

Telecontrol equipment and systems —

**Part 6-504: Telecontrol protocols
compatible with ISO standards and
ITU-T recommendations —
TASE.1 user conventions**

ICS 33.200

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW



Document Sharing Hub

National foreword

This Published Document reproduces verbatim IEC TR 60870-6-504:1998.

The UK participation in its preparation was entrusted to Technical Committee PEL/57, Telecontrol equipment and systems, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

From 1 January 1997, all IEC publications have the number 60000 added to the old number. For instance, IEC 27-1 has been renumbered as IEC 60027-1. For a period of time during the change over from one numbering system to the other, publications may contain identifiers from both systems.

Cross-references

The British Standards which implement international or European publications referred to in this document may be found in the BSI Standards Catalogue under the section entitled “International Standards Correspondence Index”, or by using the “Find” facility of the BSI Standards Electronic Catalogue.

Summary of pages

This document comprises a front cover, an inside front cover, pages i and ii, the IEC title page, pages ii to iv, pages 1 to 125 and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

This Published Document, having been prepared under the direction of the Electrotechnical Sector Committee, was published under the authority of the Standards Committee and comes into effect on 15 May 1999

© BSI 04-2000

ISBN 0 580 32414 1

Amendments issued since publication

Amd. No.	Date	Comments

Contents

	Page
National foreword	Inside front cover
Foreword	iii
Text of IEC 60870-6-504	1

**RAPPORT
TECHNIQUE – TYPE 2**

**CEI
IEC**

**TECHNICAL
REPORT – TYPE 2**

60870-6-504

Première édition
First edition
1998-12

Matériels et systèmes de téléconduite –

Partie 6-504:

**Protocoles de téléconduite compatibles avec les
normes ISO et les recommandations de l'UIT-T –
Conventions d'utilisation de TASE.1**

Telecontrol equipment and systems –

Part 6-504:

**Telecontrol protocols compatible with ISO
standards and ITU-T recommendations –
TASE.1 user conventions**



Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

Contents

	Page
Foreword	iii
Introduction	1
1 Scope and object	1
2 Normative references	1
3 Definitions and abbreviations	2
3.1 Definitions	2
3.2 Abbreviations	4
4 The TASE.1 User Element	5
4.1 Conceptual model	5
4.2 Functional units (FUs)	6
4.3 Common template for the descriptions of FUs	7
5 The Association Management Function Group	9
5.1 Addressing	9
5.2 Address part	9
5.3 Restart Marking	10
5.4 Group Configuration integrity control	11
5.5 Permanent Association FU	12
5.6 Dynamic Association FU	17
5.7 Test Association FU	23
6 The Data Identification Function Group	26
6.1 Group attributes	26
6.2 Group Management FU	30
6.3 Group Definition FU	37
6.4 Group Readout FU	45
6.5 Group Configuration FU	53
7 The Data Transfer Function Group	56
7.1 Requested Data Transfer FU	56
7.2 Periodically Requested Data Transfer FU	61
7.3 Unsolicited Data Transfer FU	63
7.4 Unsolicited Periodic Data Transfer FU	69
7.5 Unsolicited Spontaneous Data Transfer FU	76
7.6 Supervisory Control Data Transfer FU	82
8 The Restart Function Group	91
8.1 Restart Reconfigure FU	91
8.2 Restart Reactivate FU	95
9 Security	97
9.1 Introduction	97
9.2 Security services	97
9.3 Security mechanisms	98
9.4 Security classes and options	98
9.5 Suggested use of the security mechanisms	99
9.6 Definition of the security information field	100
Annex A (normative) FU invocation hierarchy	101
Annex B (normative) FU disruption hierarchy	101
Annex C (normative) Use of result code values	103
Annex D (informative) Bibliography	104
Annex E (normative) UECS proforma for TASE.1 User Elements	104
Annex F (normative) UEXIT proforma for the TASE.1 User Elements	123
Figure 1 — TASE.1 entity conceptual model	5

Foreword

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the 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, the IEC publishes International Standards. 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. The 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 the 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 National Committees.

3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical reports or guides and they are accepted by the National Committees in that sense.

4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.

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

6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

The main task of IEC technical committees is to prepare International Standards. In exceptional circumstances, a technical committee may propose the publication of a technical report of one of the following types:

- type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;
- type 2, when the subject is still under technical development or where for any other reason there is the future but no immediate possibility of an agreement on an International Standard;
- type 3, when a technical committee has collected data of a different kind from that which is normally published as an International Standard, for example “state of the art”.

Technical reports of types 1 and 2 are subject to review within three years of publication to decide whether they can be transformed into International Standards. Technical reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

IEC 60870-6-504, which is a technical report of type 2, has been prepared by IEC technical committee 57: Power system control and associated communications.

The text of this technical report is based on the following documents:

Committee draft	Report on voting
57/282/CDV	57/331/RVC

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

This document is issued in the type 2 technical report series of publications (according to **G.3.2.2** of part 1 of the IEC/ISO Directives) as a “prospective standard for provisional application” in the field of telecontrol equipment and systems because there is an urgent requirement for guidance on how standards in this field should be used to meet an identified need.

This document is not to be regarded as an “International Standard”. It is proposed for provisional application so that information and experience of its use in practice may be gathered. Comments on the content of this document should be sent to the IEC Central Office.

A review of this type 2 of technical report will be carried out not later than three years after its publication, with the options of either extension for a further three years or conversion to an International Standard or withdrawal.

Annex A, Annex B, Annex C, Annex E and Annex F form an integral part of this technical report.

Annex D is for information only.

Introduction

The TASE.1 Application Programming Interface (TAPI) (see Annex A of IEC 60870-6-501) provides a set of well-defined functions to an application programmer wishing to base an application process on the Telecontrol protocols for real-time data acquisition and setpoint control. However, the TAPI specification does not by itself ensure full functional integration between any two Telecontrol application processes.

The purpose of this technical report is twofold:

- a) To provide a consistent set of rules for the use of the TAPI, or more specifically, rules for:
 - usage of TAPI parameter values;
 - sequencing of Telecontrol Application Service Element service primitives.
- b) To be used as a general reference for specifying the capabilities of Telecontrol application processes. In order to achieve this, a number of “atomic” function packets, or Functional Units (FU), are defined: interoperational characteristics of any Telecontrol application process claiming conformity with this document should be specified by quoting the Functional Units (FU) it supports.

The structure of the “Data” parameter of the data transfer primitives and the user data parameter in the association establishing primitive are described in Annex A of IEC 60870-6-501, together with related information: data types, quality codes etc.

Annex E gives a UECS Proforma for the TASE.1 User Elements, which constitutes a User Element Conformance Statement (UECS).

Annex F gives a UEXIT Proforma, which provides a User Element Extra Information for Testing UEXIT).

1 Scope and object

This technical report defines rules for usage of the TASE.1 Application Programming Interface (TAPI), i.e. parameter usage and service primitive sequencing.

The rules which are set forth by this technical report are to be considered as an extension to the rules enforced by the Telecontrol Application Service Element 1 (TASE.1) itself, and will typically constitute further restrictions on parameter values, etc. The TASE.1 own restrictions and rules apply throughout. They are not explicitly restated in this technical report¹⁾. Whenever the present report allows simultaneous Functional Unit invocations on one association, adherence to the TASE.1 primitive sequencing rules²⁾ are implicitly assumed.

Internal Functional Unit structures, Coordinating Function structure (see clause 4) and structural relationships between these are not within the scope of this technical report, as these are only conceptual entities, serving as a convenient means for describing TASE.1 service primitive sequences.

Also not within scope are all non-communicational aspects of application processes, such as local database design/access, and possible data consistency considerations across different local data access modes.

The main object of this technical report is to present the underlying conceptual model of the Telecontrol User Element and to identify the functional units (clause 4). The functional units are described in minute detail, grouped into one function group per clause (clauses 5 to 8).

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this technical report. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this technical report are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

IEC 60870-6-1:1995, *Telecontrol equipment and systems — Part 6: Telecontrol protocols compatible with ISO standards and ITU-T recommendations — Section 1: Application context and organization of standards*.

¹⁾ Consequently, handling of purely local TAPI errors are outside the scope of this technical report. All specified TASE.1 service primitives of types *Req* and *Resp* are always assumed to be issued within legal context and have legal parameter values, as far as the TAPI is concerned. For the special case of local call status indicating flow control problems (status value = - 1), see 4.2.1.

²⁾ The TASE.1 primitive sequencing rules are embodied in clause 8 of IEC 60870-6-501, and expressed as a set of legal state transitions in clause 7 of IEC 60870-6-502.

IEC 60870-6-501:1995, *Telecontrol equipment and systems — Part 6: Telecontrol protocols compatible with ISO standards, and ITU-T recommendations — Section 501: TASE.1 Service definitions.*

IEC 60870-6-502:1995, *Telecontrol equipment and systems — Part 6: Telecontrol protocols compatible with ISO standards and ITU-T recommendations — Section 502: TASE.1 Protocol definitions.*

ISO 7498-2:1989, *Information processing systems — Open Systems Interconnection Basic Reference Model — Part 2: Security architecture.*

ITU-T Recommendation X.509:1993, *Information technology — Open systems interconnection — The directory: Authentication framework.*

3 Definitions and abbreviations

3.1 Definitions

For the purpose of this technical report, the following definitions apply:

ACCEPTOR address

the unique identification of a User Element responding to an Association attempt, see **5.1** and **5.2.1**

changing a group

modifying one or more of the descriptor attribute values for an existing group identity

composite FU

composite FUs act via invocation of other FUs by an Initiator UE only. They have no associated specific TASE.1 Service primitive sequence. Neither do they have any specific RESPONDER part

configuration Set

the currently agreed-upon group configuration database shared between a number of INITIATOR/RESPONDER systems

configuring a group

creating and defining a group

congestion error

error situation in which the TASE.1 is not able to receive *Req* or *Resp* type service primitives, because of heavy traffic

NOTE General rules for handling congestion errors are given in **4.2.1**. Special rules per FU are given in the individual FU descriptions.

coordinating Function

user Element function that controls the local Functional Unit invocations

creating a group

making a new group identity³⁾ legal, allocating a new group descriptor

defining a group

attaching a set of implicitly numbered symbolic object identifiers to an empty group identity

deleting a group

removing the group identity from the set of legal identities, de-allocating the associated group descriptor

disrupting a Functional Unit or procedure

abruptly (non-orderly) terminating that Functional Unit or procedure

dynamic Association

an association between an INITIATOR UE and a RESPONDER UE, which may be created and terminated at the discretion of the INITIATOR UE

function Group

a named collection of Functional Units of related functionalities

functional Unit invocation

a specific instance of use of the Functional Unit

³⁾ Group number.

functional Unit type

a named collection of Functional Units of related action mechanisms

functional Unit, or User Element Functional Unit

a named well-defined succession of TASE.1 service primitives at the TAPIs of two communicating TASE.1 systems, constituting a single cooperative functional capability of a INITIATOR User Element and its peer RESPONDER User Element⁴⁾

group

a numbered set of named, and implicitly numbered, data objects in a TASE.1 system

incarnation

a consistent set of data values for a group or subgroup, all sampled at a given point in time

INITIATOR address

the unique identification of INITIATOR UE

INITIATOR site

the collection of INITIATOR systems sharing a common Configuration Set. Equivalent to INITIATOR system, if no such sharing

INITIATOR system

the collection of all INITIATOR UEs in a given TASE.1 system, together with the local data processing environment of which the collection is part

INITIATOR User Element, or INITIATOR UE

a User Element controlling associations, groups and data transfer, via the TASE.1

managing a group

creating, changing or deleting a group

permanent Association

an association between an INITIATOR UE and a RESPONDER UE, which is to be maintained at all times

primary FU

primary FUs are the basic kind of FUs; those that are characterized by their individual well defined sequence of TASE.1 Service primitives, and always being invoked by an INITIATOR UE

procedure

sequence of prescribed actions in a TASE.1 INITIATOR UE and/or its peer RESPONDER UE

redefining a group

modifying the existing set of object identifiers in a defined group

RESPONDER system

the collection of all RESPONDER UEs in a given TASE.1 system, together with the local data processing environment of which the collection is part

RESPONDER site

the collection of RESPONDER systems sharing a common Configuration Set. Equivalent to RESPONDER system, if no such sharing

RESPONDER User Element, or RESPONDER UE

the peer communications UE of an INITIATOR UE

secondary FU

secondary FUs have individual well-defined TASE.1 Service primitive sequences, but are always involved by a RESPONDER UE, as a result of local decision in that User Element

subgroup

a contiguous range of objects within a group definition

⁴⁾ This is the basic definition. However, an FU may act exclusively through other FUs, having no specific TASE.1 service primitive sequence associated with it. See 4.2.

TASE.1

telecontrol Application Service Element No. 1: see clause 4

TASE.1 partner

a TASE.1 site with which a given INITIATOR UE or RESPONDER UE may communicate via the TASE.1

TASE.1 provider

the software component that implements the TASE.1 protocol in a given environment

TASE.1 system

the set of User Elements, or single User Element, that utilize the TASE.1 provider that is addressed by the low-level part of a given TASE.1 address⁵⁾. The rest of the local data processing environment of which the User Elements (or User Element), are (is) part, is also considered to belong to the same TASE.1 system

transaction, or TASE.1 transaction

a specific instance of use of an elementary TASE.1 service

3.2 Abbreviations

ADFU:	Dynamic Association FU
AE:	Application Entity
AP:	Application Process
APFU:	Permanent Association FU
ASE:	Application Service Element
ATFU:	Test Association FU
Conf:	confirm
CS:	Configuration Set
CS(I):	The CS copy at the INITIATOR site
CS(R):	The CS copy at the RESPONDER site
DPRFU:	Periodically Requested Data Transfer FU
DRFU:	Requested Data Transfer FU
DSFU:	Supervisory Control Data Transfer FU
DUFU:	Unsolicited Data Transfer FU
DUPFU:	Unsolicited Periodic Data Transfer FU
DUSFU:	Unsolicited Spontaneous Data Transfer FU
FU:	(TASE.1 User Element) Functional Unit
GCFU:	Group Configuration FU
GDFU:	Group Definition FU
GMFU:	Group Management FU
GRFU:	Group Readout FU
Ind:	indication
PDU:	Protocol Data Unit
RAFU:	Restart Reactivate FU
Req:	request
Resp:	response
RRFU:	Restart Reconfigure FU
TAPI:	TASE.1 Application Programming Interface
TASE.1:	Telecontrol Application Service Element No. 1
UE:	User Element
UTC:	Coordinated Universal Time (Greenwich Mean Time).

⁵⁾ A TASE.1 system may be addressed by more than one low-level TASE.1 address. For example, a number of different DTE numbers (which is one form of low-level TASE.1 addresses) will address the same TASE.1 system, if:

- All DTE numbers connect to X.25 lines attached to the device in which the TASE.1 system resides, and;
- All DTE numbers contain the single subaddress that is defined for TASE.1.

4 The TASE.1 User Element

4.1 Conceptual model

For the purposes of this technical report, any TASE.1 Entity [which encompasses the TASE.1 communicational functionality of the Application Process (AP) of which it is part], is modelled as shown in the figure below:

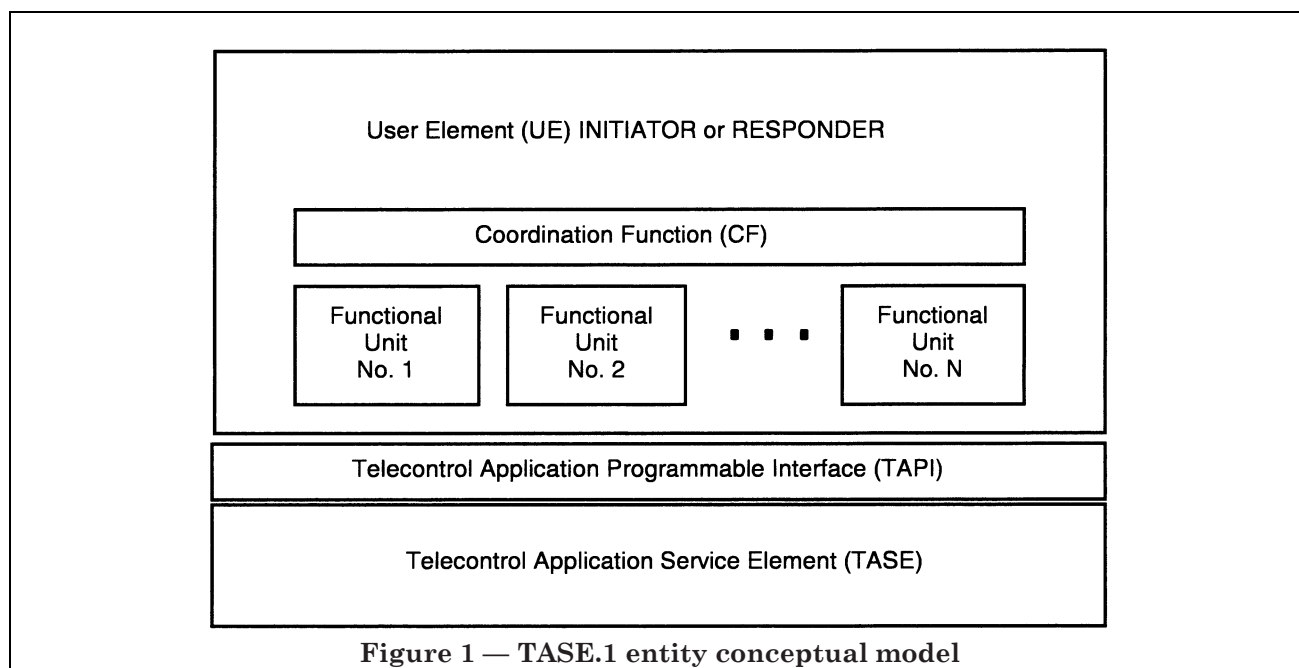


Figure 1 — TASE.1 entity conceptual model

The TASE.1 UE is defined as that part of the TASE.1 AE that is not part of the TASE.1/TAPI. It may be of either the INITIATOR type (in an INITIATOR AP) or of the RESPONDER type (in a RESPONDER AP).

The TASE.1 service primitive sequences defined by this technical report are represented by the indicated Functional Units (FUs), which are coordinated by a single Coordinating Function in each UE⁶⁾.

It should be noted that this is a purely conceptual model. It is not to be interpreted as an implementation specification with respect to the existence of any actual FU or Coordinating Function entity in a UE. Neither is the indicated hierarchical ordering of FUs and the Coordinating Function mandatory.

This technical report specifies a number of FU invocations and disruptions by other FUs. This mechanism is a purely notational convenience for describing the temporal ordering and possible abrupt termination of the various TASE.1 service primitive sequences, and shall not be interpreted as an architectural requirement⁷⁾.

The Coordinating Function represents all interactions between FUs and higher-order functions, as well as between the FU invocations themselves. In the detailed descriptions of the individual FUs below, the required Coordinating Function functionalities are stated in the subclauses “Relation to other FUs” and “Coordination rules”.

To summarize:

FUs function as *objects for TASE.1 primitive sequence specifications*, and the Coordinating Function functions as an *object for describing the coordination rules between FUs*.

The Coordinating Function itself is not described in any more detail by this technical report, allowing for any manner of implementation as long as the stated rules are adhered to.

The individual FUs are described in considerable detail below.

⁶⁾ FUs of type composite (that have no explicit primitive sequence associated with them) may be looked upon as being associated with the composite primitive sequence consisting of the primitive sequences of the “component” FUs that the FU invokes.

⁷⁾ For example, all FU invocations may be directly controlled by the Coordinating Function, and an FU invocation by another FU may be effected by request from the “invoking” FU to the Coordinating Function.

4.2 Functional units (FUs)

4.2.1 General

FUs generally consist of an INITIATOR part and a RESPONDER part. Any INITIATOR type system supporting a given FU shall implement the INITIATOR part of that Functional Unit. Correspondingly, any RESPONDER type system shall implement the RESPONDER part of any FU it supports.

The INITIATOR and the RESPONDER parts of any FU are invoked and terminated independently, upon events that generally differ in the INITIATOR and RESPONDER. However, any FU invocation in which the one part does not eventually succeed in invoking the other, will be terminated in due course, because of local error handling action.

Concurrent and multiple Functional Unit invocations are generally legal. Restrictions that this technical report imposes on a given Functional Unit in this respect are listed in the description of that Functional Unit, below⁸⁾.

Local TAPI errors of the type “*call not accepted due to flow control*” shall be treated in the following manner: either:

- perform the action specified under the heading “congestion error”, in the FU description below;

or:

- suspend the FU for a locally determined amount of time, then retry. Perform the suspend/retry cycle until success, other error or a locally determined maximum number of retries has been performed;
- if success: proceed as usual;
- if the “*call not accepted due to flow control*” has occurred persistently: perform the action specified under the heading “congestion error”, in the FU description below;
- if other error: handle as specified in the FU description, below.

4.2.2 Overview

The FU concept is presented in the preceding subclauses. In this subclause each FU defined by this technical report is listed, together with a short description of its scope. For ease of presentation, the FUs are grouped into Function Groups, each of which encompasses the FUs of a specific “family” of functions. Each FU is assigned to one of three defined Functional Unit types, reflecting different action modes: primary, secondary and composite FUs.

Primary FUs are the basic kind of FUs; those that are characterized by their individual well-defined sequence of TASE.1 service primitives, and always being invoked by an INITIATOR UE.

Example: the *Unsolicited Data Transfer FU*.

Secondary FUs have individual well-defined TASE.1 service primitive sequences, but are always invoked by a RESPONDER UE, as a result of local decision in that UE.

Example: the *Unsolicited Periodic Data Transfer FU*.

Composite FUs act via invocation of other FUs by an INITIATOR UE only. They have no associated specific TASE.1 service primitive sequence. Neither do they have any specific RESPONDER part.

Example: the *Restart Reconfigure FU*.

An FU is identified by its individual acronym, the first letter of which identifies its Function Group:

A: Association Management

G: Data Identification

D: Data Transfer

R: Restart

THE ASSOCIATION MANAGEMENT FUNCTION GROUP: A

The *Permanent Association FU*. Acronym: APFU. Type: Primary. Creation of a single association between an INITIATOR UE and a RESPONDER UE, and re-creating it whenever disrupted. Associations handled by this FU are referred to as permanent.

⁸⁾ Note, however, that the TASE.1 imposes its own set of primitive sequencing rules, in addition to those mentioned here. See clause 1.

The *Dynamic Association FU*. Acronym: ADFU. Type: Primary. Creation of a single association between an INITIATOR UE and a RESPONDER UE, registering its eventual disruption or eventually terminating it in an orderly manner. Associations created by this FU are referred to as dynamic.

The *Test Association FU*. Acronym: ATFU. Type: Primary. Checking the reachability of a RESPONDER UE from an INITIATOR UE.

THE DATA IDENTIFICATION FUNCTION GROUP: G

The *Group Configuration FU*. Acronym: GCFU. Type: Composite. Creating and defining a group in a CS(R) on request by an INITIATOR UE.

The *Group Management FU*. Acronym: GMFU. Type: Primary. Creating, changing or deleting a group in a CS(R) on request by an INITIATOR UE.

The *Group Definition FU*. Acronym: GDFU. Type: Primary. Defining a group in a CS(R) on request by an INITIATOR UE.

The *Group Readout FU*. Acronym: GRFU. Type: Primary. Transferring a readout of a group configuration from a CS(R) to an INITIATOR UE, on request by the latter.

THE DATA TRANSFER FUNCTION GROUP: D

The *Requested Data Transfer FU*. Acronym: DRFU. Type: Primary. Transferring data associated with a single group once, from a RESPONDER UE to an INITIATOR UE, on request by the latter.

The *Periodically Requested Data Transfer FU*. Acronym: DPRFU. Type: Composite. Transferring data associated with a single group, from a RESPONDER UE to an INITIATOR UE, on periodical requests by the latter.

The *Unsolicited Data Transfer FU*. Acronym: DUFU. Type: Primary. Granting permission to the RESPONDER UE to transfer data from one specified group, either periodically or spontaneously.

The *Unsolicited Periodic Data Transfer FU*. Acronym: DUPFU. Type: Secondary. Transferring data associated with any group(s) periodically, from a RESPONDER UE to an INITIATOR UE, which has granted permission for periodic data transfer concerning the group in question. Permission granting and withdrawal is part of the *Unsolicited Data Transfer FU*.

The *Unsolicited Spontaneous Data Transfer FU*. Acronym: DUSFU. Type: Secondary. Transferring data associated with any group(s), from a RESPONDER UE to an INITIATOR UE, any number of times at the RESPONDER UE's decision, after permission has been granted by the INITIATOR UE. Permission granting and withdrawal is part of the *Unsolicited Data Transfer FU*.

The *Supervisory Control Data Transfer FU*. Acronym: DSFU. Type: Primary. Transferring functional commands or setpoints from an INITIATOR UE to a RESPONDER UE.

THE RESTART FUNCTION GROUP: R

The *Restart Reconfigure FU*. Acronym: RRFU. Type: Composite. Furnishing a RESPONDER UE with a complete CS(R), by an INITIATOR UE, as seen appropriate by the latter after being requested by the former. The request mechanism is part of the *Permanent Association FU* and the *Dynamic Association FU*, and not of the *Restart Reconfigure FU* itself.

The *Restart Reactivate FU*. Acronym: RAFU. Type: Composite. Granting, to a RESPONDER UE, permission to transmit data, both periodic and spontaneous, by an INITIATOR UE for all groups that the latter judges appropriate, after being requested by the former. The request mechanism is part of the *Permanent Association FU* and the *Dynamic Association FU*, and not of the *Restart Reactivate FU* itself.

4.3 Common template for the descriptions of FUs

The individual Functional Unit descriptions conform to a common template:

Function:

A short description of the normal function

Coordination rules:

A list of rules to be observed by the Coordinating Function governing Functional

Unit invocations:

Association usage:

Requirements as to the association(s) that are used by the Functional Unit.

Relation to other FUs:

Invoking FUs:

List of other Functional Units that may invoke the Functional Unit.

Invoked FUs:

List of other Functional Units that the Functional Unit may invoke.

Disrupting FUs:

List of Functional Units that may disrupt (abruptly terminate) the Functional Unit

Disrupted FUs:

List of Functional Units that may be disrupted (abruptly terminated) by the Functional Unit.

Invocation:

Prerequisites:

List of conditions to be met before invoking the Functional Unit.

Restrictions:

List of illegal invocation combinations involving the Functional Unit.

Invoking events:

Description of the Functional Unit invocation mechanisms.

Termination:

Orderly termination:

Description of the mechanisms for orderly termination of the Functional Unit.

Disruption:

Description of the mechanisms for abrupt termination of the Functional Unit.

Procedures:

Prescribed actions within the Functional Unit.

TASE.1 service primitives:

List of elementary TASE.1 services used by the Functional Unit, with description of error-free operation:

Sequence:

Normal sequence of primitives associated with the Functional Unit.

Parameter values:

Values of parameters of primitives *issued* to the TASE.1 in the course of normal primitive sequence.

Error handling:

Set of rules for handling error conditions within the Functional Unit:

FU disruption:

Handling of abrupt termination of the Functional Unit.

Illegal invocation attempt:

Handling of Functional Unit invocation attempts that are not legal, according to the rules stated in section Restrictions (above).

Incoming TASE.1 service primitive out of context:

Handling of primitives *received* from the TASE.1, with legal parameter values, but occurring at the wrong time.

Timing errors:

Handling of missing or excessively delayed response on the part of the peer User

Element of an INITIATOR or RESPONDER User Element.

Congestion error:

Handling of situations where the TAPI is not able to accept another outgoing primitive because of heavy load.

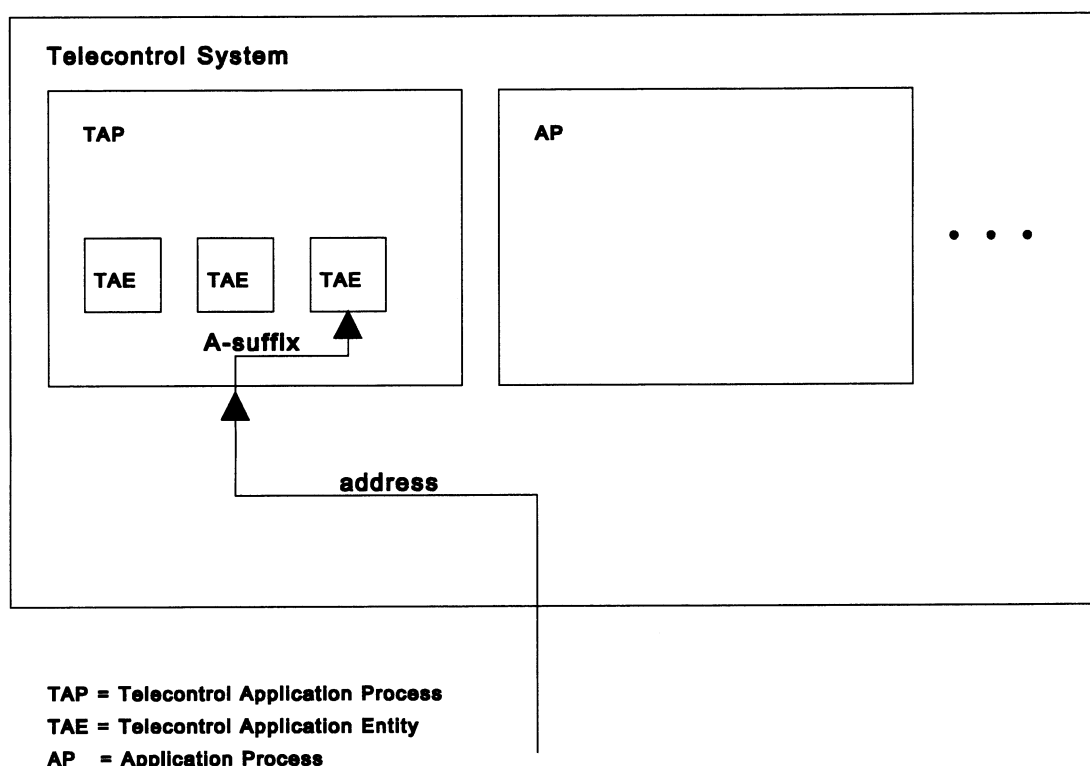
TASE.1 service primitive parameter errors:

Handling of erroneous parameter values in *incoming* primitives from the TASE.1, the primitives themselves occurring within normal sequence.

5 The Association Management Function Group

5.1 Addressing

The function of a TASE.1 address is to specify an Application Entity in a target TASE.1 system. TASE.1 addresses are composed of two parts: The application process title, hereafter called “address”, and the application entity qualifier, hereafter called “A-suffix”. The address specifies a telecontrol application process uniquely within the space of all telecontrol application processes, while the A-suffix specifies an Application Entity uniquely within the space of all Application Entities within the addressed telecontrol application process.



It is a local matter to find the presentation address for a given address. The OSI Directory may be used for that purpose.

The *Permanent Association FU* and the *Dynamic Association FU* utilize TASE.1 addresses and A-suffixes coded into parameters Initiator and Acceptor for association establishment; see below.

Any UE Conforming to this report shall be able to support invocations of both the *Permanent Association FU* and the *Dynamic Association FU* in order to establish associations with an INITIATOR or RESPONDER site via more than one TASE.1 address per site. Each TASE.1 address shall correspond to a component INITIATOR or RESPONDER system within the site.

TASE.1 address selection is a function at Coordinating Function level.

5.2 Address part

The address (application process title) has to be transformed into a presentation address before calling the *TASEBind*. The transformation is a local matter, however the OSI Directory may be used⁹⁾.

The presentation address consists of four components: presentation selector, session selector, transport selector, and network addresses.

⁹⁾ The address will then be built up as a so-called “Distinguished Name” which is the entry to the OSI Directory. The OSI Directory responds with the presentation address, and the application has to add the A-suffix (or Application Entity Qualifier) before calling the *TASEBind*.

5.2.1 A-suffixes

A-suffixes are components of the Initiator and Acceptor parameters of the *TASEBind* primitives. Only the following values are to be considered as standard within the context of this report:

Function	Functional Units	Initiator	Responder
Accessing the CS(R) from the INITIATOR	GMFU, GDFU, GRFU	"AA"	"BA"
Unsolicited Spontaneous Data Transfer	DUFU, DUSFU, DRFU ^b	"AB"	"BB"
Unsolicited Periodic Data Transfer	DUFU, DUPFU, DRFU ^c	"AC"	"BC"
Requested Data Transfer, scheduling information ^a	DRFU	"AD"	"BD"
Requested Data Transfer, pre-sent or archived information	DRFU	"AE"	"BE"
Test Association, when association is established solely for this purpose	ATFU	"AF"	"BF"
Supervisory Control Data Transfer	DSFU	"AG"	"BG"
NOTE "AB" means the ASCII representation of the character pair AB, i.e. the decimal value pair 65, 66.			
^a An implementation of the DRFU may be restricted to only support present information (no archive information) on an association where an Unsolicited Spontaneous Data Transfer FU is already running.			
^b DRFU is legal only if no Periodic data Transfer FU invocation is currently active on the association in question.			
^c Schedules (future and historic) power plant operation, energy exchange, discharges etc.			

A-suffixes not defined by the table above are legal, but only on the basis of bilateral agreement (local conventions). Such "private" A-suffixes shall never begin with characters "A" or "B", which are reserved for A-suffixes standardized by this report.

Example: If an INITIATOR UE wants to initiate unsolicited spontaneous data transfer from a RESPONDER UE, it should use the A-suffix "AB" in the Initiator parameter and the A-suffix "BB" in the Acceptor parameter of the *TASEBindReq* primitive, it will issue in order to establish an association over which it later will grant permission to send the unsolicited data.

5.2.2 Coding of addresses

Any Initiator or Acceptor address parameter shall be a string of octets with the following interpretation, with octet No. 0 denoting the first octet in the string, at lowest address:

- Octet No. 0: Binary number, expressing the number of octets in the address part (= N)
- Octets No. 1 through N: address part
- Octet No. N+1: Binary number, expressing the number of octets in the A-suffix part (= M)^a
- Octets No. N+2 through N+M+1: A-suffix part

^a M is always = 2 for the A-suffixes standardized by this report.

(Total number of octets: N+M+2.)

This report does not specify the coding of presentation addresses. Presentation addresses shall be coded according to the specifications of the underlying communications products.

5.3 Restart Marking

A RESPONDER UE may use a restart marking in the User-data parameter of the *TASEBindResp* primitive in order to signal its need for a re-initialization by the INITIATOR, with respect to the permission to send unsolicited and/or periodic data. The INITIATOR will eventually respond to this request by invoking the *Restart Reactivate FU* for the RESPONDER system in question. Only the following restart codes are legal within the context of this report¹⁰⁾:

Value	Interpretation	Functional Unit
"0" (= decimal 48)	<i>No restart</i> : No re-initialization required	—
"1" (= decimal 49)	(Reserved for future use)	—
"2" (= decimal 50)	<i>Restart, spontaneous management lost</i> : Repetition of all permissions to send required	RAFU

¹⁰⁾ See ASN.1 description IEC 60870-6-502 for coding of User-Data.

The RESPONDER UE shall always use restart code = “2” in order to signal spontaneous management lost. This will introduce some overhead, but is an unambiguous way to communicate re-establishing.

The restart codes are significant for the INITIATOR, only when Result = *ok* in the *TASEBindResp* primitive.

5.4 Group Configuration integrity control

TASE.1 offers a mechanism for detecting mismatch between the CS(I) (INITIATOR’s copy of the current Configuration Set) and the CS(R) (RESPONDER’s copy of the current Configuration Set). This mechanism is based on a Control Field (CF) value generated at the RESPONDER site and stored with the CS(R) at the RESPONDER site and with the CS(I) at the INITIATOR site.

The integrity control mechanism is divided into a procedure for Configuration Set update, and a procedure for Configuration Set check.

Procedure for Configuration Set update:

Whenever the *Group Management FU* or the *Group Definition FU* is invoked, the RESPONDER part shall generate a Control Field value upon reception of the corresponding valid *GroupMgmtInd* or *DefineGroupInd* primitive.

The Control Field value shall be an array of 9 elements. The elements shall be 16 bit integers¹¹⁾. The individual elements shall contain the following information, element No. 0 denoting the first element, at lowest address:

- Elements No. 0 through 6: Time stamp for current CS(R) update: See Annex of IEC 60870-6-501, description of *GroupMgmtResp*.
- Elements No. 7 through 8: 32-bit checksum, generated from the contents of the whole CS(R) after update:

The checksum algorithm is not specified by this report. The actual use of the checksum facility is not mandatory: it may be disabled entirely, the RESPONDER always setting all octets of the checksum part of the Control Field equal to a fixed value, for example 0.

The Time stamp of the Control Field value shall be stored with the updated CS(R) at the RESPONDER site, overwriting the previous Time element part in the local group Configuration database¹²⁾.

After the Time stamp of the new Control Field value has been successfully stored, the *GroupMgmtResp* or *DefineGroupResp* primitive terminating the current FU invocation shall be issued, containing the new Control Field value in the CF parameter (including the checksum).

Upon reception of a valid *GroupMgmtConf* or *DefineGroupConf* primitive, the INITIATOR part of the FU shall register the value in the CF parameter of the primitive as its new Control Field value. Before terminating the FU invocation locally, the INITIATOR shall store this new Control Field value (including the checksum) with the updated CS(I), overwriting the previous Control Field value in the local group Configuration database.

Procedure for Configuration Set check:

Whenever the *Permanent Association FU* or the *Dynamic Association FU* is invoked, the User-data parameter of the *TASEBindResp* primitive issued by the RESPONDER part of the FU shall contain a Control Field value. This value shall be coded according to the rules described in Annex A of IEC 60870-6-501.

The Control Field value shall have been generated in the following manner:

- The time stamp shall be equal to the time stamp part currently stored with the CS(R).
- The checksum shall be computed¹³⁾ from the current CS(R), using the same algorithm as during last Configuration Set update.

¹¹⁾ Implementations having “native” integers of more than 16 bits shall only use the 16 low-order bits of each integer element for storing significant Control Field data.

¹²⁾ Alternatively, the whole Control Field (time stamp plus checksum) may be stored with the CS(R). In this case the Control Field returned with subsequent *TASEBindResp* primitives (see procedure for Configuration Set check below) may contain the stored checksum instead of the result of another checksum computation.

¹³⁾ Alternatively, the checksum may be fetched from the local database if it resides there; see footnote in procedure for Configuration Set update.

The INITIATOR part of the *Permanent Association FU* or the *Dynamic Association FU* shall, upon reception of a valid *TASEBindConf* primitive, decode the Control Field value from the User-data parameter of the primitive, using the same octet numbering rules as those specified for the RESPONDER part. The decoded Control Field value shall then be compared to the Control Field value (including checksum) that currently resides with the CS(I).

In case of equality, no special action shall be taken. In case of non-equality, the INITIATOR shall invoke the *Restart Reconfigure FU* for the partner in question.

Note that the INITIATOR part of the various FUs involved never takes any part in Control Field value generation: Its only function in this respect is to register and compare Control Field values. Consequently, the rules for generating Control Field values (including checksum algorithms) are left to the individual RESPONDER implementations.

5.5 Permanent Association FU

Type: Primary.

5.5.1 Function

A *Permanent Association FU* invocation provides the following functional sequence:

- a) The INITIATOR UE establishes an association with a RESPONDER UE.
- b) If the RESPONDER UE has requested a re-initialization function from the INITIATOR UE during association establishment, the corresponding FU is invoked by the INITIATOR UE.
- c) At any time later, whenever the *Permanent Association FU* registers the association as being broken (abruptly terminated), it will attempt to re-establish the association. Cyclic re-establishment attempts will proceed indefinitely, until success or intervention by a higher-order function via the Coordinating Function in the INITIATOR UE.
- d) If the INITIATOR UE, after an association attempt receives a *TASEBindConf* primitive with Result = *rejected permanent*, he should not proceed with further association attempts.

5.5.2 Coordination rules

5.5.2.1 Association usage

The FU invocation itself creates and maintains an association. No other association is used by that FU invocation.

5.5.2.2 Relation to other FUs

5.5.2.2.1 Invoking FUs

The *Permanent Association FU* shall not be invoked by any other FU.

5.5.2.2.2 Invoked FUs

The *Permanent Association FU* may invoke the following FUs:

- *Restart Reconfigure FU*;
- *Restart Reactivate FU*.

5.5.2.2.3 Disrupting FUs

The *Permanent Association FU* may not be disrupted by any other FU.

5.5.2.2.4 Disrupted FUs

The *Permanent Association FU* may disrupt the following FUs, as a result of (temporary) association breakage:

- *Test Association FU*
- *Group Configuration FU*
- *Group Management FU*
- *Group Definition FU*
- *Group Readout FU*
- *Requested Data Transfer FU*
- *Periodically Requested Data Transfer FU*
- *Unsolicited Data Transfer FU*

- *Unsolicited Periodic Data Transfer FU*
- *Unsolicited Spontaneous Data Transfer FU*
- *Supervisory Control Data Transfer FU*.

5.5.2.3 Invocation

5.5.2.3.1 Prerequisites

No previous or concurrent FU invocations are required.

5.5.2.3.2 Restrictions

In a given INITIATOR system, the *Permanent Association FU* shall not be invoked for a given RESPONDER system if:

- a) *Permanent Association FU* invocation is running with the same RESPONDER system, with identical values of the parameters in the *TASEBindReq* primitive
- or:
- b) *Group Management FU* invocation or a *Group Definition FU* invocation is currently being running for the same RESPONDER User Element¹⁴⁾.

5.5.2.3.3 Invoking events

The INITIATOR part of the *Permanent Association FU* may only be invoked by local request via the Coordinating Function, at system startup time.

Such local invocation requests shall always be legal, in the sense that if the *Permanent Association FU* is present, the *Permanent Association FU* invocation in question shall enter its association re-establishment loop if the association cannot be established immediately.

Invocation of the RESPONDER part of the *Permanent Association FU* is triggered by reception of a valid *TASEBindInd* primitive.

Note that the RESPONDER part of the *Permanent Association FU* necessarily must be identical to the RESPONDER part of the *Dynamic Association FU*¹⁵⁾, because no explicit information as to permanency is conveyed by the *TASEBindInd* primitive. Therefore, the RESPONDER part of the *Permanent Association FU* will be referred to as simply the RESPONDER part of the *Association FU* throughout the remaining part of this description of the *Permanent Association FU*.

5.5.2.4 Termination

5.5.2.4.1 Orderly termination

Orderly termination of the *Permanent Association FU* is not defined.

5.5.2.4.2 Disruption

A *Permanent Association FU* invocation may be disrupted in the establishment phase by receiving a *TASEBindConf* primitive with Result = *Rejected Permanent*¹⁶⁾.

5.5.3 Procedures

5.5.3.1 TASE.1 service primitives

The following elementary TASE.1 services are used by the *Permanent Association FU*:

- *TASEBind*
- *TASEAbort*

¹⁴⁾ The purpose of this rule is to ensure the integrity of the Configuration integrity parameter for the INITIATOR system in the RESPONDER system. This parameter is updated by both the *Group Management FU* and the *Group Definition FU* and utilized by the *Permanent Association FU*; see 5.4.

¹⁵⁾ Consequently, since required by the INITIATOR part of the *Dynamic Association FU*, the RESPONDER part of the *Association FU* must contain a mechanism for Orderly Termination, triggered from the INITIATOR part. However, Orderly Termination will never be triggered by the INITIATOR part of a *Permanent Association FU* invocation, since no such mechanism is defined for this FU.

¹⁶⁾ Any other attempt to disrupt the *Association FU* on the part of the RESPONDER by abruptly terminating the association will result in the INITIATOR part eventually re-establishing the association.

5.5.3.1.1 Sequence

The normal sequence of primitives is partitioned into two phases:

Phase 1: establishing the association

INITIATOR UE	TASE.1	RESPONDER UE
<i>TASEBindReq</i> <i>TASEBindInd</i>	<----- ----->	<i>TASEBindConf</i> <i>TASEBindResp</i>

Phase 2: abruptly terminating and re-establishing the association

INITIATOR UE	TASE.1	RESPONDER UE
<i>TASEAbortInd</i> <i>TASEBindReq</i> <i>TASEBindReq</i>	<-- --> <----- ----->	<i>TASEAbortInd</i> <i>TASEBindConf</i> <i>TASEBindReq</i>

Rules:

- a) Phase 2 may occur at any time after phase 1 has been terminated, and may occur any number of times, including none at all¹⁷⁾.
- b) Whenever phase 2 occurs, the *TASEAbortInd* primitive shall trigger the following actions:
 - In the INITIATOR part:
 - Local disruption of all other FUs that are running on the association.
 - In the RESPONDER part:
 - Local disruption of all other FUs that are running on the association.
 - The next time any association with the characteristics for Unsolicited Data Transfer (for use by the *Unsolicited Periodic Data Transfer FU* or *Unsolicited Spontaneous Data Transfer FU*) is established between the same INITIATOR and RESPONDER UE pair, the RESPONDER part of the *Association FU* shall signal restart code *Restart, spontaneous management lost*.
- c) The time delay between reception of the *TASEAbortInd* primitive and the issuing of the *TASEBindReq* primitive in the INITIATOR UE is a local matter.
- d) Whenever the restart code *Restart, spontaneous management lost* is received by the INITIATOR UE with the *TASEBindConf* primitive in phase 1, the *Permanent Association FU* shall try to invoke the *Restart Reactivate FU* for the RESPONDER in question.
- e) If Configuration control field error (see 5.4) is detected by the INITIATOR UE with the *TASEBindConf* primitive in phase 1, the *Permanent Association FU* shall try to invoke the *Restart Reconfigure FU* for the RESPONDER in question.
- f) The rules stated in the procedure for Configuration Set check (see 5.4) also apply.

¹⁷⁾ In contrast to the *Dynamic Association FU*, any occurrence of the *TASEAbortInd* primitive is handled within the FU invocation, and thus does not constitute a *Permanent Association FU* disruption event or error.

5.5.3.1.2 Parameter values

TASEBind:

Parameter	Req (INITIATOR)	Resp (RESPONDER)
Initiator	As specified in description of <i>TASEBindReq</i> call in Annex A of IEC 60870-6-501. The “address part” shall be equal to the address of the INITIATOR’s own node as specified in 5.2. The “A-suffix” part shall be as specified in 5.2.1.	Copy of value from <i>Ind</i>
Acceptor	As specified in description of <i>TASEBindReq</i> call in Annex A of IEC 60870-6-501. The “address part” shall be equal to the address of the RESPONDER’s own node as specified in 5.2. The “A-suffix” part shall be as specified in 5.2.1.	Copy of value from <i>Ind</i>
User-data	Octets, user defined, optional.	First octet (octet No. 0): Optional Octets No. 1 through 16: Control Field value. See the procedure for Configuration Set check, in 5.4, for coding rules.
	The security information field starts from octet 2 if present. The field is encoded as described in the 9.6.	The security information field starts from octet 17 if present. The field is encoded as described in 9.6. The security information field must be present if security classes 2 or 3 are used. It must be present if security class 1 with two-way authentication is selected, as well.
Result	(Not applicable)	Association established: = <i>result-ok</i> Association not established, due to error condition: Other value, see 5.5.3.2.

5.5.3.2 Error handling

5.5.3.2.1 FU disruption

The *Permanent Association FU* may be disrupted by receiving a *TASEBindConf primitive* with Result = *Rejected Permanent* (see above).

5.5.3.2.2 Illegal invocation attempt

FU not present:

If the *Permanent Association FU* is not present in an INITIATOR UE:

Invocation requests are always generated locally at system startup time; see section “Invocation”, above. Consequently, the handling of this type of error is a local issue, outside the scope of this report.

In a RESPONDER UE, the *Association FU* must be present, to make the communication system able to function at all.

FU present, but attempt illegal:

Attempts of multiple simultaneous invocations of the *Permanent Association FU* for identical parameter sets is a local issue, outside the scope of this report.

Attempted *Permanent Association FU* invocation violating rule 6 in 5.5.2.3.2 (*Group Management FU* or *Group Definition FU* running): Start local retry timer, and enter normal retry loop upon expiry.

5.5.3.2.3 Incoming TASE.1 service primitive out of context

INITIATOR part:

State	TASEBindConf
FU not yet running	Ignore, or local error indication
FU running, waiting for TASEBindConf	(Normal)
Running	Ignore, or local error indication/logging
Waiting for retry timer expiry	Ignore, or local error indication/logging
NOTE The TASEAbortInd primitive is never considered out of context.	

RESPONDER part:

State	TASEBindInd
Idle	(Normal)
Running	Ignore, or local error indication/logging
NOTE The TASEAbortInd primitive is never considered out of context	

5.5.3.2.4 Timing errors

There can occur no timing errors in the INITIATOR part.

Error in RESPONDER part:

Error	Reaction from TASE.1	Specified action in FU
UE too late responding to TASEBindInd	In RESPONDER: TASEAbortInd, with Reason = <i>misbehaviour-of-local-service-user</i> . Local error from eventual attempt to issue TASEBindResp In INITIATOR: TASEBindConf, with Result = <i>no-answer-from-remote-system</i> .	RESPONDER part: Go to idle state. INITIATOR part: See 5.5.3.2.6.

5.5.3.2.5 Congestion error

Congestion errors do not occur in the *Permanent Association FU*. See 4.2.1.

5.5.3.2.6 TASE.1 service primitive parameter errors

Errors in the TASEBindInd primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Address part of the Initiator (identifying the INITIATOR node) not valid, according to local access tables	Issue TASEBindResp primitive with Result = <i>authentication-failure</i> . Enter idle state, if in other state.
INITIATOR requested security class > 0, but the RESPONDER does not support any security mechanism	Issue TASEBindResp primitive with Result = <i>security-is-not-supported-by-service-user</i> Enter idle state, if in other state.
INITIATOR requested security class > 0, but the length of the authentication information field is zero	Issue TASEBindResp primitive with Result = <i>authentication-failure</i> . Enter idle state, if in other state.

Error	Action in RESPONDER part of FU
INITIATOR requested a security class or security options not supported by the RESPONDER	Issue <i>TASEBindResp</i> primitive with Result = <i>incompatible-security-options-requested</i> . Enter idle state, if in other state.
INITIATOR requested field authentication or encipherment for a parameter not present in the selected TASE.1 version	Issue <i>TASEBindResp</i> primitive with Result = <i>incompatible-security-options-requested</i> . Enter idle state, if in other state.
INITIATOR requested a security option (field authentication or encipherment) not supported by the RESPONDER	Issue <i>TASEBindResp</i> primitive with Result = <i>incompatible-security-options-requested</i> . Enter idle state, if in other state.
RESPONDER could not authenticate the INITIATOR on the basis of the received authentication information field	Issue <i>TASEBindResp</i> primitive with Result = <i>authentication-failure</i> . Enter idle state, if in other state.

Errors in the *TASEBindConf* primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Zero octets in User-data, or restart marking in User-data not defined in table in, 5.3, and Result = <i>result-ok</i> .	Treat as normal User-data.
Result >< <i>result-ok</i>	NOT an error; enter running state as normal. Start local retry timer, and enter normal retry loop upon expiry ^a .
^a If security classes 1-3 are selected, the retry strategy should be carefully considered. It may not be advisable to enter normal retry. This may give the receiver new opportunities to receive authentication fields which may be used for subsequent replay/masquerade or for attempts to break the encipherment keys.	

5.6 Dynamic Association FU

Type: Primary.

5.6.1 Function

A *Dynamic Association FU* invocation provides the following functional sequence:

- The INITIATOR UE establishes an association with a RESPONDER UE.
- If the RESPONDER UE has requested a re-initialization function from the INITIATOR UE during association establishment, the corresponding FU is invoked in the INITIATOR UE.
- At any time later, whenever the *Dynamic Association FU* registers the association as being broken (abruptly terminated), the *Dynamic Association FU* invocation will be disrupted.
- During orderly termination of the *Dynamic Association FU* invocation, the INITIATOR UE releases the association.

5.6.2 Coordination rules

5.6.2.1 Association usage

The FU invocation itself creates and releases an association. No other association is used by that FU invocation.

5.6.2.2 Relation to other FUs

5.6.2.2.1 Invoking FUs

The *Dynamic Association FU* may be invoked by the following FU:

— *Restart Reconfigure FU*.

5.6.2.2.2 Invoked FUs

The *Dynamic Association FU* may invoke the following FUs:

— *Restart Reconfigure FU*.

5.6.2.2.3 Disrupting FUs

The *Dynamic Association FU* may not be disrupted directly by any other FU.

Indirectly the *Dynamic Association FU* may be disrupted by any of the following FUs triggering abrupt association termination:

- *Test Association FU*
- *Group Configuration FU*
- *Group Management FU*
- *Group Definition FU*
- *Group Readout FU*
- *Requested Data Transfer FU*
- *Periodically Requested Data Transfer FU*
- *Supervisory Control Data Transfer FU*

5.6.2.2.4 Disrupted FUs

The *Dynamic Association FU* may disrupt the following FUs, as a result of disruption of the *Dynamic Association FU* itself:

- *Test Association FU*
- *Group Configuration FU*
- *Group Management FU*
- *Group Definition FU*
- *Group Readout FU*
- *Requested Data Transfer FU*
- *Periodically Requested Data Transfer FU*
- *Supervisory Control Data Transfer FU*

5.6.2.3 Invocation

5.6.2.3.1 Prerequisites

No previous or concurrent FU invocations are required.

5.6.2.3.2 Restrictions

In a given INITIATOR system, the *Dynamic Association FU* shall not be invoked for a given RESPONDER system if:

- a) a *Dynamic Association FU* invocation is already running with the same RESPONDER system, with identical values of the parameters in the *TASEBindReq* primitive;
- or:
- b) a *Group Management FU* invocation or a *Group Definition FU* invocation is running for the same RESPONDER¹⁸⁾.

5.6.2.3.3 Invoking events

The INITIATOR part of the *Dynamic Association FU* may be invoked by:

- local request via the Coordinating Function, the original source of which is outside the scope of this report;
- The *Restart Reconfigure FU* (which in turn has been invoked by another *Dynamic Association FU* or *Permanent Association FU* invocation controlling a different association).

Invocation of the RESPONDER part of the *Dynamic Association FU* is triggered by reception of a valid *TASEBindInd* primitive.

As explained in the corresponding section of the *Permanent Association FU* description, the RESPONDER part of the *Dynamic Association FU* is identical to the RESPONDER part of the *Permanent Association FU*. Therefore, the RESPONDER part of the *Dynamic Association FU* will be referred to as simply the RESPONDER part of the *Association FU* throughout the remaining part of this description of the *Dynamic Association FU*.

¹⁸⁾ The purpose of this rule is to ensure the integrity of the Configuration integrity parameter, which is updated by both the *Group Management FU* and the *Group Definition FU* and utilized by the *Dynamic Association FU*.

5.6.2.4 Termination

5.6.2.4.1 Orderly termination

Orderly termination of the INITIATOR part of a *Dynamic Association FU* invocation may be triggered by a local request via the Coordinating Function, originating from:

- a higher-level function, which is outside the scope of this technical report¹⁹⁾
- the *Restart Reconfigure FU* invocation that invoked the *Dynamic Association FU*.

Orderly termination of the RESPONDER part of an *Association FU* invocation is triggered by reception of an *A-ReleaseInd* primitive.

5.6.2.4.2 Disruption

An invocation of the *Dynamic Association FU* may be disrupted by:

- fatal error in an invocation of any other FU currently being supported by the *Dynamic Association FU* invocation. See 5.6.2.2.3;
- the *Dynamic Association FU* invocation itself, upon fatal error. See 5.6.3.2.

5.6.3 Procedures

5.6.3.1 TASE.1 service primitives

The following elementary TASE.1 services are used by the *Dynamic Association FU*:

- *TASEBind*
- *TASEAbort*
- *TASEUnbind*

5.6.3.1.1 Sequence

The normal sequence of primitives is partitioned into two phases:

Phase 1: establishing the association

INITIATOR UE	TASE	RESPONDER UE
<i>TASEBindReq</i>	----->	<i>TASEBindInd</i>
<i>TASEBindConf</i>	<-----	<i>TASEBindResp</i>

Phase 2: orderly termination: releasing the association

INITIATOR UE	TASE	RESPONDER UE
<i>TASEUnbindReq</i>	----->	<i>TASEUnbindInd</i>
<i>TASEUnbindConf</i>	<-----	<i>TASEUnbindResp</i>

Rules:

- a) Any *TASEAbortInd* primitive received by the INITIATOR UE on the association, subsequent to the phase 1 sequence above, shall trigger, locally:
 - Disruption of the *Dynamic Association FU* invocation²⁰⁾.
 - Disruption of all other FUs that are running on the association.
- b) Any *TASEAbortInd* primitive received by the RESPONDER UE on the association, subsequent to the phase 1 sequence above, shall trigger, locally:
 - Disruption of the *Association FU* invocation²¹⁾.
 - Disruption of all other FUs that are running on the association.

¹⁹⁾ Consequently, this technical report places no restriction on the strategy adopted by the INITIATOR UE for determining the point in time at which a dynamic association will be released.

²⁰⁾ Unlike the *Permanent Association FU*, the *Dynamic Association FU* considers the *TASEAbortInd* primitive an error event.

²¹⁾ This disruption is indistinguishable from the normal action performed in the *Association FU* in the RESPONDER when the association is controlled by a *Permanent Association FU* invocation in the INITIATOR.

c) The time span between end of phase 1 and start of phase 2 is a local issue in the INITIATOR, outside the scope of this technical report²²⁾.

d) Whenever the restart marking *Restart, spontaneous management lost* is received by the INITIATOR UE with the *TASEBindConf* primitive in phase 1, the *Dynamic Association FU* shall try to invoke the *Restart Reactivate FU* for the RESPONDER in question.

e) If Configuration control field error²³⁾ is detected by the INITIATOR UE with the *TASEBindConf* primitive in phase 1, the *Dynamic Association FU* shall try to invoke the *Restart Reconfigure FU* for the RESPONDER in question.

f) The rules stated in the procedure for Configuration Set check in 5.4 also apply.

5.6.3.1.2 Parameter values

TASEBind:

Parameter	Req (INITIATOR)	Resp (RESPONDER)
Initiator	As specified in description of <i>TASEBindReq</i> call in Annex A of IEC 60870-6-501. The “Address part” shall be equal to the address of the INITIATOR’s own node as specified in 5.2. The “A-suffix” part shall be as specified in 5.2.1.	Copy of value from <i>Ind</i>
Acceptor	As specified in description of <i>TASEBindReq</i> call in Annex A of IEC 60870-6-501. The “Address part” shall be equal to the address of the RESPONDER’s own node as specified in 5.2. The “A-suffix” part shall be as specified in 5.2.1.	Copy of value from <i>Ind</i>
User-data	Any value allowed by the TAPI ^a	First octet (octet No. 0): Optional in the above. Octets No. 1 through 16: Control Field value. See the procedure for Configuration Set check, in 5.4 for coding rules.
	The security information field starts from octet 2 if present. The field is encoded as described in 9.6.	The security information field starts from octet 17 if present. The maximum length is 66 octets. The field is encoded as described in 9.6. The security information field must be present if security classes 2 or 3 are used. It must be present if security class 1 with two-way authentication is selected, as well.

^a Unsolicited and periodic data transfers never occur via associations set up by the *Dynamic Association FU*; only by the *Permanent Association FU*.

²²⁾ Three possible strategies:

- a) Enter phase 2 immediately, when there is no more traffic.
- b) Enter phase 2 after a fixed delay time after the occurrence of no more traffic.
- c) Enter phase 2 only after operator action.

²³⁾ See 5.4.

Parameter	Req (INITIATOR)	Resp (RESPONDER)
Result	(Not applicable)	Association established: = <i>result-ok</i> Association not established, due to error condition: Other value. See 5.6.3.2. (This means that the RESPONDER part of the <i>Association FU</i> has not been invoked, for the association).

TASEUnbind:

Parameter	Req (INITIATOR)	Resp (RESPONDER)
User-reason	As specified in description of <i>TASEUnbindReq</i> call in Annex A of IEC 60870-6-501. Choice outside scope of this report.	(Not applicable)
Result	(Not applicable)	Always equal to <i>result-ok</i>

5.6.3.2 Error handling

5.6.3.2.1 FU disruption

Disruption by another FU is only done indirectly:

Whenever a *Dynamic Association FU* invocation is disrupted, it shall in turn trigger direct disruption of any FU invocations that use the association controlled by the disrupted *Dynamic Association FU*.

Local clean-up procedures are not specified by this technical report.

5.6.3.2.2 Illegal invocation attempt

FU not present:

If the *Dynamic Association FU* is not present in an INITIATOR UE:

The handling of failed local invocation requests via the Coordinating Function is a local issue, outside the scope of this technical report.

Failing invocation attempts by the *Restart Reconfigure FU*: see description of the *Restart Reconfigure FU*.

In a RESPONDER UE, the *Association FU* must be present to be able to function at all.

FU present, but attempt illegal:

Attempts of multiple simultaneous invocations of the *Dynamic Association FU* for identical parameter sets via the Coordinating Function is a local issue, outside the scope of this technical report.

Attempted *Dynamic Association FU* invocation attempt violating rule b) in 5.6.2.3.2 (*Group Management FU* or *Group Definition FU* running): report the error back to the local Coordinating Function.

Failing invocation attempts by the *Restart Reconfigure FU*: see description of the *Restart Reconfigure FU*.

5.6.3.2.3 Incoming TASE.1 service primitive out of context

INITIATOR part:

State	<i>TASEBindConf</i>	<i>TASEUnbindConf</i>
FU not running	Ignore, or local error indication	Ignore, or local error indication
FU running, waiting for <i>TASEBindConf</i>	(Normal)	Go to idle state
Running	Ignore, or local error indication/logging	Go to idle state
Waiting for <i>TASEUnbindConf</i>	Go to idle state	(Normal)
NOTE The <i>TASEAbortInd</i> primitive is never considered out of context.		

RESPONDER part:

State	<i>TASEBindInd</i>	<i>TASEUnbindInd</i>
Idle	(Normal)	Issue normal <i>TASEUnbindResp</i>
Running	Ignore, or local error indication/logging	(Normal)
NOTE The <i>TASEAbortInd</i> primitive is never considered out of context.		

5.6.3.2.4 Timing errors

There can occur no timing errors in the INITIATOR part.

Error in RESPONDER part:

Error	Reaction from TASE.1	Specified action in FU
UE too late responding to <i>TASEBindInd</i>	In RESPONDER: <i>TASEAbortInd</i> , with Reason = <i>misbehaviour-of-local-service-user</i> . Local error from eventual attempt at issuing <i>TASEBindResp</i> In INITIATOR: <i>TASEBindConf</i> , with Result = <i>no-answer-from-remote-system</i> .	RESPONDER part: Go to idle state. INITIATOR part: See 5.6.3.2.6.
UE too late responding to <i>TASEUnbindInd</i>	In RESPONDER: <i>TASEAbortInd</i> , with Reason = <i>misbehaviour-of-local-Service-user</i> . Local error from any attempt at issuing <i>TASEUnbindResp</i> In INITIATOR: <i>TASEAbortInd</i> , with Reason = <i>no-answer-from-remote-system</i>	RESPONDER part: Handle as abrupt termination of association. INITIATOR part: Handle as abrupt termination of association.

5.6.3.2.5 Congestion error

Congestion errors do not occur in the *Dynamic Association FU*. See 4.2.1.

5.6.3.2.6 TASE.1 service primitive parameter errors

Errors in the *TASEBindInd* primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Lower level part of Initiator (identifying the INITIATOR node) not valid, according to local access tables	Issue <i>TASEBindResp</i> primitive with Result = <i>authentication-failure</i> . Enter idle state if in other state.
INITIATOR requested security class > 0, but the RESPONDER does not support any security mechanism	Issue <i>TASEBindResp</i> primitive with Result = <i>security-is-not-supported-by-service-user</i> . Enter idle state, if in other state.
INITIATOR requested security class > 0, but the length of the authentication information field is zero.	Issue <i>TASEBindResp</i> primitive with Result = <i>authentication-failure</i> . Enter idle state, if in other state.
INITIATOR requested a security class or security options not supported by the RESPONDER	Issue <i>TASEBindResp</i> primitive with Result = <i>incompatible-security-options-requested</i> . Enter idle state, if in other state.
INITIATOR requested field authentication or encipherment for a parameter not present in the selected ELCOM class or version	Issue <i>TASEBindResp</i> primitive with Result = <i>incompatible-security-options-requested</i> . Enter idle state, if in other state.

Error	Action in RESPONDER part of FU
INITIATOR requested a security option (field authentication or encipherment) not supported by the RESPONDER	Issue <i>TASEBindResp</i> primitive with Result = <i>incompatible-security-options-requested</i> . Enter idle state if in other state.
RESPONDER could not authenticate the INITIATOR on the basis of the received authentication information field	Issue <i>TASEBindResp</i> primitive with Result = <i>authentication-failure</i> . Enter idle state if in other state.

Note that User-data is always valid within the *Dynamic Association FU*, provided it is valid within the TAPI.

Errors in the *TASEUnbindInd* primitive (detected by the RESPONDER part):

The only error possible is illegal User-reason value, which is considered outside the scope of this report.

Errors in the *TASEBindConf* primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Initiator >< Initiator in corresponding <i>TASEBindReq</i> primitive, and Result = <i>result-ok</i> . ^a (INITIATOR FU identity mix-up in RESPONDER.)	Report the error to the Coordinating Function, and trigger local abrupt termination of the association.
Zero octets in User-data, and Result = <i>result-ok</i> .	Treat as normal User-data. NOT an error; enter running state as normal.
Result >< <i>result-ok</i>	Report the error to the local Coordinating Function, and enter the idle state.

^a NO checking of the Acceptor parameter is specified, in order to accommodate network address transformations: the RESPONDER is thus allowed to respond with a network address other than the address by which it is known to the INITIATOR.

Errors in the *TASEUnbindConf* primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Result >< <i>result-ok</i>	Report the error to the Coordinating Function, and trigger local abrupt termination of the association.

5.7 Test Association FU

Type: Primary.

5.7.1 Function

A *Test Association FU* invocation provides the following functional sequence:

- The INITIATOR UE solicits an “*I-am-well*” type answer from the RESPONDER UE, on an established association.
- The RESPONDER UE answers, on the same association.

5.7.2 Coordination rules

5.7.2.1 Association usage

All TASE.1 interactions that are part of one invocation of the *Test Association FU* are conveyed by one single association. The association may be permanent or dynamic, and shall have the characteristics as specified in 5.7.2.3.1.

5.7.2.2 Relation to other FUs

5.7.2.2.1 Invoking FUs

The *Test Association FU* is not invoked by any other FU.

5.7.2.2.2 Invoked FUs

The *Test Association FU* does not invoke any other FU.

5.7.2.2.3 Disrupting FUs

The *Test Association FU* may be disrupted by the *Permanent Association FU* (permanent association) or the *Dynamic Association FU* (dynamic association), subsequent to reception of the *TASEAbortInd* primitive for the association concerned.

5.7.2.2.4 Disrupted FUs

The *Test Association FU* does not disrupt any other FU.

5.7.2.3 Invocation

5.7.2.3.1 Prerequisites

The following FUs shall have been invoked preceding any invocation of the *Test Association FU*:

- The *Permanent Association FU* or the *Dynamic Association FU*

The *Permanent Association FU* (*Dynamic Association FU*) invocation shall still be running at the time of invocation of the *Test Association FU*.

The *Permanent Association FU* (*Dynamic Association FU*) shall have been invoked in order to create and maintain the association to be used for the interactions related to the current invocation of the *Test Association FU*. The *Test Association FU* may be invoked on any association, at the INITIATOR UE's discretion. However, if an association is to be established solely for the purpose of invoking the *Test Association FU*, this association shall have the A-suffix pair as specified in the table in 5.2.1.

5.7.2.3.2 Restrictions

Multiple simultaneous invocations of the *Test Association FU* on a single association are not allowed.

5.7.2.3.3 Invoking events

The INITIATOR part of the *Test Association FU* may be invoked by:

- Local request via the Coordinating Function, the original source of which is outside the scope of this technical report²⁴⁾.

Invocation of the RESPONDER part of the *Test Association FU* is attempted whenever a *TestConnectionInd* primitive is received via an association, regardless of the characteristics of the association.

5.7.2.4 Termination

5.7.2.4.1 Orderly termination

The INITIATOR part of a *Test Association FU* invocation always terminates itself in an orderly manner upon reception of the expected *TestConnectionConf* primitive.

The RESPONDER part of a *Test Association FU* invocation terminates itself in an orderly manner after issuing the *TestConnectionResp* primitive.

Such termination of the *Test Association FU* may also be triggered by congestion error. See 5.7.3.2.5.

5.7.2.4.2 Disruption

Both the INITIATOR and the RESPONDER part of a *Test Association FU* invocation may be disrupted by:

- Disruption of another FU invocation. See 5.7.2.2.3.

5.7.3 Procedures

5.7.3.1 TASE.1 service primitives

The following elementary TASE.1 service is used by the *Test Association FU*:

Test-Connection

5.7.3.1.1 Sequence

INITIATOR UE	TASE.1	RESPONDER UE
<i>TestConnectionReq</i> <i>TestConnectionConf</i>	-----> <-----	<i>TestConnectionInd</i> <i>TestConnectionResp</i>

²⁴⁾ Such invocations typically may be timer-driven or operator-triggered.

5.7.3.1.2 Parameter values

Test-Connection:

Parameter	Req (INITIATOR)	Resp (RESPONDER)
Result	(Not applicable)	Always equal to value <i>result-ok</i>

5.7.3.2 Error handling

5.7.3.2.1 FU disruption

Disruption by the *Permanent Association FU* (*Dynamic Association FU*):

Disruption of both the INITIATOR part and the RESPONDER part of the current invocation of the *Test Association FU* shall be triggered locally, as a part of the disruption procedure for the corresponding part of the *Permanent Association FU* (*Dynamic Association FU*).

Both parts of the current invocation of the *Test Association FU* shall be terminated gracefully, neither part attempting to issue any primitive associated with the termination itself²⁵⁾.

5.7.3.2.2 Illegal invocation attempt

FU not present:

If the *Test Association FU* is not present in an INITIATOR UE:

Invocation requests are always generated locally; see 5.7.2.3. Consequently, the handling of this type of error is a local issue, outside the scope of this report.

If the *Test Association FU* is not present in an RESPONDER UE:

The RESPONDER User Element shall ignore the incoming *TestConnectionInd* primitive altogether.

FU present, but attempt illegal:

Attempts of multiple simultaneous invocations of the *Test Association FU* on a given association in an INITIATOR UE is a local issue, outside the scope of this technical report.

Attempts of multiple simultaneous invocations of the *Test Association FU* on a given association in a RESPONDER UE are precluded by mechanisms within the TASE.1.

5.7.3.2.3 Incoming TASE.1 service primitive out of context

INITIATOR part:

State	<i>TestConnectionConf</i>
FU not running	Ignore, or local error indication
FU running, idle	Ignore, or local error indication
FU running, waiting for answer from RESPONDER	(Normal)

In the RESPONDER part, an incoming *TestConnectionInd* primitive is never considered out of context.

5.7.3.2.4 Timing errors

Error in INITIATOR part:

There can occur no timing errors in the INITIATOR part.

²⁵⁾ Local clean-up procedures are not specified by this technical report.

Error in RESPONDER part:

Error	Reaction from TASE.1	Specified action in FU
UE too late responding to <i>TestConnectionInd</i>	In RESPONDER: <i>TASEAbortInd</i> , with Reason = <i>misbehaviour-of-local-service-user</i> . Local error from eventual attempt at issuing <i>TestConnectionResp</i> In INITIATOR: <i>TestConnectionConf</i> , with Result = <i>remote-service-user-unavailable</i> .	RESPONDER part: Ignore, or local error indication INITIATOR part: See 5.7.3.2.6.

5.7.3.2.5 Congestion error

In INITIATOR part of the FU, occurring with a *TestConnectionReq* attempt:

- Terminate FU invocation locally. See 4.2.1.

In RESPONDER part of the FU, occurring with a *TestConnectionResp* attempt:

- Terminate FU invocation locally, as normal except for local error report if desired.
(INITIATOR part of FU will be terminated after timeout.)

5.7.3.2.6 TASE.1 service primitive parameter errors

Errors in the *TestConnectionConf* primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Result $\neq$ <i>result-ok</i>	No special action other than registering “ <i>Result not ok</i> ”.

6 The Data Identification Function Group

6.1 Group attributes

A group is a logical collection of data objects of similar characteristics. One object may be a member of more than one group. A valid group may contain $1^{26)}$ – 255 objects; see 6.1.3.

Groups may be created and/or modified in a RESPONDER system in two different ways:

- Via the TASE.1 interface, as specified by this technical report.
- By any means not involving the TASE.1 interface, not specified by this technical report (fixed predefined groups).

No FU shall during its handling of a group configuration or its associated data discriminate between the two possible configuration origins (TASE/non-TASE), as such. Discrimination based on attribute values reflecting the origin is both allowed and mandatory, however; see attribute Persistent, below.

Any group, irrespective of configuration origin, is characterised by the following set of attributes, described in subsequent sections:

- Group type;
- Group number;
- Group size;
- Object identifier size;
- Persistent;
- Static;
- Priority;
- Object set:

For each object in the group, the component attribute pair:

- Object number;
- Object identifier.

²⁶⁾ A group containing 0 object is created but not defined, and is not considered valid here.

The following attributes are set in the CS(R) by the *Group Management FU* invoked with function *G-Create*:

- Group type;
- Group number;
- Group size;
- Object identifier size;
- Persistent;
- Static;
- Priority.

The following attribute is set in the CS(R) by the *Group Definition FU*:

- Object set (some or all objects)

Only the following²⁷⁾ attributes shall be modified in the CS(R) by the *Group Management FU* invoked with function *G-Change*:

- Persistent (only transition from *false* to *true* is legal);
- Static;
- Priority.

6.1.1 Group type

The table below lists the group types that are standardized by the TASE.1 specification²⁸⁾. A short description is given for each type, and the FUs within the DATA TRANSFER FUNCTION GROUP that may be invoked for that group type are identified.

Group type names do not imply any restrictions on the usage.

Group type	Description	DATA TRANSFER FUs
<i>Measurand-group</i>	Floating-point values	DRFU, DPRFU, DUPFU, DUSFU
<i>Status-group</i>	Three-state values (on/off/between)	DRFU, DPRFU, DUPFU, DUSFU
<i>Discrete-group</i>	Two-complement integers	DRFU, DPRFU, DUPFU, DUSFU
<i>Logical-breaker-status-group</i>	Composite three-state values	DRFU, DPRFU, DUPFU, DUSFU
<i>Binary-command-group</i>	Two-state commands (on/off)	DSFU
<i>Analog-setpoint-group</i>	Floating-point command values	DSFU
<i>Digital-setpoint-group</i>	Two-complement integer command values	DSFU
<i>Text-message-group</i>	General ASCII character string	DRFU, DPRFU, DUPFU, DUSFU

Further information may be found in the individual FU descriptions and in relevant TASE.1 reportation (IEC 60870-6-501 and IEC 60870-6-502).

6.1.2 Group number

Each group has an identifying associated integer, for use with the TAPI²⁹⁾. This Group number shall be unique within the scope of the Configuration Set to which it belongs. Consequently, in any INITIATOR or RESPONDER system, the Group number is by itself sufficient information to identify the group.

Legal value for Group numbers are 1 – 32767. Note that Group number 0 is not allowed³⁰⁾.

Group numbers are proposed by the INITIATOR UE to the RESPONDER UE, within a *Group Management FU* invocation for that number, with function *G-Create*.

²⁷⁾ See IEC 60870-6-501, 9.4.

²⁸⁾ Additional group types, defined for regional conventions, are allowed.

²⁹⁾ For local use, TASE.1 groups may well be identified by other means. However, at the TAPI, the identifying unit is always this group number.

³⁰⁾ In the *MixedDataReq* PDU, for example, group number 0 terminates the data field.

The RESPONDER UE may accept or reject a proposed number, according to local conditions³¹⁾. As a minimum, however, any RESPONDER UE shall support the following range of Group numbers: 1 – 30. Because the different Configuration Sets that any INITIATOR or RESPONDER system shares shall be independently defined, any RESPONDER system is required to support use of a Group number given by the peer INITIATOR system with any number of partners simultaneously. The total number of groups in a UE is not restricted by other than implementation dependent reasons.

Group numbers shall not be logically coupled with association identifiers.

6.1.3 Group size

Group size is the maximum number of objects that the attribute Object set of the group may contain. The legal value range is 1 – 255.

Group sizes are proposed by the INITIATOR UE to the RESPONDER UE in a manner similar to Group numbers (above). The RESPONDER UE may support only a subset of the legal value range. Handling of *Group Management FU (G-Create)* failures because of this, is not specified by this report. As a minimum, any RESPONDER UE shall support the following value of Group size: 20.

Group size is denoted by the parameter MaxObj.

6.1.4 Object identifier size

Object identifier size is the maximum number of octets that any Object identifier within the attribute Object set of the group may contain, excluding the length indicator octet. The legal range is 1 – 255.

Object identifier sizes are proposed by the INITIATOR UE to the RESPONDER UE in a manner similar to Group numbers (above). The RESPONDER UE may support only a subset of the legal value range. Handling of *Group Management FU (G-Create)* failures because of this, is not specified by this report. As a minimum, any RESPONDER UE shall support the following value of Object identifier size: 12.

Object identifier size is denoted by the parameter MaxObjSize.

6.1.5 Persistent

A boolean value, with the following semantics:

- true:* The group can be neither deleted/changed nor redefined via the TASE.1 interface. Any invocation of the *Group Management FU* or the *Group Definition FU* shall fail for such groups^a. Such groups are termed permanent, and if also of non-TASE.1 origin, predefined.
- false:* The group configuration may be manipulated at will via the TASE.1 interface. Such groups are termed non-permanent, or dynamic.

^a Rules for manipulation by non-TASE.1 means are not within the scope of this technical report.

Note that if it is desirable to exclude a group configuration of non-TASE.1 origin from any manipulative access via the TASE.1 interface, it suffices to set the attribute Persistent = *true* for that group in the CS(R).

Note also that changing Persistent from *false* to *true* in the CS(R) with the *Group Management FU* is allowed, effectively transforming any dynamic group into a permanent one. Transformation in the opposite direction, on the other hand, is illegal: a permanent group must always be changed or removed by non-TASE.1 means, if necessary.

6.1.6 Static

A boolean value, with the following semantics:

- true:* The object set of the group can be neither defined nor redefined with a *Group Definition FU* invocation for the group, except for the case of the group containing no Object identifier in the CS(R), which shall be treated as if Static were equal to *false*, by the RESPONDER UE^a.
- false:* The object set of the group may be defined or redefined at will, with *Group Definition FU* invocations for the group.

^a The purpose of this exception is to enable reconfiguration (invocation of *Restart Reconfigure FU*) of groups for which Static = *true*.

³¹⁾ For example, there may exist permanent inconsistencies between the CS(I) and CS(R), concerning fixed, predefined groups: a group number may be free from the INITIATOR UE's point of view [in the CS(I)], but occupied by a fixed, predefined group in the CS(R). Also, the RESPONDER UE may not be designed to support the full range of legal group numbers. The RESPONDER UE will have to reject the proposed group number in both these cases. Recovery mechanisms for such situations are not specified by this technical report.

The value of Persistent shall always override the value of Static, allowing (re)definition only when Persistent = *false*.

6.1.7 Priority

Priority is an integer value within the range 0 – 15, governing:

a) For Spontaneous Data Transfer:

The mechanism for ordering the output data queue across all *Unsolicited Spontaneous Data Transfer FUs* and invocations in a given RESPONDER system: Data belonging to a group with a given Priority value shall be given precedence over all data belonging to groups with lower Priority values. See 7.5 for details.

b) For Periodic Data Transfer:

The period lengths in the RESPONDER UE for the individual *Unsolicited Periodic Data Transfer FU* invocations, in an indirect manner. See 7.4.

Whether Priority will have the effect of governing output data queue or period length on behalf of a given group is determined exclusively by which FU is running for the group. The distinction is not a part of the group configuration, as such.

Priority values are proposed by the INITIATOR UE to the RESPONDER UE in a manner similar to Group numbers (above).

A RESPONDER UE may or may not support this priority mechanism³²⁾. RESPONDER UEs that support the mechanism shall accept all Priority values within the legal range³³⁾ RESPONDER UEs that do not support the mechanism, shall accept only the value 0. Handling of *Group Management FU* (*G-Create/G-Change*) failures because of a RESPONDER UE not supporting the mechanism, is not specified by this technical report.

6.1.8 Object set

The Object set of a group is the set of symbolic data value identifiers, or Object identifiers, for the group. Within the group, each Object identifier is given a reference number: the Object number. A given Object identifier may be a member of any number of groups, and may have different Object number in different groups.

The *Group Definition FU* is the means by which an INITIATOR UE instructs the RESPONDER system to bring the Object sets of (re)definable groups at the CS(R) into accordance with the CS(I) at the INITIATOR system.

Exact match between the CS(I) and the CS(R) for fixed, predefined groups must be ensured by separate non-TASE.1 means, not within the scope of this technical report.

6.1.8.1 Object number

The reference number of an Object identifier within a given group. The legal range is 1 – <value of Group size>. Note that Object number 0 is not legal.

Object numbers are assigned to the group definition in the CS(R) by the RESPONDER UE implicitly, by the order in which the individual Object identifiers appear in the Objid parameter of the *Define Group Ind* service primitive; see 6.3. The value of the Index1 parameter of the primitive directly determines the Object number of the first Object identifier in the Objid parameter. The next Object identifier in the Objid parameter is given Object number Index1+1, the next thereafter is given Index1+2, and so on, until either the Objid parameter is exhausted or Object number Index2 has been assigned.

6.1.8.2 Object identifier

The Object identifier serves as the logical link between the TASE.1 environment and the local database/data-acquisition environment in an INITIATOR/RESPONDER system: the INITIATOR/RESPONDER system shall use the Object identifier in order to identify³⁴⁾ which local data value is to be logically associated with a given combination of Group number and Object number. Further specification of database/data-acquisition interface mechanisms is outside the scope of this technical report.

³²⁾ Support of this mechanism shall mean support of both output queue ordering and period length determination.

³³⁾ However, the granularity of actual decisions may be less than 1/15: local subranges with equal effective value of Priority may be defined, but at least two effective values shall be the result.

³⁴⁾ Within a given system or site, objects may be identified by other mechanisms, with local reference to the corresponding TASE.1 Object identifier.

Object identifiers are copied by the RESPONDER UE from the Objid parameter of the *DefineGroupInd* service primitive into the CS(R); see 6.3.

6.2 Group Management FU

Type: Primary.

6.2.1 Function

A *Group Management FU* invocation allows the invoking INITIATOR UE to modify group attributes in the CS(R) of a RESPONDER UE, with the following effect, according to specified *Group Management FU* function:

Function	Effect
<i>G-Create</i>	Creates new group: allocates a new set of attributes, with specified values
<i>G-Change</i>	Changes existing attribute values, for a group
<i>G-Delete</i>	Deletes a single group: de-allocates the corresponding set of attributes
<i>Delete-all-groups</i>	Deletes ALL groups: de-allocates the set of attributes for all groups in the CS(R)

6.2.2 Coordination rules

6.2.2.1 Association usage

All TASE.1 interactions that are part of one invocation of the *Group Management FU* are conveyed by one single association. The association shall have the characteristics as specified in 6.2.2.3.1.

6.2.2.2 Relation to other FUs

6.2.2.2.1 Invoking FUs

The *Group Management FU* may be invoked by:

- *Group Configuration FU*;
- *Restart Reconfigure FU*.

6.2.2.2.2 Invoked FUs

The *Group Management FU* shall not invoke any other FU.

6.2.2.2.3 Disrupting FUs

The *Group Management FU* may be disrupted by:

- *Permanent Association FU*;
- *Dynamic Association FU*.

6.2.2.2.4 Disrupted FUs

The *Group Management FU* may disrupt the following FUs:

- *Group Configuration FU*;
- *Requested Data Transfer FU*;
- *Periodically Requested Data Transfer FU*;
- *Unsolicited Data Transfer FU*;
- *Unsolicited Periodic Data Transfer FU*;
- *Unsolicited Spontaneous Data Transfer FU*;
- *Supervisory Control Data Transfer FU*.

6.2.2.3 Invocation

6.2.2.3.1 Prerequisites

One³⁵⁾ of the following FUs shall have been invoked preceding any invocation of the *Group Management FU*:

- *Permanent Association FU*;
- *Dynamic Association FU*.

³⁵⁾ Subject to local decision in the INITIATOR UE.

The *Permanent Association FU* or *Dynamic Association FU* invocation shall still be running at the time of invocation of the *Group Management FU*. In the case of *Permanent Association FU* the association maintained by the invocation shall be running (not temporarily broken) at that time.

The *Permanent Association FU* or *Dynamic Association FU* shall have been invoked in order to create (and, for the *Permanent Association FU*, also to maintain) the association to be used for the interactions related to the current invocation of the *Group Management FU*, with the A-suffix pair specified by the table in 5.2.1.

6.2.2.3.2 Restrictions

For any given INITIATOR/RESPONDER system combination, multiple simultaneous invocations of the *Group Management FU* for any given group are not allowed.

A *Group Management FU* shall not be invoked while a *Group Definition FU* is running for the group involved.

A *Group Management FU* shall not be invoked while a *Group Management FU* invocation or a *Group Definition FU* invocation is running for another group on another association with the same partner³⁶⁾.

6.2.2.3.3 Invoking events

The INITIATOR part of the *Group Management FU* may be invoked by:

- local request via the Coordinating Function, the original source of which is outside the scope of this report;
- the *Group Configuration FU*.

Invocation of the RESPONDER part of the *Group Management FU* is attempted whenever a valid *GroupMgntInd* primitive is received via an association with the characteristics as defined for the *Group Management FU*.

6.2.2.4 Termination

6.2.2.4.1 Orderly termination

Orderly termination of the INITIATOR part of a *Group Management FU* invocation may be triggered by:

- congestion error. See 6.2.3.2.5;
- reception of a *GroupMgntConf* service primitive after issuing a *GroupMgntReq* service primitive (normal termination).

The RESPONDER part of a *Group Management FU* invocation always terminates itself in an orderly manner (normal termination) after issuing a *GroupMgntResp* service primitive.

6.2.2.4.2 Disruption

Both the INITIATOR and the RESPONDER part of a *Group Management FU* invocation may be disrupted by:

- disruption of another FU invocation. See 6.2.2.2.3;
- a fatal error condition encountered during operation. See 6.2.3.2.

6.2.3 Procedures

6.2.3.1 TASE.1 service primitives

The following elementary TASE.1 service is used by the *Group Management FU*:

GroupMgnt

6.2.3.1.1 Sequence

The normal sequence of primitives is as follows:

INITIATOR UE	TASE.1	RESPONDER UE
<i>GroupMgntReq</i>	----->	<i>GroupMgntInd</i>
<i>GroupMgntConf</i>	<-----	<i>GroupMgntResp</i>

³⁶⁾ The purpose of this rule is to ensure the integrity of the configuration integrity control parameter for the INITIATOR system at the RESPONDER system. This parameter is updated by both the *Group Management FU* and the *Group Definition FU*; see 5.4.

Rules:

- a) Upon reception of a Valid *GroupMgmtInd* service primitive, the RESPONDER UE shall:
- Update group attributes in the CS(R) with information from the primitive:

Attribute	Parameter	Function
Group type	Group-type	<i>G-Create</i>
Group number	Group-number (Gnr)	<i>G-Create, G-Delete, Delete-all-groups</i>
Persistent	Persist	<i>G-Create, G-Change^a</i>
Static	Static	<i>G-Create, G-Change</i>
Priority	Priority	<i>G-Create, G-Change</i>
Group size	MaxObj	<i>G-Create</i>
Object identifier size	MaxObjSize	<i>G-Create</i>

^a The attribute “Persistent” is not allowed to change *from true to false*, only from *false to true*.

— Generate new value of the configuration integrity control parameter and store it as part of the data specific for the current INITIATOR UE, in the RESPONDER system³⁷⁾.

— Report the new value of the configuration integrity control parameter back to the INITIATOR UE, via the CF parameter in the *GroupMgmtResp* service primitive.

- b) The RESPONDER UE shall return “OK” status (parameter Result = *result-ok*) if and only if the configuration integrity control parameter has been updated OK.
- c) Upon reception of a valid *GroupMgmtConf* service primitive, the INITIATOR UE shall store the value of the configuration integrity control parameter received from the primitive as part of the data specific for the current RESPONDER UE, in the INITIATOR system³⁸⁾.
- d) The rules stated in 6.1, above, also apply here.
- e) The rules stated in the procedure for Configuration Set update, in 5.4, also apply.

6.2.3.1.2 Parameter values

GroupMgmt:

Parameter	Req (INITIATOR)	Resp (RESPONDER)
Function	<i>G-Create, G-Change, G-Delete</i> or <i>Delete-all-groups</i> , according to the function with which the FU is invoked	Copy of value from <i>Ind</i>
Gtype (Group type)	Value <i>Measurand-group, Status-group, Discrete-group, Logical-breaker-status-group, Binary-command-group, Analog-setpoint-group, Digital-setpoint-group</i> or <i>Text-message-group</i> . Function = <i>G-Create</i> : Value of attribute Group type to be included in the CS(R) for group No. Group-number. Function = <i>G-Change</i> : Value shall be equal to value of attribute Group type in the CS(R) for group No. Group-number, for the request to be legal. Function = <i>G-Delete</i> or <i>Delete-all-groups</i> : Value is irrelevant.	Copy of value from <i>Ind</i>

³⁷⁾ See the APDU description, about the configuration integrity control parameter and associated functions.

³⁸⁾ The INITIATOR UE will later check his value of the configuration integrity control parameter against the value conveyed by the *TASEBindConf* primitive, during subsequent *Permanent Association FU* or *Dynamic Association FU* invocations against the same RESPONDER system. This is done in order to determine whether the group configuration is still to be considered valid or not: see clause 5.

Parameter	Req (INITIATOR)	Resp (RESPONDER)
Gnr (Group-number)	Reference number for the group in question. Function = <i>G-Create</i> : Value of attribute Group number in set of attributes to be created in the CS(R). Function = <i>G-Change</i> : Value of attribute Group number in set of attributes in the CS(R), for which attributes are to be modified Function = <i>G-Delete</i> : Value of attribute Group number in set of attributes to be removed from the CS(R). Function = <i>Delete-all-groups</i> : Value is irrelevant.	Copy of value from <i>Ind</i>
CF	(Not applicable)	Control Field value for the INITIATOR UE that invoked the <i>Group Management FU</i> . See 5.4.
MaxObj	Maximum number of objects in group. Function <i>G-Create</i> : Value of attribute Group size to be included in the CS(R) for group No. Group-number. Function = <i>G-Change</i> : Value shall be equal to value of attribute Group size in the CS(R) for group No. Group-number, for the request to be legal. Function = <i>G-Delete</i> or <i>Delete-all-groups</i> : Value is irrelevant.	(Not applicable)
MaxObjSize	Max Object identifier size (in octets). Function <i>G-Create</i> : Value of attribute Object identifier size to be included in the CS(R) for group No. Group-number. Function = <i>G-Change</i> : Value shall be equal to value of attribute Object identifier size in the CS(R) for group No. Group-number, for the request to be legal. Function = <i>G-Delete</i> or <i>Delete-all-groups</i> : Value is irrelevant.	(Not applicable)
Persist	Persistent flag. Function = <i>G-Create</i> or <i>G-Change</i> : Value of attribute Persistent to be included in the CS(R) for group No. Group-number. Function = <i>G-Delete</i> or <i>Delete-all-groups</i> : Value is irrelevant.	(Not applicable)