reject the key change operation.

7.3.1.3 Key change processing with key expiry

The following defines the key change processing required if key expiry is implemented in the payment meter:

- Compare the token KT value against the decoder KT value:
  - if KT values are equal, change the DecoderKeyRegister content, decoder KEN, decoder KRN and payment meter TI to the corresponding token values;
  - if KT values are not equal, validate KT rules (see 6.5.2.4):
    - if key change is allowed, change the DecoderKeyRegister content, decoder KEN, decoder KRN, decoder KT and payment meter TI to the corresponding token values;
    - if key change not is allowed, reject the key change operation.

7.3.2 DKR: DecoderKeyRegister

The payment meter shall store the values given in Table 39 in secure non-volatile memory.

Table 39 – Values stored in the DKR

Value	Reference
DecoderKey	6.5.2.3 6.5.3
TI	6.1.7
KRN	6.1.8
KT	6.1.9
KEN (optional)	6.1.10
NOTE The TI may be associated with a tariff table that is managed outside of the domain of the payment meter. This implies that should a utility make use of the association, then the payment meter would require a key change each time that the customer is associated with a different tariff structure.	

In all cases where the payment meter provides configuration information, the KT shall be considered part of the KeyRevisionNumber information. The payment meter shall therefore always provide the KT information together with, or else directly after, the KRN information.

7.3.3 STA: DecryptionAlgorithm07

7.3.3.1 Decryption process

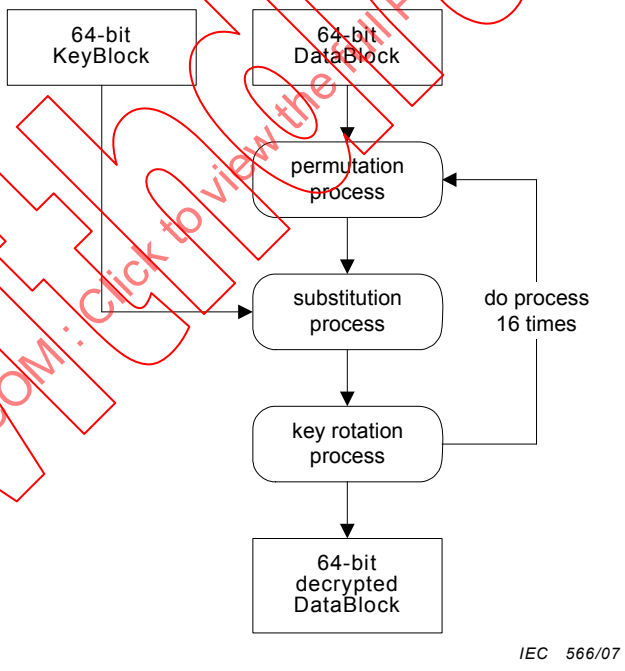


Figure 22 – STA DecryptionAlgorithm07

The Standard Transfer Algorithm decryption process is shown in Figure 22 and comprises a key alignment process and 16 iterations of a permutation, substitution and key rotation process.

The decryption algorithm and encryption algorithm are complementary and thus share the same EA code.

7.3.3.2 Permutation process

The decryption permutation process is illustrated in Figure 23.

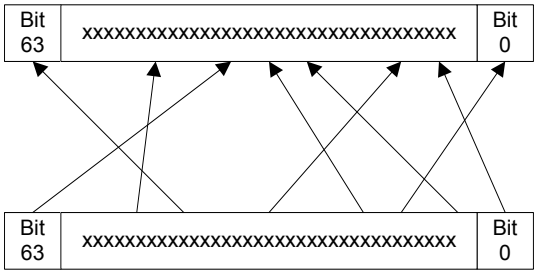


Figure 23 – STA decryption permutation process

A sample permutation table is given in Table 40.

Table 40 – Sample permutation table

PermutationTable4	44, 16, 7, 32, 51, 22, 49, 52, 63, 3, 42, 36, 39, 56, 35, 21, 4, 27, 57, 24, 62, 18, 26, 15, 30, 11, 43, 1, 29, 0, 14, 40, 58, 12, 2, 53, 34, 46, 10, 31, 8, 17, 20, 47, 48, 45, 60, 59, 28, 9, 55, 41, 37, 25, 38, 6, 54, 19, 23, 50, 33, 13, 5, 61

NOTE This table contains only sample values (see Clause C.5 for access to table with actual values).

The first entry in the permutation table corresponds to the least significant bit position 0 in the DataBlock and the last entry to the most significant bit position 63 in the DataBlock.

Use the bit position of the source DataBlock as an index into the permutation table; then use the value found in the permutation table at that entry position as a pointer to the bit position in the destination DataBlock. For example, for the source DataBlock bit position 7 corresponds to the value 52 in the permutation table, thus the value of bit 7 from the source DataBlock is placed in bit position 52 in the destination DataBlock.

It can be seen that this gives the inverse result of the process in 6.5.4.3.

7.3.3.3 Substitution process

The decryption substitution process is illustrated in Figure 24.

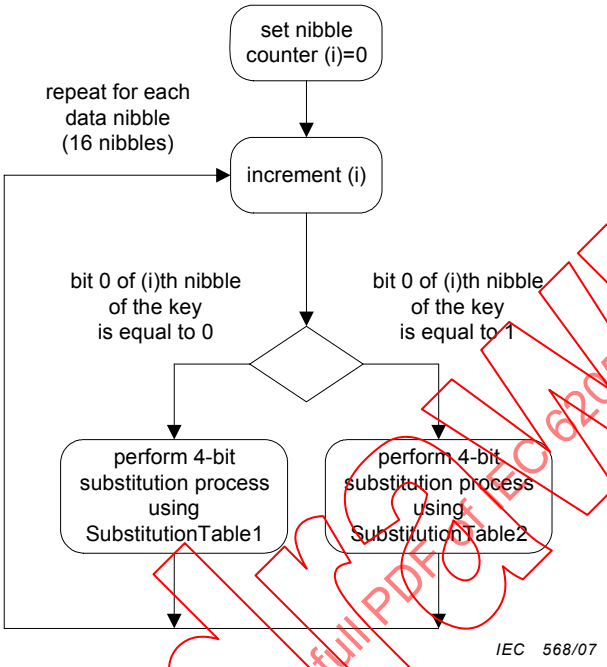


Figure 24 – STA decryption substitution process

There is a 4-bit substitution process for each of the 16 nibbles in the data stream. The substitution table used is one of two 16-value substitution tables and is dependent on the least significant bit setting of the corresponding nibble in the key. A sample substitution table is given in Table 41.

Table 41 – Sample substitution tables

SubstitutionTable1	12, 10, 8, 4, 3, 15, 0, 2, 14, 1, 5, 13, 6, 9, 7, 11
SubstitutionTable2	6, 9, 7, 4, 3, 10, 12, 14, 2, 13, 1, 15, 0, 11, 8, 5

NOTE This table contains only sample values (see Clause C.5 for access to table with actual values).

The first entry in the substitution table corresponds to entry position 0 and the last to entry position 15.

Use the value of the data nibble as an index to an entry position in the substitution table; then replace the nibble value with the value from the substitution table found at that entry position. For example, if the value of the data nibble is 8 and SubstitutionTable1 is being used, then the entry at position 8 is the value 14, thus the data nibble value is replaced with the value 14.

It can be seen that this gives the inverse result of the process in 6.5.4.2.

7.3.3.4 Key rotation process

The entire key is rotated one bit position to the right as illustrated in Figure 25.

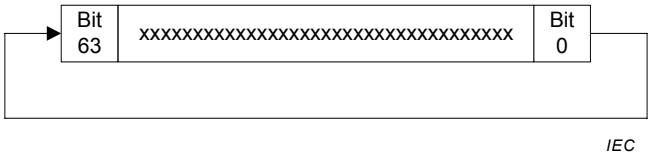


Figure 25 – STA decryption DecoderKey rotation process

7.3.3.5 Worked example to decrypt a TransferCredit token using the STA

A worked example using the sample substitution and permutation tables is illustrated in Figure 26.

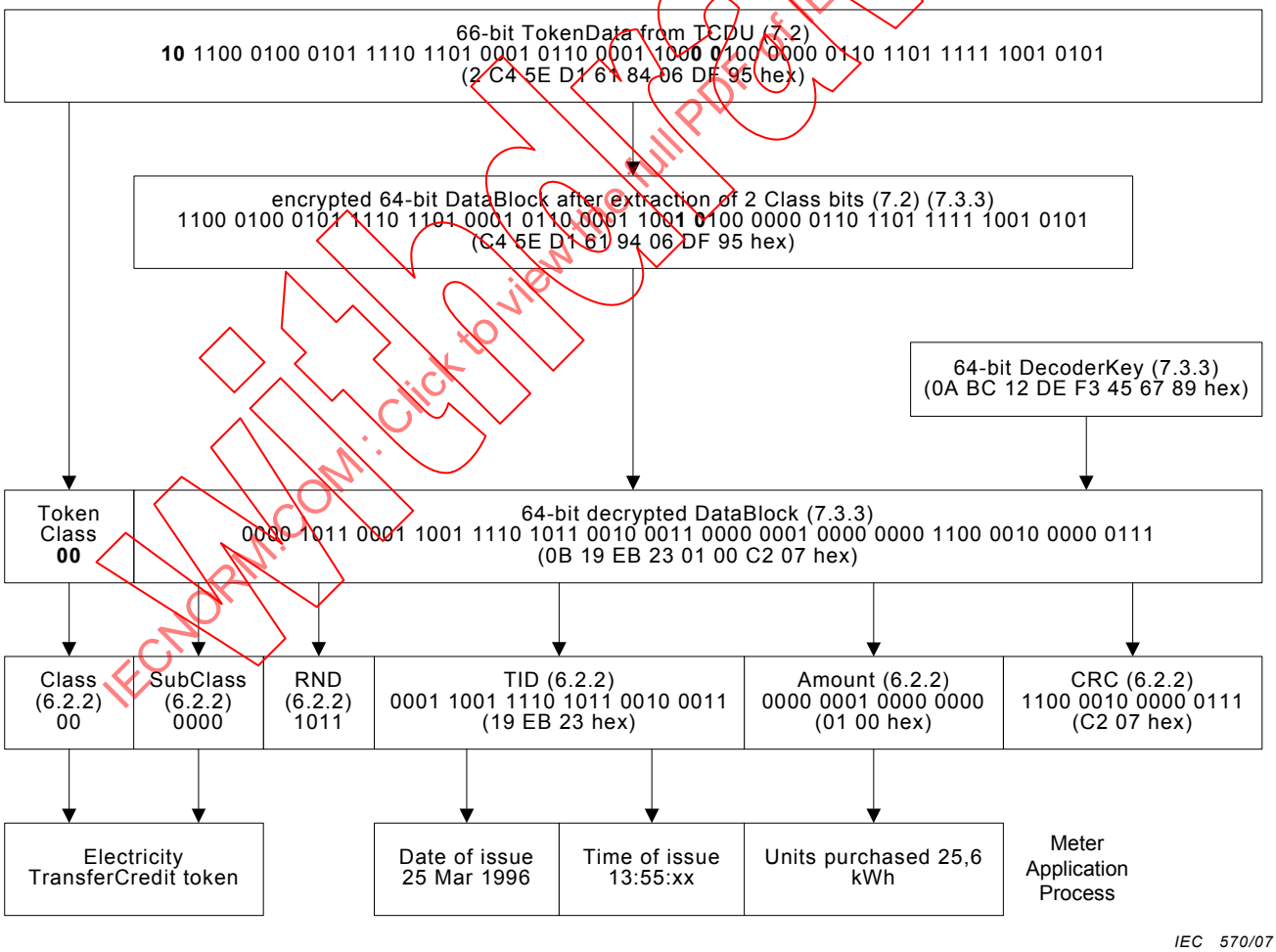


Figure 26 – STA decryption worked example for TransferCredit token

7.3.4 DEA: DecryptionAlgorithm09

The decryption process using the DEA is shown in Figure 27.

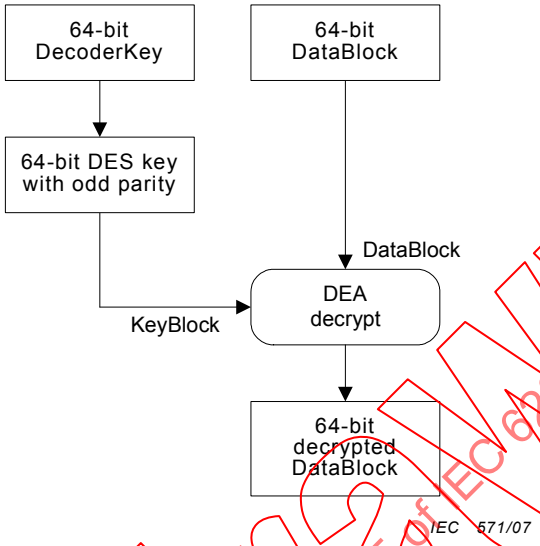


Figure 27 – DEA DecryptionAlgorithm09

The DEA is a 64-bit block cipher in accordance with FIPS 46-3 operating in ECB mode.

The DecoderKey is converted into a 64-bit DES Key with odd parity in accordance with FIPS 46-3 by changing every eighth bit into a parity bit, starting with the least significant bit. Thus, bit 0, bit 8, bit 16, bit 24, bit 32, bit 40, bit 48 and bit 56 are converted into parity bits, where bit 0 is the least significant bit.

The decryption algorithm and encryption algorithm are complementary and thus share the same EA code.

Decryption is DEA in accordance with FIPS 46-3, single DES in ECB mode, using a single 64-bit DES Key with odd parity.

7.3.5 TokenAuthentication

Validating the CRC checksum after decryption shall authenticate Class 0 and Class 2 tokens.

Validating the CRC and the MfrCode shall authenticate Class 1 tokens.

In the case of a Class 0 or a Class 2 token the AuthenticationResult status shall indicate Authentic when the following condition is met.

- The CRC checksum in the token has the same value as that calculated from the data elements in the token.

If the above condition is not met, then the AuthenticationResult status shall indicate CRCError.

In the case of a Class 1 token the AuthenticationResult status shall indicate Authentic when both of the following conditions are met.

- The CRC checksum in the token has the same value as that calculated from the data elements in the token.
- The MfrCode value in the token is the same as the MfrCode for the Decoder.

If any of the above conditions are not met, then the AuthenticationResult status shall indicate CRCError, or MfrCodeError, or both.

If the token cannot be authenticated, it shall be rejected in accordance with the requirements given in 8.2 and 8.3.

7.3.6 TokenValidation

Class 0 and Class 2 tokens shall primarily be validated against the TID encoded in the token, except for Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens.

Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens are validated by the MeterApplicationProcess once the payment meter has read both tokens and combined them into the new DecoderKey. See 8.2 for validation requirements of the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens.

If key expiry is implemented in the payment meter, then the KEN stored in the payment meter shall also be used to validate tokens of Class 0 and Class 2 (see 6.5.2.6.), except for Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens.

A status of Valid shall be indicated if none of the following conditions are true.

- If a TID is received that has a value smaller than the smallest value of TID stored in the memory store (in other words, that was issued by a POS on a date before the earliest TID stored in the memory store), then such token containing this TID shall be rejected and indicate such condition as an OldError status (see 7.1.4).
- If a TID is received that is already stored in the memory store (see 7.3.7), the token shall be rejected and indicate such condition as a UsedError status (see 7.1.4).
- If key expiry is implemented in the payment meter and a TID is received that is greater than the KEN in the Decoder, the token shall be rejected and indicate such condition as a KeyExpiredError status (see 7.1.4).
- If a Class 0 token is presented to the Decoder with a DDTK value in the DKR, the token shall be rejected (see 6.5.2.3.3) and indicate such condition as a DDTKError status (see 7.1.4).

See also 8.2 and 8.3 for acceptance, rejection and indication requirements in the MeterApplicationProcess.

A payment meter loaded with a DDTK value shall accept all the relevant "non-meter-specific management tokens" (Class 1 tokens) as well as Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens encrypted under a DDTK.

7.3.7 TokenCancellation

Cancellation of a token shall be by means of storing the TID associated with that token in a secure non-volatile memory store in addition to erasure of the token data record from magnetic card token carriers (see 6.1.3 and 6.2.5 of IEC 62055-51).

A time-based TID is used to uniquely identify each Class 0 and Class 2 token (except for the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens). The payment meter shall store, in a secure non-volatile memory store, at least the last 50 TID values received.

If a valid token is received with a TID that has a value greater than the smallest value of TID value in the memory store and there is no available space in the memory store to store the received TID value, the payment meter shall accept this token, remove the smallest TID value (in other words, the oldest TID) from the memory store, and replace it with the new TID value.

If the payment meter accepts a Set1stSectionDecoderKey and Set2ndSectionDecoderKey token pair, the TID memory store shall remain unchanged, unless the RolloverKeyChange (see 6.3.18) field specifies that the memory store shall be cleared.

The payment meter shall not accept tokens that were created prior to the date of manufacture or repair of the payment meter.

NOTE A suggested method is for the manufacturer to fill the TID memory store with values that indicate the date and time of manufacture or repair.

The payment meter shall read and process a token (as well as erase it when required) on a single insertion of the TokenCarrier without further action from the user.

All payment meters operating with a DCTK (see 6.5.2.3.1) shall erase token data (Class 0 and Class 2 tokens) from the TokenCarrier after successful transfer of the token data from the TokenCarrier to the payment meter, with the exception of the Set1stSectionDecoderKey token data and Set2ndSectionDecoderKey token data.

The following tokens shall not be erased:

- any token carrying a TID which is judged by the payment meter as being old;
- "non-meter-specific management tokens" of Class 1;
- the Set1stSectionDecoderKey or a Set2ndSectionDecoderKey token, whichever is inserted first.

The Set1stSectionDecoderKey or a Set2ndSectionDecoderKey token, whichever is inserted last, shall be erased upon successful completion of the key change operation.

8 MeterApplicationProcess requirements

8.1 General requirements

In addition to the requirements given in Clause 8, the MeterApplicationProcess shall execute tokens in accordance with the definitions given in Clauses 6 and 7 and shall be further subject to the requirements given in IEC 62055-31 at all times, in particular the action of the load switch in response to remote replenishment of credit and the closing of the load switch from a remote location.

8.2 Token acceptance/rejection

An STS-compliant payment meter shall be capable of reading, interpreting and executing all of the categories of tokens successfully.

The payment meter shall still accept tokens when in the power limiting or tampered state.

Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens are validated by the MeterApplicationProcess once the payment meter has read both tokens and combined them into the new DecoderKey.

A token shall be accepted when all of the following conditions are true:

- AuthenticationResult indicates a status value of Authentic in the APDU (see 7.1.3);
- ValidationResult indicates a status value of Valid in the APDU (see 7.1.4);
- the token can be correctly interpreted and the instruction executed by the MeterApplicationProcess.

If all the above conditions are met, TokenResult (see 7.1.5) shall indicate Accept with the following exceptions:

- successful processing of the first entered token of a key change token pair shall not indicate Accept, but it shall indicate 1stKCT if it is a Set1stSectionDecoderKey or 2ndKCT if it is a Set2ndSectionDecoderKey token; this indicates provisional acceptance until the second token of the key change token pair is also accepted;
- successful processing of the second entered token of a key change token pair shall indicate Accept.

The token shall be rejected and TokenResult shall not indicate Accept if any of the following conditions are true.

- AuthenticationResult does not indicate a status value of Authentic in the APDU (see 7.1.3).
- AuthenticationResult indicates a status value of CRCError in the APDU (see 7.1.3).
- AuthenticationResult indicates a status value of MfrCodeError in the APDU (see 7.1.3).
- ValidationResult does not indicate a status value of Valid in the APDU (see 7.1.4).
- ValidationResult indicates a status value of OldError in the APDU (see 7.1.4).
- ValidationResult indicates a status value of UsedError in the APDU (see 7.1.4).
- ValidationResult indicates a status value of KeyExpiredError in the APDU (see 7.1.4).
- ValidationResult indicates a status value of DDTKError in the APDU (see 7.1.4).
- In the case where completing the transaction execution of a TransferCredit token would cause the credit register in the payment meter to overflow, the TokenResult shall indicate OverflowError in the APDU (see 7.1.5) instead of Accept, the token shall be rejected and shall not be further processed.
- In the case where execution of a key change token would violate the key change rules as given in 6.5.2.4, the TokenResult shall indicate KeyTypeError in the APDU (see 7.1.5) instead of Accept, the token shall be rejected and shall not be further processed. See also 7.3.1 for further key change processing requirements.
- In the case where the structure of the token does not comply with the definitions given in 6.2, 6.3 or in the the application for that token, the TokenResult shall indicate FormatError in the APDU (see 7.1.5) instead of Accept, the token shall be rejected and shall not be further processed.
- In the case where one or more data elements in the token have a value that is outside of the defined range of values defined in 6.2, 6.3 or in the the application for that element, the TokenResult shall indicate RangeError in the APDU (see 7.1.5) instead of Accept, the token shall be rejected and shall not be further processed.
- In the case where the particular function to execute the token is not implemented, the TokenResult shall indicate FunctionError in the APDU (see 7.1.5) instead of Accept, the token shall be rejected and shall not be further processed.

8.3 Display indicators and markings

The payment meter shall uniquely indicate the following conditions:

- the acceptance of a token (see 8.2);
- the rejection of a token (see 8.2);

- when a token is old (see 7.1.4);
- when a token has already been used i.e. duplicate token (see 7.1.4);
- when the DecoderKey has expired (see 7.1.4);
- when a TransferCredit token is presented with a DDTK in the DKR (See 7.1.4);
- when the MeterApplicationProcess cannot execute the token (see 8.2);
- after a successful completion of a key change operation (see 8.2 and 8.9);
- whether accepting the credit on a token would cause the credit register to overflow (see 8.2).

The DRN and the EA code shall be marked on the part of the payment meter that contains the decoder part (see Clause 3) and shall be legible from the outside of the decoder.

In the case where the decoder part is separate from the TokenCarrier interface where the user presents the TokenCarrier to the payment meter, then it shall be possible for the user to determine the DRN and the EA code from the user interface on demand.

Indicators relating to the result of token entry shall only be displayed on the same user interface where the token was entered. In the case of a virtual token carrier for example, it is the task of the application layer protocol and the relevant physical layer protocol to feed back the ValidationResult, AuthenticationResult and TokenResult values.

8.4 TransferCredit tokens

See 6.2.2 for more detail on the structure of this token.

The credit value in the Amount field in the token shall be added to the available credit in the Accounting function in accordance with the specific implementation of the Accounting function and the service type as indicated by the SubClass field in the token.

8.5 InitiateMeterTest/Display tokens

See 6.2.3 for more detail on the structure of this token.

All payment meters shall support test number 0; if any of the incorporated tests are not supported the payment meter must perform the subset of tests that are supported.

The relevant test shall be executed or the relevant information shall be displayed in accordance with the bit pattern in the Control field of the token.

When more than one output is required, for example for test number 0, the outputs shall be initiated in the order in which they are defined in 6.3.8. An optional test may be omitted if it is not implemented. A single test, for example test number 3, may provide more than one field of information.

Any optional tests not supported by the payment meter shall result in the rejection of the optional test token by the payment meter.

In the case where the SubClass value is in the range 11 to 15, the relevant test or display function shall be executed according to the manufacturer's specification, but the payment meter shall verify the MfrCode field value before such a token is accepted.

8.6 SetMaximumPowerLimit tokens

See 6.2.4 for more detail on the structure of this token.

The present value of the maximum power limit register shall be replaced with the new limit.

The action of this function shall be agreed between the utility and the payment meter supplier.

NOTE 1 In a poly-phase payment meter this value is per phase.

NOTE 2 This function is not intended to be used as an overcurrent protection mechanism, which requires adherence to other relevant standards.

8.7 ClearCredit tokens

See 6.2.5 for more detail on the structure of this token.

The available credit in the Accounting function shall be cleared to zero in accordance with the indicated value in the Register field of the token.

8.8 SetTariffRate tokens

See 6.2.6 for more detail on the structure of this token.

The present value in the Tariff Rate Register shall be replaced with the new rate.

8.9 Set1stSectionDecoderKey tokens

See 6.2.7 for more detail on the structure of this token.

The present value of the DecoderKey shall be replaced with the new DecoderKey. The DecoderKey includes its associated attributes like KRN, KT, KEN and TI as defined in 7.3.2.

This action is subject to the successful receipt of both the Set1stSectionDecoderKey and Set2ndSectionDecoderKey tokens.

The payment meter shall have only one active DecoderKey at any stage of its operation. Dual DecoderKeys shall not be used.

It shall be possible to enter the Set1stSectionDecoderKey and Set2ndSection DecoderKey tokens in any order to affect a successful key change.

It shall be possible to enter at least two other invalid tokens of any type and in any order, along with any one of a Set1stSectionDecoderKey and Set2ndSectionDecoderKey token and still perform a successful key change.

It shall be possible to enter the same Set1stSectionDecoderKey and Set2ndSectionDecoderKey token more than once, if the key has not been changed already, and still perform a successful key change.

A time-out function shall be used to cancel a partially completed key change procedure after duration of between 30 s and 10 min.

8.10 Set2ndSectionDecoderKey tokens

See 6.2.8 for more detail on the structure of this token.

The requirements for the processing of the Set2ndSectionDecoderKey tokens are the same as 8.9 above.

8.11 ClearTamperCondition tokens

See 6.2.9 for more detail on the structure of this token.

The control status and indicator that indicates a tamper condition shall be reset to indicate a non-tamper condition. Any internal payment meter control process resultant from such a tamper condition shall also be cancelled.

8.12 SetMaximumPhasePowerUnbalanceLimit tokens

See 6.2.10 for more detail on the structure of this token.

The present value of the maximum phase unbalance power limit register shall be replaced with the new limit.

The action of this function shall be agreed between the utility and the payment meter supplier.

NOTE This function is only applicable to poly-phase payment meters

8.13 SetWaterMeterFactor

See 6.2.11 for more detail on the structure of this token.

The action of this token is reserved for future definition by the STS Association.

8.14 Class 2: Reserved for STS use tokens

See 6.2.12 for more detail on the structure of this token.

The payment meter shall reject these token types.

8.15 Class 2: Reserved for Proprietary use tokens

See 6.2.13 for more detail on the structure of this token.

The actions performed in the payment meter shall be in accordance with the manufacturer's specifications.

NOTE This standard does not provide protection against collision between manufacturer uses of this token space (see also note to 6.2.13).

8.16 Class 3: Reserved for STS use tokens

See 6.2.14 for more detail on the structure of this token.

The payment meter shall reject these token types.

9 KMS: KeyManagementSystem generic requirements

It is recognized that KMS requirements are essentially outside the scope of IEC 62055-41 and the reader is therefore referred to relevant industry standards, some of which are listed in the bibliography.

The STS Association has established well-proven codes of practice for the management of cryptographic keys within STS-compliant systems, utilizing those industry standards, and it is therefore recommended that new systems implementing this standard should follow the STS Association codes of practice.

By virtue of its Registration Authority status with IEC TC 13, the STS Association has undertaken to provide such certification services that are deemed necessary to ensure that key management systems comply with the relevant parts of this standard (see Clause C.1.) For further guidelines on the functioning of a KeyManagementSystem as envisaged in this standard, see Annex A.

10 Maintenance of STS entities and related services

10.1 General

See also Clause C.1 for more information relating to maintenance and support services.

The maintenance activity on certain STS entities requires a revision/amendment of IEC 62055-41. Where this is the case, it is explicitly indicated as such.

Annexes B and C are not normative, and any changes in these clauses due to maintenance activities would not require revision/amendment of IEC 62055-41, but may require appropriate amendments to other relevant specifications or COP.

The STS entities and services that require maintenance are given in Table 42.

Table 42 – Entities/services requiring maintenance service

Entity/service	Definition origin	Responsible maintenance body	Reference
Product certification	Clause C.10	STSA/CA	10.2.1
DSN	6.1.2.3.3 C.3.4	Mfr	10.2.2
RO	6.3.18	utility	10.2.3
TI	6.1.7	utility	10.2.4
TID	6.3.5.1	utility	10.2.5
SpecialReservedTokenIdentifier	6.3.5.2 Clause C.4	utility	10.2.6
MfrCode	6.1.2.3.2 C.3.3	STSA	10.2.7
Substitution tables	6.5.4.2 7.3.3.3 Clause C.5	STSA	10.2.8
Permutation tables	6.5.4.3 7.3.3.2 Clause C.5	STSA	10.2.9
SGC	6.1.6 C.2.2	STSA/KMC	10.2.10
VendingKey	6.5.2.2 Clause 9 C.2.2	STSA/KMC	10.2.11
KRN	6.1.8 6.5.2.5	STSA/KMC	10.2.12

Table 42 *(continued)*

Entity/service	Definition origin	Responsible maintenance body	Reference
KT	6.1.9 6.5.2 Table 30	STSA/KMC	10.2.13
KEN	6.1.10 6.5.2.6 C.2.4	STSA/KMC	10.2.14
KEK	Annex B Table B.1	STSA/KMC	10.2.15
CC	Annex B Table B.2	STSA/KMC	10.2.16
UC	Annex B Table B.2	STSA/KMC	10.2.17
KMCID	Annex B Table B.2	STSA/KMC	10.2.18
CMID	Annex B Table B.2	Mfr/KMC	10.2.19
CMAC	Annex B Table B.2	Mfr/KMC	10.2.20
IIN	6.1.2.2 C.3.2	ISO/IEC	10.3.1
TCT	6.1.3 Table 5	STSA/IEC	10.3.2
DKGA	6.1.4 Table 6	STSA/IEC	10.3.3
EA	6.1.5 Table 7	STSA/IEC	10.3.4
TokenClass	6.3.2 Table 13 Table 14	STSA/IEC	10.3.5
TokenSubClass	6.3.3 Table 14	STSA/IEC	10.3.6
InitiateMeterTest/DisplayControlField	6.3.8 Table 22	STSA/IEC	10.3.7
RegisterToClear	6.3.13 Table 23	STSA/IEC	10.3.8
STS base date	6.3.5.1	STSA/IEC	10.3.9
Rate	6.3.11	STSA/IEC	10.3.10
WMFactor	6.3.12	STSA/IEC	10.3.11
MFO	5.5	STSA/(IEC)	10.3.12

Table 42 *(continued)*

Entity/service	Definition origin	Responsible maintenance body	Reference
FOIN	5.5 Clause C.8	STSA/(IEC)	10.3.13
Companion Specification	5.5 Clause C.8	STSA/(IEC)	10.3.14

10.2 Operations

10.2.1 Product certification maintenance

The STS Association, as a registered Registration Authority with the IEC, shall assure access to product certification services to users of the STS.

It shall also assure that such service providers are duly accredited and authorized to provide this service and that they comply with the requirements of this standard and any other relevant COP or specification.

10.2.2 DSN maintenance

The payment meter manufacturer is in complete control of his allocated range of DSN values (within his allocated MfrCode domain) and it thus requires no further maintenance.

10.2.3 RO maintenance

The utility shall manage the operational use of this data element in conjunction with the STS base date.

10.2.4 TI maintenance

The utility shall manage the operational use of this element.

10.2.5 TID maintenance

The utility shall manage the operational use of this data element by means of appropriate programming of the token vending or POS systems.

10.2.6 SpecialReservedTokenIdentifier maintenance

The utility shall manage the operational use of this data element by means of appropriate programming of the token vending or POS systems.

10.2.7 MfrCode maintenance

The STS Association, as a registered Registration Authority with the IEC, shall provide the service of allocating MfrCode values to payment meter manufacturers and making the list of allocated MfrCode values available to users of the STS upon request.

10.2.8 Substitution tables maintenance

The STS Association, as a registered Registration Authority with the IEC, shall provide the service of making the actual values for Table 33 and Table 41 available to users of the STS upon request.

10.2.9 Permutation tables maintenance

The STS Association, as a registered Registration Authority with the IEC, shall provide the service of making the actual values for Table 34 and Table 40 available to users of the STS upon request.

10.2.10 SGC maintenance

The STS Association, as a registered Registration Authority with the IEC, shall assure access to SGC allocation services to users of the STS and that SGC values are globally unique. Such services are typically provided by a KMC.

10.2.11 VendingKey maintenance

The STS Association, as a registered Registration Authority with the IEC, shall assure access to VendingKey allocation services to users of the STS, that VendingKey values are globally unique and that VendingKey values are made available between KMC service providers. Such services are typically provided by a KMC.

The STS Association shall also assure the compliance of such service providers to the requirements and recommendations given in this International Standard and any other relevant COP or specification.

10.2.12 KRN maintenance

This element is intrinsically coupled to the VendingKey and is managed by the KMC service provider, subject to the same conditions as for VendingKey maintenance.

10.2.13 KT maintenance

This element is intrinsically coupled to the VendingKey and is managed by the KMC service provider, subject to the same conditions as for VendingKey maintenance.

The STS Association in liaison with working group 15 of IEC TC 13 shall administer any further additions to the range of KeyType values as given in Table 30.

The process shall follow the standard procedures for submission of new work item proposals, as instituted by these organisations.

An additional KeyType definition shall require a revision/amendment of IEC 62055-41.

10.2.14 KEN maintenance

This element is intrinsically coupled to the VendingKey and is managed by the KMC service provider, subject to the same conditions as for VendingKey maintenance.

10.2.15 KEK maintenance

The KMC service provider is exclusively in control of this data element as it forms an intrinsic part of its key management operations.

The STS Association, as a registered Registration Authority with the IEC, shall assure that KMC service providers comply with the requirements of this standard and any other relevant COP.

10.2.16 CC maintenance

The STS Association, as a registered Registration Authority with the IEC, shall assure access to CC allocation services to users of the STS and that CC values are globally unique. Such services are typically provided by a KMC.

10.2.17 UC maintenance

The STS Association, as a registered Registration Authority with the IEC, shall assure access to UC allocation services to users of the STS and that UC values are globally unique. A KMC typically provides such services.

10.2.18 KMCID maintenance

The STS Association, as a registered Registration Authority with the IEC, shall assure access to KMCID allocation services to users of the STS and that KMCID values are globally unique. The STS Association typically provides such services.

10.2.19 CMID maintenance

The CM manufacturer is in complete control of allocating CMID values to his manufactured CM devices and there is no service in place to ensure uniqueness of this data element.

Once a particular CM is registered in an STS system (typically with a KMC service provider), then the CMID is simply recorded for reference purposes and no further maintenance service on this data element is required.

10.2.20 CMAC maintenance

The CM manufacturer is in complete control of allocating CMAC values to his manufactured CM devices and there is no service in place to ensure uniqueness of this data element.

The registration transaction of a CMAC value is typically conducted between the CM manufacturer and the KMC service provider, and then it remains in the operations domain of the two parties.

The STS Association, as a registered Registration Authority with the IEC, shall assure the compliance of such manufacturers and service providers to the requirements and recommendations given in this standard and any other relevant COP.

10.3 Standardization

10.3.1 IIN maintenance

This standard defines a constant value for electricity payment meters worldwide.

ISO may issue different values for other services upon application by service providers.

Any changes to the rules as defined in this standard, would require a revision/amendment of IEC 62055-41.

10.3.2 TCT maintenance

The STS Association in liaison partnership with working group 15 of IEC TC 13 shall administer any further additions to the range of TCT values given in Table 5.

The process shall follow the standard procedures for submission of new work item proposals, as instituted by these organisations.