 <p>SINTEF Energy Research</p> <p>Address: 7034 Trondheim NORWAY</p> <p>Reception: Sem Sælands vei 11 Telephone: +47 73 59 72 00 Telefax: +47 73 59 72 50</p> <p>http://www.energy.sintef.no</p> <p>E. No.: NO 939 350 675</p>		<h1>TECHNICAL REPORT</h1>	
		SUBJECT/TASK (title)	
		Elcom-90 User Element Conventions	
		CONTRIBUTOR(S)	
		ELCOM Working Group Convener Ove Grande	
		CLIENT(S)	
		Joint Project: ABB AS, Siemens AS, SINTEF Energy Research AS, Statnett SF	
TR NO.	DATE	CLIENT'S REF.	PROJECT NO.
A3825.03	2009-12-16		12X513
ELECTRONIC FILE CODE		RESPONSIBLE (NAME, SIGN.)	CLASSIFICATION
		Ove Grande	Unrestricted
ISBN NO.	REPORT TYPE	RESEARCH DIRECTOR (NAME, SIGN.)	COPIES PAGES
		Petter Støa	295
DIVISION		LOCATION	LOCAL TELEFAX
Energy Systems		Sem Sælandsvei 11	+47 73 59 72 50
RESULT (summary)			
<p>This document is one of a series of technical reports which form the complete ELCOM-90 documentation. This is version .03 of the report with minor changes regarding responsible people and references. Future updates and new versions will NOT be published for this reason. New versions will only be submitted when technical changes are made.</p> <p>Please see SINTEF's homepage at: http://www.sintef.no/ELCOM-90. From here you can download the latest version of all relevant documents as pdf-files for free.</p>			
<h2>KEYWORDS</h2>			
SELECTED BY AUTHOR(S)			

TABLE OF CONTENTS

0. INTRODUCTION	0-1
1. ABOUT THIS DOCUMENT	1-1
2. ASSOCIATED DOCUMENTS	2-1
3. DEFINITIONS AND ABBREVIATIONS	3-1
4. THE ELCOM USER ELEMENT	4-1
5. THE ASSOCIATION MANAGEMENT FUNCTION GROUP	5-1
6. THE DATA IDENTIFICATION FUNCTION GROUP	6-1
7. THE DATA TRANSFER FUNCTION GROUP	7-1
8. THE RESTART FUNCTION GROUP	8-1
9. SECURITY	9-1
APPENDIX A: Encoding of user defined data parameters	11-1
APPENDIX B: Guide lines for local conventions	12-1
APPENDIX C: FU invocation hierarchy	13-1
APPENDIX D: FU disruption hierarchy	14-1
APPENDIX E: Use of result code values	15-1
APPENDIX F: Communicating with Elcom-83 systems	16-1
APPENDIX G: Address formats	17-1
APPENDIX H: State diagram descriptions of functional units	18-1

This document contains ten sections, (section 0 – section 9) and eight appendices (appendix A – appendix H). Each section and each appendix starts with a separate, detailed table of contents.

Section 0 : INTRODUCTION

TABLE OF CONTENTS

0. INTRODUCTION

0-1

0. INTRODUCTION

The Elcom Application Programming Interface (EAPI) [8] provides a set of well-defined functions to an application programmer wishing to base an application process on the Elcom protocol for real-time data acquisition and setpoint control. However, the EAPI specification does not by itself ensure full functional integration between any two Elcom application processes.

The purpose of this technical report is twofold:

1. To provide a consistent set of rules for the use of the EAPI, or more specifically, rules for:
 - usage of EAPI parameter values
 - sequencing of Elcom Application Service Element service primitives.
2. To be used as a general reference for specifying the capabilities of Elcom application processes. In order to achieve this, a number of "atomic" function packets, or Functional Units, are defined: The interoperational characteristics of any Elcom application process claiming conformity with this document shall be specified by quoting the Functional Units it supports.

Section 1 : ABOUT THIS DOCUMENT

TABLE OF CONTENTS

1. ABOUT THIS DOCUMENT	1-2
1.1. Scope and object	1-2
1.2. Structure	1-3
1.3	Acknowledgement 1-4

1. ABOUT THIS DOCUMENT

1.1. Scope and object

This technical report defines rules for usage of the **Elcom Application Programming Interface (EAPI)** only, 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 Elcom Service Element (EASE) itself, and will typically constitute further restrictions on parameter values, etc. The EASE's own restrictions and rules apply throughout. They are not explicitly restated in this document.¹ Whenever the present document allows simultaneous Functional Unit invocations on one association, adherence to the EASE primitive sequencing rules² are implicitly assumed.

Internal Functional Unit structures, Coordinating Function structure (see section 4 "THE ELCOM USER ELEMENT", below) 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 EASE service primitive sequences.

Also not within scope are all non-communicational aspects of application processes, 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 mode of ELCOM User Element and to identify the functional units (section 4). The functional units are described in minute detail, grouped into one function group per section (sections 5 through 8).

¹Consequently, handling of purely local EAPI errors are outside the scope of this document. All specified EASE service primitives of types *req.* and *res.* are always assumed to be issued within legal context and have legal parameter values, as far as the EAPI is concerned. For the special case of local call status indicating flow control problems (status value = -1), see the "General" section on "Functional Units", chapter 4.2.1.

²The EASE primitive sequencing rules are embodied in Appendices B (State Diagrams) and C (Decision Tables) of the EASE Protocol Specification [10]; expressed as the set of legal state transitions.

1.2. Structure

Section 2 contains a list of the documents on which this document is based, and to which it is referring.

The definitions and abbreviations used throughout this document are presented in **Section 3**.

Section 4 presents the underlying conceptual model of the Elcom User Element, and identifies the Functional Units which are the main subject of this document.

Sections 5 through 8 describe the Functional Units in minute detail, grouped into one Function Group per Section.

Security issues are addressed in **Section 9**.

The structure of the "Data" parameter of the data transfer primitives and the user data parameter in the association establishing primitive are described in **APPENDIX A**, together with related information: Data types, quality codes etc.

APPENDIX B presents a set of hints and recommendations for users wishing to define local conventions, or extensions to this document, on a bilateral basis.

APPENDIX C and **APPENDIX D** gives condensed versions of the contents of sections Invoking/Invoked FUs and Disrupting/Disrupted FUs, respectively, for the FUs defined by this document.

The use of the different values defined for the Result parameter in outgoing EASE service primitive is summarized in **APPENDIX E**.

APPENDIX F lists a set of issues to consider when an Elcom-90 based application is to communicate with an Elcom-83 based application.

APPENDIX G describes the address formats for TCP/IP and X.25.

Finally, an **APPENDIX H** is provided, supplying the reader with some examples of implementing the rules of this document, in the form of hypothetical state machines.

1.3 Acknowledgement

This document is the result of a joint work where the following persons have contributed:

Bakken, Ruth	Statnett SF
Bolsø, Anne-Grethe	ABB Energi AS
Conrad, Reinhold	Siemens AS
Eggen, Nils	Powel ASA
Gjerde, Ole	Statnett SF
Hegge, Jan	Sintef Energy Research AS
Kaasa, Harald	Statnett SF
Krystad, Jens	Powel Data AS
Larsen, Anders	Statnett SF
Lund, Tormod	ABB Kraft AS
Magnus, Helge	Netconsult
Paulsen, Alf	Statnett SF
Pille, Hans	KEMA
Randen, Hans	Statnett SF
Rindal, Lars	Siemens AS
Stene, Birger	Sintef Energy Research AS
Storve, Jan	Avenir
Sveen, Arne	ABB Energi AS
Torkilseng, Åge	AS Salten Kraftsamband

The ELCOM Working Group consists at present of the following members:

Grande, Ove, convener	Sintef Energy Research AS
Eggen, Nils	Powel ASA
Kaasa, Harald	Statnett SF
Larsen, Anders	Statnett SF
Lund, Tormod	ABB Kraft AS
Rindal, Lars	Siemens AS
Torkilseng, Åge	AS Salten Kraftsamband

Section 2 : ASSOCIATED DOCUMENTS

TABLE OF CONTENTS

2.	ASSOCIATED DOCUMENTS	2-1
2.1.	ELCOM-83 documentation	2-1
2.2.	ELCOM-90 documentation	2-1
2.3.	Other References	2-2

2. ASSOCIATED DOCUMENTS

2.1. ELCOM-83 documentation

- [1]: TR 3522: **ELCOM-83 Application Service Definition**
Norwegian Electric Power Research Institute, Trondheim, Norway, 1988-07-05
- [2]: TR 3528: **ELCOM-83 Application Protocol Definition**
Norwegian Electric Power Research Institute, Trondheim, Norway, 1988-07-14
- [3]: TR 3523: **ELCOM-83 Definition of Local Application Interface**
Norwegian Electric Power Research Institute, Trondheim, Norway, 1988-07-05
- [4]: TR 3524: **ELCOM-83 Presentation Service Definition**
Norwegian Electric Power Research Institute, Trondheim, Norway, 1988-07-06
- [5]: TR 3527: **ELCOM-83 Presentation Protocol Definition**
Norwegian Electric Power Research Institute, Trondheim, Norway, 1988-07-13
- [6]: TR 3532: **ELCOM-83 Definition of Local Presentation Interface**
Norwegian Electric Power Research Institute, Trondheim, Norway, 1988-09-12
- [7]: TR 3649: **ELCOM-83 Conventions**
Norwegian Electric Power Research Institute, Trondheim, Norway, 1989-12-20
ISBN 82-594-0086-3

2.2. ELCOM-90 documentation

This document is one of a series of technical reports which form the complete ELCOM-90 documentation. Below you will find the numbers and titles for all the associated technical reports. New versions may be submitted when technical changes are made. Please see SINTEF's homepage at: <http://www.sintef.no//ELCOM-90>. From here you can download the latest version of all relevant documents as pdf-files for free.

- [8]: TR 3701: **ELCOM-90 Application Programming Interface Specification**
- [9]: TR 3702: **ELCOM-90 Application Service Element. Service Definition**
- [10]: TR 3703: **ELCOM-90 Application Service Element. Protocol Specification**
- [11]: TR 3704: **ELCOM-90 Presentation Programming Interface Specification**
- [12]: TR 3705: **ELCOM-90 Presentation Service Definition**
- [13]: TR 3706: **ELCOM-90 Presentation Protocol Specification**

- [14]: TR 3825: **ELCOM-90 User Element Conventions**
- [15]: TR A3933: **ELCOM-90 Local Conventions**
- [16] TR A4687: **PONG. The ELCOM net-watch procedure for TCP/IP networks**
- [17] TR A4124: **ELCOM-90 Application Service Element, User's manual.**
- [18] TR A6196: **Securing ELCOM-90 with TLS.**

2.3. Other References

- [19] ISO 7498 - 2: Information Processing Systems - Open Systems Interconnection Security Architecture
- [20] ITU-T X.509 (1988) The Directory - Authentication framework
- [21] ISO TR 8509: Service Conventions
- [22] ISO 7498-1 OSI Basic Reference Model
- [23] ISO 8348 Network Service Definition
- [24] John F. Wakerly: "Microcomputer Architecture and Programming", John Wiley & Sons Inc., 1981.

Section 3 : DEFINITIONS AND ABBREVIATIONS

TABLE OF CONTENTS

3.	DEFINITIONS AND ABBREVIATIONS	3-1
3.1.	Definitions	3-1
3.2.	Abbreviations	3-4

3. DEFINITIONS AND ABBREVIATIONS

3.1. Definitions

- ACCEPTOR address:** The unique identification, octet string, of the responding service user.
- 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 EASE 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 EASE is not able to receive *req.* or *res.* type service primitives, because of heavy traffic. General rules for handling congestion errors are given in chapter 4.2.1. Special rules per FU are given in the individual FU descriptions.
- Coordinating Function:** An Elcom User Element function that controls the local Functional Unit invocations.
- Creating a group:** Making a new group identity¹ 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, deallocating 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.
- EASE:** Elcom Application Service Element.
- Elcom partner:** An Elcom site with which a given INITIATOR UE or RESPONDER UE may communicate via the EASE.

¹Group number.

- Elcom provider:** The software component that implements the Elcom protocol in a given environment.
- Elcom system:** The set of User Elements, or single User Element, that utilise the Elcom provider that is addressed by the low-level part of a given Elcom address². 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 Elcom system.
- Function Group:** A named collection of Functional Units of related functionality.
- Functional Unit invocation:**
A specific instance of use of the Functional Unit.
- Functional Unit type:** A named collection of Functional Units of related action mechanisms.
- Functional Unit, or Elcom User Element Functional Unit:**
A named well-defined succession of EASE service primitives at the E APIs of two communicating Elcom systems, constituting a single co-operative functional capability of an Elcom INITIATOR User Element and its peer Elcom RESPONDER User Element³.
- Group:** A numbered set of named, and implicitly numbered, data objects in an Elcom 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, octet string, of an Initiator User Element.
- 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 Elcom 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 EASE.
- Low-level Elcom address:**
What is left of an Elcom address if the A-suffix character pair is removed.
- 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.

²An Elcom system may be addressed by more than one low-level Elcom address. For example, a number of different DTE numbers (which is one form of low-level Elcom addresses) will address the same Elcom system, if:

- All DTE numbers connect to X.25 lines attached to the device in which the Elcom system resides, and:
- All DTE numbers contain the single sub-address that is defined for Elcom-90.

³This is the basic definition. However, an FU may act exclusively through other FUs, having no specific EASE service primitive sequence associated with it. See about FU types, in chapter 4.2 "Functional Units (FUs)".

- Primary FU:** Primary FUs are the basic kind of FUs; these are characterized by their individual well defined sequence of EASE service primitives, and are always being invoked by an Initiator UE.
- Procedure:** Sequence of prescribed actions in an Elcom 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 Elcom 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:**
The peer communications UE of an INITIATOR UE.
- Secondary FU:** Secondary FUs have individual well defined EASE service primitive sequences, but are always invoked by a Responder UE, as a result of local decision in that user element.
- Subgroup:** A contiguous range of objects within a group definition.
- Transaction, or Elcom transaction:**
A specific instance of use of an elementary EASE service.
- User Element:** The ELCOM User Element is defined as that part of the Elcom Application Entity that is not part of the EASE/EAPI. It may be of either the initiator type or of the responder type (see chapter 4.1).

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
CS:	Configuration Set
CS(I):	The CS copy at the INITIATOR site
CS(R):	The CS copy at the RESPONDER site
<i>cnf.:</i>	<i>confirm</i>
DPFU:	Periodic Data Transfer FU
DPRFU:	Periodically Requested Data Transfer FU
DRFU:	Requested Data Transfer FU
DSFU:	Supervisory Control Data Transfer FU
DUFU:	Unsolicited Data Transfer FU
DUMFU:	Unsolicited Mixed Data Transfer FU
EAPI:	Elcom Application Programming Interface
EASE:	Elcom Application Service Element
FU:	(Elcom User Element) Functional Unit
GCFU:	Group Configuration FU
GDFU:	Group Definition FU
GMFU:	Group Management FU
GRFU:	Group Readout FU
<i>ind.:</i>	<i>indication</i>
PDU:	Protocol Data Unit.
RAFU:	Restart Reactivate FU
<i>req.:</i>	<i>request</i>
<i>res.:</i>	<i>response</i>
RRFU:	Restart Reconfigure FU
UE:	User Element

Section 4 : THE ELCOM USER ELEMENT

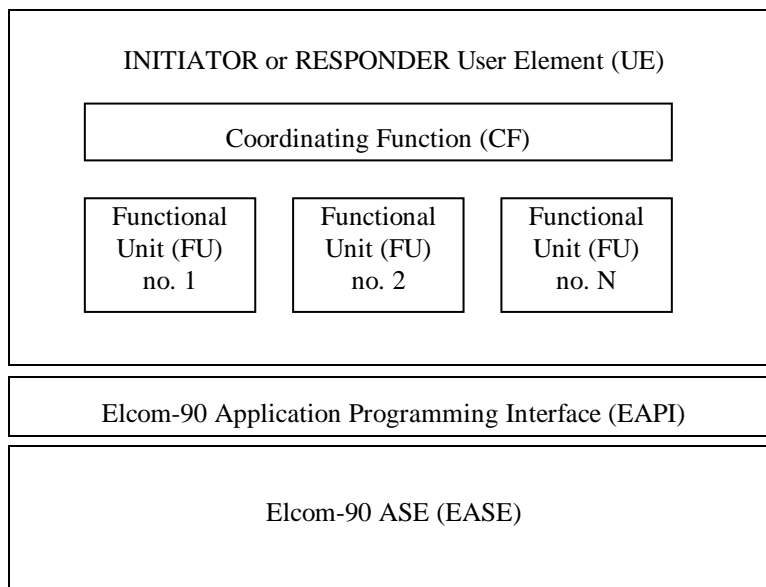
TABLE OF CONTENTS

4.	THE ELCOM USER ELEMENT	4-1
4.1.	Conceptual Model	4-1
4.2.	Functional Units (FUs)	4-3
4.2.1.	General	4-3
4.2.2.	Overview	4-4
4.3	Common template for the description of FU's	4-6

4. THE ELCOM USER ELEMENT

4.1. Conceptual Model

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



Conceptual model of an Elcom-90 Application Entity

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

The EASE 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 a UE¹.

¹FUs 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.

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 document 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 EASE service primitive sequences, and shall not be interpreted as an architectural requirement².

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 chapters "Relation to other FUs" and "Coordination rules".

To summarize:

FUs function as *objects for EASE 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 document, allowing for any manner of implementation as long as the stated rules are adhered to.

The individual FUs are described in considerable detail below.

²For 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 document imposes on a given Functional Unit in this respect are listed in the chapters "Multiple invocations" and "Illegal combinations" in the description of each Functional Unit, below.³

Local EAPI 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 retrials 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.

³Note, however, that the EASE imposes its own set of primitive sequencing rules, in addition to those mentioned here. See chapter 1.1 "Scope and object".

4.2.2. Overview

The FU concept is presented in the preceding chapters. In this chapter each FU defined by this document 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 3 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 EASE service primitives, and always being invoked by an INITIATOR UE.

Example: The *Unsolicited Data Transfer FU*.

Secondary FUs have individual well-defined EASE service primitive sequences, but are always invoked by a RESPONDER UE, as a result of local decision in that UE. They do have a specific INITIATOR part.

Example: The *Unsolicited Mixed Data Transfer FU*.

Composite FUs act via invocation of other FUs by an INITIATOR UE, only. They have no associated specific EASE 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**.

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 **Periodic Data Transfer FU**. Acronym: **DPFU**. Type: **Primary**.
Transferring data associated with a single group 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 *Periodic data Transfer FU*.

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

The **Unsolicited Mixed Data Transfer FU**. Acronym: **DUMFU**. Type: **Secondary**.
Transferring data associated with any number of groups, from a RESPONDER UE to an INITIATOR UE, any number of times at the RESPONDER UE's discretion, after permission has been granted by the INITIATOR UE, for each of the groups involved. Permission granting and withdrawal is part of an *Unsolicited Data Transfer FU* or *Periodic data Transfer FU* invocation, for each group.

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 unsolicited, 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 description of FU's

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.

EASE service primitives:

List of elementary EASE 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 EASE 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 chapter **Restrictions**.

Incoming EASE service primitive out of context:

Handling of primitives *received* from the EASE, 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 EAPI is not able to accept another outgoing primitive because of heavy load.

EASE service primitive parameter errors:

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

Section 5 : THE ASSOCIATION MANAGEMENT FUNCTION GROUP

TABLE OF CONTENTS

5.	THE ASSOCIATION MANAGEMENT FUNCTION GROUP	5-1
5.1.	Addressing	5-1
5.1.1.	Lower-level address part	5-1
5.1.2.	A-suffices	5-2
5.1.3.	Coding of addresses	5-3
5.2.	Elcom class and version negotiation	5-4
5.3.	Spontaneous mode codes	5-5
5.4.	Restart codes	5-6
5.5.	Group configuration integrity control	5-7
5.6.	Permanent Association FU	5-9
5.6.1.	Function	5-9
5.6.2.	Coordination rules	5-9
5.6.2.1.	Association usage	5-9
5.6.2.2.	Relation to other FUs	5-9
5.6.2.2.1.	Invoking FUs	5-9
5.6.2.2.2.	Invoked FUs	5-10
5.6.2.2.3.	Disrupting FUs	5-10
5.6.2.2.4.	Disrupted FUs	5-10
5.6.2.3.	Invocation	5-10
5.6.2.3.1.	Prerequisites	5-10
5.6.2.3.2.	Restrictions	5-10
5.6.2.3.3.	Invoking events	5-11
5.6.2.4.	Termination	5-11
5.6.2.4.1.	Orderly termination	5-11
5.6.2.4.2.	Disruption	5-11
5.6.3.	Procedures	5-12
5.6.3.1.	EASE service primitives	5-12
5.6.3.1.1.	Sequence	5-12
5.6.3.1.2.	Parameter values	5-14
5.6.3.2.	Error handling	5-16
5.6.3.2.1.	FU disruption	5-16
5.6.3.2.2.	Illegal invocation attempt	5-16
5.6.3.2.3.	Incoming EASE service primitive out of context	5-17
5.6.3.2.4.	Timing errors	5-18
5.6.3.2.5.	Congestion error	5-18
5.6.3.2.6.	EASE service primitive parameter errors	5-19
5.7.	Dynamic Association FU	5-21
5.7.1.	Function	5-21
5.7.2.	Coordination rules	5-21
5.7.2.1.	Association usage	5-21
5.7.2.2.	Relation to other FUs	5-21
5.7.2.2.1.	Invoking FUs	5-21
5.7.2.2.2.	Invoked FUs	5-22

5.7.2.2.3.	Disrupting FUs	5-22
5.7.2.2.4.	Disrupted FUs	5-22
5.7.2.3.	Invocation	5-22
5.7.2.3.1.	Prerequisites	5-22
5.7.2.3.2.	Restrictions	5-23
5.7.2.3.3.	Invoking events	5-23
5.7.2.4.	Termination	5-24
5.7.2.4.1.	Orderly termination	5-24
5.7.2.4.2.	Disruption	5-24
5.7.3.	Procedures	5-24
5.7.3.1.	EASE service primitives	5-24
5.7.3.1.1.	Sequence	5-25
5.7.3.1.2.	Spontaneous mode codes	5-26
5.7.3.1.3.	Parameter values	5-27
5.7.3.2.	Error handling	5-29
5.7.3.2.1.	FU disruption	5-29
5.7.3.2.2.	Illegal invocation attempt	5-29
5.7.3.2.3.	Incoming EASE service primitive out of context	5-30
5.7.3.2.4.	Timing errors	5-31
5.7.3.2.5.	Congestion error	5-31
5.7.3.2.6.	EASE service primitive parameter errors	5-32
5.8.	Test Association FU	5-34
5.8.1.	Function	5-34
5.8.2.	Coordination rules	5-34
5.8.2.1.	Association usage	5-34
5.8.2.2.	Relation to other FUs	5-34
5.8.2.2.1.	Invoking FUs	5-34
5.8.2.2.2.	Invoked FUs	5-34
5.8.2.2.3.	Disrupting FUs	5-35
5.8.2.2.4.	Disrupted FUs	5-35
5.8.2.3.	Invocation	5-35
5.8.2.3.1.	Prerequisites	5-35
5.8.2.3.2.	Restrictions	5-35
5.8.2.3.3.	Invoking events	5-36
5.8.2.4.	Termination	5-36
5.8.2.4.1.	Orderly termination	5-36
5.8.2.4.2.	Disruption	5-36
5.8.3.	Procedures	5-37
5.8.3.1.	EASE service primitives	5-37
5.8.3.1.1.	Sequence	5-37
5.8.3.1.2.	Parameter values	5-37
5.8.3.2.	Error handling	5-38
5.8.3.2.1.	FU disruption	5-38
5.8.3.2.2.	Illegal invocation attempt	5-38
5.8.3.2.3.	Incoming EASE service primitive out of context	5-39
5.8.3.2.4.	Timing errors	5-40
5.8.3.2.5.	Congestion error	5-40
5.8.3.2.6.	EASE service primitive parameter errors	5-41

5. THE ASSOCIATION MANAGEMENT FUNCTION GROUP

5.1. Addressing

The function of an Elcom address is to specify an Application Entity in a target Elcom system. Elcom addresses are composed of two parts: The **lower-level address part** and the **A-suffix**. The lower-level part specifies an Elcom provider uniquely within the space of all Elcom providers, while The A-suffix specifies an Application Entity uniquely within the space of all Application Entities locally connected with a given Elcom provider.

The *Permanent Association FU* and the *Dynamic Association FU* utilize Elcom addresses coded into parameters Initiator and Acceptor for association establishment; see below.

The lower-level address contains information specifying a single INITIATOR or RESPONDER system uniquely. However, an INITIATOR or RESPONDER site has no unique lower-level address, unless some sort of dynamic routing of calls, not part of this document, is employed.

Any UE conforming to this document 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 Elcom address per site**. Each Elcom address shall correspond to a component INITIATOR or RESPONDER system within the site.

Elcom address selection is a function at Coordinating Function level.

At one responder site with one or several redundant systems the redundant systems shall not be reachable. An initiator user element should apply a trial-and-error strategy with successive FU invocations.

5.1.1. Lower-level address part

The lower-level address part has two components: The **transport protocol identifier** and the **transport address**.

The transport protocol identifier serves to identify the transport protocol used. It implicitly determines the format of the transport address. The transport address itself expresses the location of any given Elcom provider.

The Elcom-90 transport protocols are:

- X.25
- ISO Transport Protocol
- TCP

5.1.2. A-suffices

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

Function	Functional Units	Initiator	Responder
Accessing the CS(R) from the INITIATOR	GMFU, GDFU, GRFU	'AA'	'BA'
Unsolicited Data Transfer	DUFU, DRFU ¹	'AB'	'BB'
Periodic Data Transfer	DPFU, DRFU ²	'AC'	'BC'
Requested Data Transfer, scheduling information ³	DRFU	'AD'	'BD'
Requested Data Transfer, present or archived information	DRFU	'AE'	'BE'
Supervisory Control Data Transfer	DSFU	'AG'	'BG'
Test Association, when association is established solely for this purpose	ATFU	'AF'	'BF'

Note: 'AB' means the ASCII representation of the character pair AB, ie. the decimal value pair 65, 66.

Example: If an INITIATOR UE wants to initiate unsolicited 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 *A-Connect req.* primitive it will issue in order to establish an association over which it later will grant permission to send the unsolicited data.

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

¹An implementation of the DRFU may be restricted to only support present information (no archive information) on an association where an Unsolicited Data Transfer FU is already running.

²DRFU legal only if no *Periodic Data Transfer FU* invocation is currently active on the association in question.

³Schedules (future and historic) re power plant operation, energy exchange, discharges etc.

5.1.3. 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 lower-level address part (= N)
- Octets no. 1 through N: Lower-level address part
- Octet no. N+1: Binary number, expressing the number of octets in the A-suffix part (=M)⁴
- Octets no. N+2 through N+M+1: A-suffix part

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

The lower-level address part shall be coded as follows, using the same octet numbering scheme as above:

If value of octet no. 1 is within range 48 through 57 (ASCII '0' through '9'):

Elcom-83 address format:

- Octets no. 1 through N contain an ASCII coded X.25 DTE number

If not:

Elcom-90 address format:

- Octet no. 1 is a binary value containing the transport protocol identifier:

- 128: X.25

- 129: ISO Transport Protocol

- 130: TCP

- 131: (Reserved for future extension)

- 132: (Reserved for future extension)

Other values: Illegal

- Octets no. 2 through N contain a transport address, on the format of the transport protocol indicated by the transport protocol identifier.

Note 1:

The transport address is documented in Appendix G.

Note 2:

Mechanisms and rules for selecting outgoing line (eg. for network selection) for an *A-Connect req.* primitive over a given protocol are local issues, outside the scope of this document.

⁴Note that M is always =2 for the A-suffixes standardized by this document.

5.2. Elcom class and version negotiation

Elcom Version and Class values are coded into the Version parameter of the *A-Connect* service primitives. The following values are defined:

Version:	0: Elcom-83
	1: Elcom-90
Class:	0, 1, 2, 3

The reader is referred to [9], chapter 8, for an interpretation of the Class values.

During association establishment, the RESPONDER part of the *Permanent Association FU* and the RESPONDER part of the *Dynamic Association FU* shall conform to the following negotiation rules:

1. If the RESPONDER UE can support the functionality implied by the Version and Class values in the Version parameter of the *A-Connect ind.* primitive which invokes the RESPONDER part of the FU, the Version parameter of the outgoing *A-Connect res.* primitive shall have identical value.
2. If the RESPONDER UE cannot support the version identified by the Version value, the *A-Connect res.* primitive shall contain Result = *incompatible-versions*, and Version value shall express the resources available in the RESPONDER UE. The RESPONDER UE shall not consider association as established, in this case. Rules for determining the actual Version value to be returned in this case is a local issue, not within the scope of this document.
3. If the RESPONDER UE cannot support the required class, the *A-Connect res.* primitive shall contain Result = *o.k.*, and Class value shall express the resources available in the RESPONDER UE.

Correspondingly, the INITIATOR part of the *Permanent Association FU* and the INITIATOR part of the *Dynamic Association FU* shall conform to the following rules:

1. Association shall not be considered established unless the version identified by the Version value in the *A-Connect cnf.* primitive is equal to the Version value in the original *A-Connect req.* primitive.
2. If the value of the Result parameter in the *A-Connect cnf.* primitive is equal to *incompatible-versions*, the *Permanent (or Dynamic) Association FU* may be re-invoked, this time using the Version value received with the *A-Connect cnf.* primitive. Such re-invocation is the responsibility of the Coordinating Function, subject to local rules not within the scope of this document. Neither does this document impose a limit on the number of such re-invocations.

Also note that the Elcom provider itself imposes some Version and Class control on its local communication with the UE:

In the INITIATOR:

- If Class < 3 for an association, the primitives *A-Command-Transfer req.* and *A-Mixed-Data req.* are both locally illegal for that association
- If an association is set up with an Elcom-83 RESPONDER UE (Version = 0), *A-Get-Group req.* with Index1><0 or Index2><0 and *A-Def-Group req.* with Index1><0 or Index2><0 are both locally illegal for that association

In the RESPONDER:

- *A-Connect res.* with Version = 1 responding to Version = 0 is locally illegal.

5.3. Spontaneous mode codes

Whenever permission to send unsolicited or periodic data is granted by an INITIATOR UE to a RESPONDER UE, the decision whether actually to send unsolicited or periodically is made by the RESPONDER UE based on a **spontaneous mode code**, per association. This code is passed from the INITIATOR UE during association establishment, in the User-data parameter of the *A-Connect req.* primitive. Only the following spontaneous function codes are legal within the context of this document⁵:

Value	Effect	Functional Units invoked
'0' (= decimal 48)	Unsolicited mode	DUFU, DUMFU
'1' (= decimal 49)	Periodic mode	DPFU

Example: Whenever a RESPONDER UE receives an *A-Spont-Mgmt (start) ind.* primitive on an association which was established with the spontaneous mode code '0', it will attempt to invoke the *Unsolicited Data Transfer FU* or *Unsolicited Mixed Data Transfer FU* for the group in question.

Note that the value of the spontaneous mode code has no effect except in conjunction with the FUs listed in the table above. For all other FUs, the INITIATOR UE may put any value into the spontaneous mode code when establishing an association, as the RESPONDER UE shall not act upon it.

Spontaneous mode codes are included in this specification only for the purpose of backward compatibility with Elcom-83 based systems:

- The INITIATOR UE shall always use spontaneous mode codes with Elcom-83 based partners.
- The RESPONDER in an Elcom-90 based system shall use the A-suffix instead of spontaneous mode codes for distinguishing between periodic and unsolicited transfer, when communicating with Elcom-90 based systems, and shall use the spontaneous mode codes when communicating with Elcom-83 based systems.

⁵See APPENDIX A: Encoding of user defined data parameters

5.4. Restart codes

A RESPONDER UE may use a **restart code** in the User-data parameter of the *A-Connect res.* primitive in order to signal its need for a re-initialization by the INITIATOR, with respect to the CS(R) and/or permission to send unsolicited and/or periodic data. The INITIATOR will eventually respond to this request by invoking either the *Restart Reconfigure FU* or the *Restart Reactivate FU* for the RESPONDER system in question. Only the following restart codes are legal within the context of this document⁶:

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. The code shall only be used when establishing associations for spontaneous or periodic data transfer.	RAFU
'3' (= decimal 51)	<i>Restart, group management lost.</i> Total update of CS(R), followed by repetition of all permissions to send, required. This code shall only be used when communicating in the Elcom-83 compatible mode.	RRFU

The decision when to use which restart code is a local issue in the RESPONDER, outside the scope of this document. There is one exception, however: Rule no. 2 in the primitive sequence description for the *Permanent Association FU*.

The restart codes are significant for the INITIATOR, only when Result = *ok* in the *A-Connect-response* primitive (for Elcom-83 compatibility please ref. App. F, "Communicating with a Elcom-83 system").

⁶See APPENDIX A: Encoding of user defined data parameters

5.5. Group configuration integrity control

In Elcom-90, the 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) 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 *A-Group-Mgmt ind.* or *A-Def-Group ind.* primitive.

The Control Field value shall be an array of 9 elements. The elements shall be 16 bit integers⁷. 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 [8], chapter 5.5.3.
- Elements no. 7 through 8: 32-bit checksum, generated from the contents of the whole **CS(R)** after update:

Neither the checksum algorithm itself nor the rules for partitioning the 32-bit checksum value into two 16-bits integer values is specified by this document. The actual use of the checksum facility is not mandatory: It may be disabled entirely, the RESPONDER always setting both integer elements of the checksum part of the Control Field equal to a fixed value, for example 0.

The time stamp part of the new Control Field value shall be stored with the updated **CS(R)** at the RESPONDER site, overwriting the previous time stamp part in the local group configuration database.⁸

After the time stamp part of the new Control Field value has been successfully stored, the *A-Group-Mgmt res.* or *A-Def-Group res.* 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 *A-Group-Mgmt cnf.* or *A-Def-Group cnf.* 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.

⁷Implementations having "native" integers of more than 16 bit shall only use the 16 low-order bits of each integer element for storing significant Control Field data.

⁸Alternatively, the whole Control Field (time stamp plus checksum) may be stored with the **CS(R)**. In this case the Control Field returned with subsequent *A-Connect res.* primitives (see procedure for Configuration Set check below) may contain the stored checksum instead of the result of another checksum computation.

Procedure for Configuration Set check:

Whenever the *Permanent Association FU* or the *Dynamic Association FU* is invoked, the User-data parameter of the *A-Connect res.* 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 App. A, "Encoding of user defined parameters".

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 computed⁹ from the current **CS(R)**, using **the same algorithm and the same 32 to 16 bit partition rules** as during last Configuration Set update.

The first time a connection to a partner is established, no groups are defined. In this case it is recommended that the Responder returns a CS(R) containing all zeroes except CS(2)=1 and CS(3)=1.

The INITIATOR part of the *Permanent Association FU* or the *Dynamic Association FU* shall, upon reception of a valid *A-Connect cnf.* 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.

⁹Alternatively, the checksum may be fetched from the local database if it resides there; see footnote in procedure for Configuration Set update.

5.6. Permanent Association FU

Type: Primary.

5.6.1. Function

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

1. The INITIATOR UE establishes an association with a RESPONDER UE.
2. 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.
3. At any time later, whenever the *Permanent Association FU* registers the association as being broken (abruptly terminated), it will attempt to reestablish the association. Cyclic reestablishment attempts will proceed indefinitely, until success or intervention by a higher-order function via the Coordinating Function in the INITIATOR UE.

5.6.2. Coordination rules

5.6.2.1. Association usage

The FU invocation itself creates and maintains 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 *Permanent Association FU* shall not be invoked by any other FU.

5.6.2.2.2. Invoked FUs

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

- *Restart Reconfigure FU*
- *Restart Reactivate FU*

5.6.2.2.3. Disrupting FUs

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

5.6.2.2.4. 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*
- *Periodic data Transfer FU*
- *Unsolicited Data Transfer FU*
- *Unsolicited Mixed 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 *Permanent Association FU* shall not be invoked for a given RESPONDER system if:

1. A *Permanent Association FU* invocation is running with the same RESPONDER system, with identical values of the parameters in the *A-Connect req.* primitive
- or:
2. A *Group Management FU* invocation or a *Group Definition FU* invocation is currently being running for the same RESPONDER User Element¹⁰.

5.6.2.3.3. 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 reestablishment loop if the association cannot be established immediately.

Mechanisms for breaking into the reestablishment loop from a higher-level function, for example with the purpose of negotiating down to an Elcom Version supported by both the INITIATOR and the RESPONDER, is considered outside the scope of this document.

Invocation of the RESPONDER part of the *Permanent Association FU* is triggered by reception of a valid *A-Connect ind.* primitive.

Note that the RESPONDER part of the *Permanent Association FU* necessarily must be identical to the RESPONDER part of the *Dynamic Association FU*¹¹, because no explicit information as to permanency is conveyed by the *A-Connect ind.* 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*.

5.6.2.4. Termination

5.6.2.4.1. Orderly termination

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

5.6.2.4.2. Disruption

¹⁰The 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 chapter 5.5 "Group configuration integrity control".

¹¹Consequently, 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.

A *Permanent Association FU* invocation may not be disrupted.¹²

5.6.3. Procedures

5.6.3.1. EASE service primitives

The following elementary EASE services are used by the *Permanent Association FU*:

- *A-Connect*
- *A-P-Abort*

5.6.3.1.1. Sequence

The normal sequence of primitives is partitioned into 2 phases:

Phase 1: Establishing the association.

INITIATOR UE	EASE	RESPONDER UE
<i>A-Connect req.</i> <i>A-Connect cnf.</i>		<i>A -Connect ind.</i> <i>A-Connect res.</i>

Phase 2: Abruptly terminating and reestablishing the association.

INITIATOR UE	EASE	RESPONDER UE
<i>A-P-Abort ind.</i> <i>A-Connect req.</i> <i>A-Connect cnf.</i>		<i>A-P-Abort ind.</i> <i>A -Connect ind.</i> <i>A-Connect res.</i>

¹²Any attempt to disrupt the *Association FU* on the part of the RESPONDER by abruptly terminating the association will result in the INITIATOR part eventually reestablishing the association.

Rules:

1. Phase 2 may occur at any time after phase 1 has been terminated, and may occur any number of times, including none at all.¹³
2. Whenever phase 2 occurs, the *A-P-Abort ind.* 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 or Periodic Data Transfer (for use by the *Unsolicited Data Transfer FU*, *Unsolicited Mixed Data Transfer FU*, or *Periodic 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* (or *Restart*, *group management lost*, but only if communicating in the Elcom-83 compatible mode).¹⁴
3. The time delay between reception of the *A-P-Abort ind.* primitive and the issuing of the *A-Connect req.* primitive in the INITIATOR UE is a local matter.
4. Whenever the restart code *Restart*, *spontaneous management lost* is received by the INITIATOR UE with the *A-Connect cnf.* primitive in phase 1, the *Permanent Association FU* shall try to invoke the *Restart Reactivate FU* for the RESPONDER in question.
5. Whenever the restart code *Restart*, *group management lost* is received by the INITIATOR UE with the *A-Connect cnf.* primitive in phase 1 while communicating in the Elcom-83 compatible mode, the *Permanent Association FU* shall try to invoke the *Restart Reconfigure FU* for the RESPONDER in question.
6. If configuration control field error¹⁵ is detected by the INITIATOR UE with the *A-Connect cnf.* primitive in phase 1, the *Permanent Association FU* shall try to invoke the *Restart Reconfigure FU* for the RESPONDER in question.
7. The rules stated in chapter 5.2 "Elcom class and version negotiation" also apply.
8. The rules stated in the procedure for Configuration Set check, in chapter 5.5 "Group configuration integrity control", also apply.

¹³In contrast to the *Dynamic Association FU*, any occurrence of the *A-P-Abort ind.* primitive is handled **within** the FU invocation, and thus does not constitute a *Permanent Association FU* disruption event or error.

¹⁴The RESPONDER UE may be programmed to **always** signal the restart code "*Restart*, *spontaneous management lost*" during creation of such associations.

¹⁵See chapter 5.5 "Group configuration integrity control".

5.6.3.1.2. Parameter values

A-Connect:

Parameter	req. (INITIATOR)	res. (RESPONDER)
Version	As specified in description of ACONRQ call in [8].	If Result >< <i>incompatible-versions</i> : Copy of value from <i>ind</i> . If Result = <i>incompatible-versions</i> : Value corresponding to version supported by RESPONDER.
Initiator	As specified in description of ACONRQ call in [8]. The "lower level part" shall be equal to the address of the INITIATOR's own node as specified in the section "Lower-level address part", above. The "A-suffix" part shall be as specified in the section "A-suffices", above.	Copy of value from <i>ind</i> .
Acceptor	As specified in description of ACONRQ call in [8]. The "lower level part" shall be equal to the address of the RESPONDER's own node as specified in chapter 5.1.1 "Lower-level address part". The "A-suffix" part shall be as specified in chapter 5.1.2 "A-suffices".	Copy of value from <i>ind.</i> , or any other value allowed by the EAPI.
User-data	Two octets, the second of which is the spontaneous mode code , as specified in chapter 5.3 "Spontaneous mode codes".	First octet (octet no. 0): restart code , as specified in chapter 5.4 "Restart codes", above. Octets no. 1 through 12: Control Field value. See the procedure for Configuration Set check, in chapter 5.5 "Group configuration integrity control", for coding rules.

cont.

Parameter	<i>req.</i> (INITIATOR)	<i>res.</i> (RESPONDER)
Result	<p>The security information field starts from octet 2 if present. The maximum length is 66 octets. The field is encoded as described in chapter A.9 "A-Connect request".</p> <p>(Not applicable)</p>	<p>The security information field starts from octet 13 if present. The maximum length is 66 octets. The field is encoded as described in chapter A.10 "A-Connect response".</p> <p>The security information field must be present if security classes 2 or 3 are used.</p> <p>It must be present if security class 1 with two-way authentication is selected, as well.</p> <p>Association established: = <i>result-ok</i></p> <p>Association not established, due to error condition: Other value. See chapter 5.6.3.2 "Error handling". (This means that the RESPONDER part of the <i>Association FU</i> has not been invoked, for the association).</p>

5.6.3.2. Error handling

5.6.3.2.1. FU disruption

The *Permanent Association FU* is never disrupted (see above).

5.6.3.2.2. 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 chapter 5.6.2.3 "Invocation", above. Consequently, the handling of this type of error is a local issue, outside the scope of this document.

In a RESPONDER UE, the *Association FU* must be present, for Elcom communications to be 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 document.

Attempted *Permanent Association FU* invocation violating rule 2 in chapter 5.6.2.3.2 "Restrictions", above (*Group Management FU* or *Group Definition FU* running): Start local retry timer, and enter normal retry loop upon expiry.

5.6.3.2.3. Incoming EASE service primitive out of context

INITIATOR part:

State	<i>A-Connect cnf.</i>
FU not yet running	Ignore, or local error indication
FU running, waiting for <i>A-Connect cnf.</i>	(Normal)
Running	Ignore, or local error indication/logging
Waiting for retry timer expiry	Ignore, or local error indication/logging

(The *A-P-Abort ind.* primitive is never considered out of context.)

RESPONDER part:

State	<i>A-Connect ind.</i>
Idle	(Normal)
Running	Issue normal <i>A-Connect cnf.</i>

(The *A-P-Abort ind.* 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 EASE	Specified action in FU
<p>UE too late responding to <i>A-Connect ind.</i></p>	<p>In RESPONDER: <i>A-P-Abort ind.</i>, with Reason = <i>misbehaviour-of-local-service-user</i>. Local error from eventual attempt at issuing <i>A-Connect res.</i></p> <p>In INITIATOR: <i>A-Connect cnf.</i>, with Result = <i>no-answer-from-remote-system</i>.</p>	<p>RESPONDER part: Go to idle state.</p> <p>INITIATOR part: See chapter 5.6.3.1 "EASE service primitive parameter errors", below.</p>

5.6.3.2.5. Congestion error

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

5.6.3.2.6. EASE service primitive parameter errors

Errors in the **A-Connect ind.** primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Version value indicating version not supported by the RESPONDER	Issue <i>A-Connect res.</i> primitive with Result = <i>incompatible-versions</i> . Enter idle state, if in other state.
Lower level part of Initiator (identifying the INITIATOR node) not valid, according to local access tables	Issue <i>A-Connect res.</i> primitive with Result = <i>authentication-failure</i> . Enter idle state, if in other state.
Less than 2 octets in User-data, or spontaneous mode code in User-data not defined in table in chapter 5.3 "Spontaneous mode codes", above	Treat as normal User-data, with spontaneous mode code for unsolicited mode. NOT an error; issue <i>A-Connect res.</i> with Result = <i>result-ok</i> , and enter running state as normal.
INITIATOR requested security class > 0, but the RESPONDER does not support any security mechanism.	Issue <i>A-Connect res.</i> primitive with Result = <i>security-is-not-supported-by-A-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>A-Connect res.</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>A-Connect res.</i> primitive with Result = <i>incompatible-security-options</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>A-Connect res.</i> primitive with Result = <i>incompatible-security-options</i> . Enter idle state, if in other state.
INITIATOR requested a security option (field authentication or encipherment) not supported by the RESPONDER.	Issue <i>A-Connect res.</i> primitive with Result = <i>incompatible-security-options</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>A-Connect res.</i> primitive with Result = <i>authentication-failure</i> Enter idle state, if in other state.

Errors in the **A-Connect *cnf.*** primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Version value >< Version value in corresponding <i>A-Connect req.</i> primitive, and Result = <i>result-ok</i> . (Version incompatibility undetected by the RESPONDER)	Trigger local abrupt termination of the association ¹⁶ .
Initiator >< Initiator in corresponding <i>A-Connect req.</i> primitive, and Result = <i>result-ok</i> . (INITIATOR FU identity mix-up in RESPONDER)	Report the error back to the Coordinating Function, and trigger local abrupt termination of the association.
Zero octets in User-data, or restart code in User-data not defined in table in chapter 5.4 "Restart codes", above, and Result = <i>result-ok</i> .	Treat as normal User-data, with restart code <i>Restart</i> , group management lost. NOT an error.
Security returned, but not requested.	Trigger local abrupt termination of the association.
INITIATOR expects contents in Authentication information field, but this field is empty.	Trigger local abrupt termination of the association.
Result >< <i>result-ok</i>	Start local retry timer, and enter normal retry loop upon expiry ¹⁷ .

¹⁶Such abrupt termination is effected by a local "detach" call (ADET).

¹⁷ 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.

5.7. Dynamic Association FU

Type: Primary.

5.7.1. Function

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

1. The INITIATOR UE establishes an association with a RESPONDER UE.
2. 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.
3. 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.
4. During orderly termination of the *Dynamic Association FU* invocation, the INITIATOR UE releases the association.

5.7.2. Coordination rules

5.7.2.1. Association usage

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

5.7.2.2. Relation to other FUs

5.7.2.2.1. Invoking FUs

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

5.7.2.2.2. Invoked FUs

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

- *Restart Reconfigure FU*

5.7.2.2.3. 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*

5.7.2.2.4. 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*

5.7.2.3. Invocation

5.7.2.3.1. Prerequisites

No previous or concurrent FU invocations are required.

5.7.2.3.2. Restrictions

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

1. A *Dynamic Association FU invocation* is already running with the same RESPONDER system, with identical values of the parameters in the *A-Connect req.* primitive,

or:

2. A *Group Management FU invocation* or a *Group Definition FU invocation* is running for the same RESPONDER¹⁸.

5.7.2.3.3. 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 document.
- 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 *A-Connect ind.* primitive.

As explained in the corresponding chapter 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*.

¹⁸The 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*.

5.7.2.4. Termination

5.7.2.4.1. 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 document.¹⁹
- 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-Release ind.* primitive.

5.7.2.4.2. 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 chapter 5.7.2.2.3 "Disrupting FUs", above.
- The *Dynamic Association FU* invocation itself, upon fatal error. See chapter 5.7.3.2 "Error handling", below.

5.7.3. Procedures

5.7.3.1. EASE service primitives

The following elementary EASE services are used by the *Dynamic Association FU*:

- *A-Connect*
- *A-P-Abort*
- *A-Release*

¹⁹Consequently, this document 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.

5.7.3.1.1. Sequence

The normal sequence of primitives is partitioned into 2 phases:

Phase 1: Establishing the association.

INITIATOR UE	EASE	RESPONDER UE
<i>A-Connect req.</i> <i>A-Connect cnf.</i>		<i>A -Connect ind.</i> <i>A-Connect res.</i>

Phase 2: Orderly termination: Releasing the association.

INITIATOR UE	EASE	RESPONDER UE
<i>A-Release req.</i> <i>A-Release cnf</i>		<i>A-Release ind.</i> <i>A-Release res.</i>

Rules:

- Any *A-P-Abort ind.* 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* invocation.²⁰
 - Disruption of all other FUs that are running on the association.
- Any *A-P-Abort ind.* primitive received by the RESPONDER UE on the association, subsequent to the phase 1 sequence above, shall trigger, locally:
 - Disruption of the *Association FU* invocation.²¹
 - Disruption of all other FUs that are running on the association.
- 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 document.²²

²⁰Unlike the *Permanent Association FU*, the *Dynamic Association FU* considers the *A-P-Abort ind.* primitive an error event.

²¹This 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.

²²Three possible strategies:

- Enter phase 2 immediately, when there is no more traffic.
- Enter phase 2 after a fixed delay time after the occurrence of no more traffic.
- Enter phase 3 only after operator action.

4. Whenever the restart code *Restart, spontaneous management lost*, is received by the INITIATOR UE with the *A-Connect cnf.* primitive in phase 1, the INITIATOR shall ignore this restart code for the Dynamic Association FU (the code is valid only for Periodic and Unsolicited data transfer).
5. When communicating in the Elcom-83 compatible mode, the RESPONDER UE shall during phase 1 signal restart code *Restart, group management lost*, whenever local conditions indicate that a complete reconfiguration of all groups for the INITIATOR system is necessary. Typically: After RESPONDER system restart.
6. Whenever the restart code *Restart, group management lost*, is received by the INITIATOR UE with the *A-Connect cnf.* primitive in phase 1 while communicating in the Elcom-83 compatible mode, the *Dynamic Association FU* shall try to invoke the *Restart Reconfigure FU* for the RESPONDER in question.
7. If configuration control field error²³ is detected by the INITIATOR UE with the *A-Connect cnf.* primitive in phase 1, the *Dynamic Association FU* shall try to invoke the *Restart Reconfigure FU* for the RESPONDER in question.
8. The rules stated in chapter 5.2 "Elcom class and version negotiation" also apply.
9. The rules stated in the procedure for Configuration Set check, in chapter 5.5 "Group configuration integrity control", also apply.

5.7.3.1.2. Spontaneous mode codes

Both periodic and unsolicited data transfer (*Unsolicited Data Transfer FU*, *Unsolicited Mixed Data Transfer FU* or *Periodic data Transfer FU*) shall occur only over permanent associations, that is only over associations controlled by invocations of the *Permanent Association FU*. Spontaneous mode codes are consequently insignificant in *Dynamic Association FU* primitives.

²³See chapter 5.5 "Group configuration integrity control".

5.7.3.1.3. Parameter values

A-Connect:

Parameter	req. (INITIATOR)	res. (RESPONDER)
Version	As specified in description of ACONRQ call in [8].	If Result >< <i>incompatible-versions</i> : Copy of value from <i>ind</i> . If Result = <i>incompatible-versions</i> : Value corresponding to version supported by RESPONDER.
Initiator	As specified in description of ACONRQ call in [8]. The "lower level part" shall be equal to the address of the INITIATOR's own node as specified in chapter 5.1.1 "Lower-level address part", above. The "A-suffix" part shall be as specified in chapter 5.1 "Addressing", above.	Copy of value from <i>ind</i> .
Acceptor	As specified in description of ACONRQ call in [8]. The "lower level part" shall be equal to the address of the RESPONDER's own node as specified in chapter 5.1.1 "Lower-level address part", above. The "A-suffix" part shall be as specified in chapter 5.1 "Addressing", above.	Copy of value from <i>ind</i> ., or any other value allowed by the EAPI.

cont.

Parameter	req. (INITIATOR)	res. (RESPONDER)
User-data	Any value allowed by the EAPI. ²⁴ The security information field starts from octet 2 if present. The maximum length is 66 octets. The field is encoded as described in chapter A.9 "A-Connect request".	First octet (octet no. 0): restart code , as specified in chapter 5.4 "Restart codes", above. Octets no. 1 through 12: Control Field value. See the procedure for Configuration Set check, in chapter 5.5 "Group configuration integrity control", for coding rules. The security information field starts from octet 13 if present. The maximum length is 66 octets. The field is encoded as described in chapter A.10 "A-Connect response". 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 chapter 5.7.3.2 "Error handling". (This means that the RESPONDER part of the <i>Association FU</i> has not been invoked, for the association).

A-Release:

Parameter	req. (INITIATOR)	res. (RESPONDER)
User-reason	As specified in description of ARELRQ call in [8]; choice outside scope of this document.	(Not applicable)
Result	(Not applicable)	Always equal to <i>result-ok</i>

²⁴Unsolicited and periodic data transfers never occur via associations set up by the *Dynamic Association FU*; only by the *Permanent Association FU*.

5.7.3.2. Error handling

5.7.3.2.1. FU disruption

Disruption by another FU is only done indirectly:

The disrupting FU invocation shall trigger abrupt disruption of the *Dynamic Association FU* invocation controlling the association over which the disrupting FU invocation operates, by local EAPI "detach" calls. This will result in *A-P-Abort ind.* primitives arriving at the other end of the association, abruptly terminating the *Dynamic Association FU* invocation.

Likewise, whenever the *Dynamic Association FU* shall trigger abrupt termination of itself, this shall be done by local EAPI "detach" calls.

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 document.

5.7.3.2.2. 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 document.

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, for Elcom communications 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 document.

Attempted *Dynamic Association FU* invocation attempt violating rule 2 in chapter 5.7.2.3.2 "Restrictions", above (*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*.

5.7.3.2.3. Incoming EASE service primitive out of context

INITIATOR part:

State	<i>A-Connect cnf.</i>	<i>A-Release cnf.</i>
FU not running	Ignore, or local error indication	Ignore, or local error indication
FU running, waiting for <i>A-Connect cnf.</i>	(Normal)	Trigger local abrupt termination of the association ²⁵ , then go to idle state.
Running	Ignore, or local error indication/logging	Trigger local abrupt termination of the association, then go to idle state.
Waiting for <i>A-Release cnf.</i>	Trigger local abrupt termination of the association, then go to idle state.	(Normal)

(The *A-P-Abort ind.* primitive is never considered out of context.)

RESPONDER part:

State	<i>A-Connect ind.</i>	<i>A-Release ind.</i>
Idle	(Normal)	Issue normal <i>A-Release cnf.</i>
Running	Issue normal <i>A-Connect cnf.</i>	(Normal)

(The *A-P-Abort ind.* primitive is never considered out of context.)

²⁵Such abrupt termination is effected by a local "detach" call (ADET).

5.7.3.2.4. Timing errors

There can occur no timing errors in the INITIATOR part.

Error in RESPONDER part:

Error	Reaction from EASE	Specified action in FU
<p>UE too late responding to <i>A-Connect ind.</i></p>	<p>In RESPONDER: <i>A-P-Abort ind.</i>, with Reason = <i>misbehaviour-of-local-service-user</i>. Local error from eventual attempt at issuing <i>A-Connect res.</i></p> <p>In INITIATOR: <i>A-Connect cnf.</i>, with Result = <i>no-answer-from-remote-system</i>.</p>	<p>RESPONDER part: Go to idle state.</p> <p>INITIATOR part: See chapter 5.7.3.1 "EASE service primitive parameter errors", below.</p>
<p>UE too late responding to <i>A-Release ind.</i></p>	<p>In RESPONDER: <i>A-P-Abort ind.</i>, with Reason = <i>misbehaviour-of-local-service-user</i>. Local error from any attempt at issuing <i>A-Release res.</i></p> <p>In INITIATOR: <i>A-P-Abort ind.</i>, with Reason = <i>no-answer-from-remote-system</i>.</p>	<p>RESPONDER part: Handle as abrupt termination of association.</p> <p>INITIATOR part: Handle as abrupt termination of association.</p>

5.7.3.2.5. Congestion error

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

5.7.3.2.6. EASE service primitive parameter errors

Errors in the **A-Connect ind.** primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Version value indicating version not supported by the RESPONDER	Issue <i>A-Connect res.</i> primitive with Result = <i>incompatible-versions</i> . Enter idle state, if in other state.
Lower level part of Initiator (identifying the INITIATOR node) not valid, according to local access tables	Issue <i>A-Connect res.</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>A-Connect res.</i> primitive with Result = <i>security-is-not-supported-by-A-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>A-Connect res.</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>A-Connect res.</i> primitive with Result = <i>incompatible-security-options</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>A-Connect res.</i> primitive with Result = <i>incompatible-security-options</i> . Enter idle state, if in other state.
INITIATOR requested a security option (field authentication or encipherment) not supported by the RESPONDER.	Issue <i>A-Connect res.</i> primitive with Result = <i>incompatible-security-options</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>A-Connect res.</i> primitive with Result = <i>authentication-failure</i> . Enter idle state, if in other state.
<p>Note: User-data is always valid within the <i>Dynamic Association FU</i>, provided it is valid within the EAPI.</p>	

Errors in the **A-Release ind.** primitive (detected by the RESPONDER part):

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

Errors in the **A-Connect *cnf.*** primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Version >< Version in corresponding <i>A-Connect req.</i> primitive, and Result = <i>result-ok.</i> (Version incompatibility undetected by the RESPONDER)	Trigger local abrupt termination of the association ²⁶ .
Initiator >< Initiator in corresponding <i>A-Connect req.</i> primitive, and Result = <i>result-ok.</i> ²⁷ (INITIATOR FU identity mix-up in RESPONDER.)	Report the error back to the Coordinating Function, and trigger local abrupt termination of the association.
Zero octets in User-data, or restart code in User-data not defined in table in chapter 5.4 "Restart codes", above, and Result = <i>result-ok.</i>	Treat as normal User-data, with restart code <i>Restart, group management lost.</i> NOT an error.
Security returned, but not requested.	Trigger local abrupt termination of the association ²⁶ .
INITIATOR expects contents in Authentication information field, but this field is empty.	Trigger local abrupt termination of the association ²⁶ .
Result >< <i>result-ok</i>	Report the error back to the local Coordinating Function, and enter the idle state.

Errors in the **A-Release *cnf.*** primitive (detected by the INITIATOR part):

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

²⁶Such abrupt termination is effected by a local "detach" call (ADET).

²⁷No 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.

5.8. Test Association FU

Type: Primary.

5.8.1. Function

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

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

5.8.2. Coordination rules

5.8.2.1. Association usage

All Elcom 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 chapter 5.8.2.3.1 "Prerequisites", below.

5.8.2.2. Relation to other FUs

5.8.2.2.1. Invoking FUs

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

5.8.2.2.2. Invoked FUs

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

5.8.2.2.3. 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 *A-P-Abort ind.* primitive for the association concerned.

5.8.2.2.4. Disrupted FUs

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

5.8.2.3. Invocation

5.8.2.3.1. 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 chapter 5.1.2 "A-suffices".

5.8.2.3.2. Restrictions

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

5.8.2.3.3. 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 document²⁸.

Invocation of the RESPONDER part of the *Test Association FU* is attempted whenever an *A-Test-Connection ind.* primitive is received via an association, regardless of the characteristics of the association.

5.8.2.4. Termination

5.8.2.4.1. Orderly termination

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

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

Such termination of the *Test Association FU* may also be triggered by congestion error. See relevant chapter, below.

5.8.2.4.2. Disruption

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

- Disruption of another FU invocation. See chapter 5.8.2.2.3 "Disrupting FUs", above.

²⁸Such invocations typically may be timer-driven or operator-triggered.

5.8.3. Procedures

5.8.3.1. EASE service primitives

The following elementary EASE service is used by the *Test Association FU*:

- *A-Test-Connection*

5.8.3.1.1. Sequence

INITIATOR UE	EASE	RESPONDER UE
<i>A-Test-Connection req.</i> <i>A-Test-Connection cnf.</i>		<i>A-TestConnection ind.</i> <i>A-Test-Connection res.</i>

5.8.3.1.2. Parameter values

A-Test-Connection:

Parameter	<i>req. (INITIATOR)</i>	<i>res. (RESPONDER)</i>
Result	(Not applicable)	Always equal to value <i>result-ok</i>

5.8.3.2. Error handling

5.8.3.2.1. 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 itself.²⁹

5.8.3.2.2. 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 chapter 5.8.2.3 "Invocation", above. Consequently, the handling of this type of error is a local issue, outside the scope of this document.

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

The RESPONDER User Element shall ignore the incoming *A-Test-Connection ind.* 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 document.

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

²⁹Local clean-up procedures are not specified by this document.

5.8.3.2.3. Incoming EASE service primitive out of context

INITIATOR part:

State	<i>A-Test-Connection cnf.</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 *A-Test-Connection ind.* primitive is never considered out of context.

5.8.3.2.4. Timing errors

There can occur no timing errors in the INITIATOR part.

Error in RESPONDER part:

Error	Reaction from EASE	Specified action in FU
<p>UE too late responding to <i>A-Test-Connection ind.</i></p>	<p>In RESPONDER: <i>A-P-Abort ind.</i>, with Reason = <i>misbehaviour-of-local-service-user</i>. Local error from eventual attempt at issuing <i>A-Test-Connection res.</i></p> <p>In INITIATOR: <i>A-Test-Connection cnf.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Ignore, or local error indication</p> <p>INITIATOR part: See chapter 5.8.3.2.6 "EASE service primitive parameter errors", below.</p>

5.8.3.2.5. Congestion error

In INITIATOR part of the FU, occurring with an *A-Test-Connection req.* attempt:
Terminate FU invocation locally.

In RESPONDER part of the FU, occurring with an *A-Test-Connection res.* attempt:
Terminate FU invocation locally, as normal except for local error report if desired.
(INITIATOR part of FU will be terminated after timeout.)

5.8.3.2.6. EASE service primitive parameter errors

Errors in the ***A-Test-Connection cnf.*** primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Result >< <i>result-ok</i>	No special action other than registering "Result not OK".

Section 6 : THE DATA IDENTIFICATION FUNCTION GROUP

TABLE OF CONTENTS

6. THE DATA IDENTIFICATION FUNCTION GROUP	6-1
6.1. Group attributes	6-1
6.1.1. Group type	6-3
6.1.2. Group number	6-4
6.1.3. Group size	6-4
6.1.4. Object identifier size	6-5
6.1.5. Persistent	6-5
6.1.6. Static	6-6
6.1.7. Priority class	6-7
6.1.8. Object set	6-7
6.1.8.1. Object number	6-8
6.1.8.2. Object identifier	6-8
6.2. Group Management FU	6-9
6.2.1. Function	6-9
6.2.2. Coordination rules	6-9
6.2.2.1. Association usage	6-9
6.2.2.2. Relation to other FUs	6-10
6.2.2.2.1. Invoking FUs	6-10
6.2.2.2.2. Invoked FUs	6-10
6.2.2.2.3. Disrupting FUs	6-10
6.2.2.2.4. Disrupted FUs	6-10
6.2.2.3. Invocation	6-11
6.2.2.3.1. Prerequisites	6-11
6.2.2.3.2. Restrictions	6-11
6.2.2.3.3. Invoking events	6-12
6.2.2.4. Termination	6-12
6.2.2.4.1. Orderly termination	6-12
6.2.2.4.2. Disruption	6-12
6.2.3. Procedures	6-12
6.2.3.1. EASE service primitives	6-12
6.2.3.1.1. Sequence	6-13
6.2.3.1.2. Parameter values	6-14
6.2.3.2. Error handling	6-17
6.2.3.2.1. FU disruption	6-17
6.2.3.2.2. Illegal invocation attempt	6-17
6.2.3.2.3. Incoming EASE service primitive out of context	6-18
6.2.3.2.4. Timing errors	6-19
6.2.3.2.5. Congestion error	6-19
6.2.3.2.6. EASE service primitive parameter errors	6-20
6.3. Group Definition FU	6-23
6.3.1. Function	6-23
6.3.2. Coordination rules	6-24
6.3.2.1. Association usage	6-24
6.3.2.2. Relation to other FUs	6-24
6.3.2.2.1. Invoking FUs	6-24
6.3.2.2.2. Invoked FUs	6-24

6.3.2.2.3. Disrupting FUs	6-24
6.3.2.2.4. Disrupted FUs	6-24
6.3.2.3. Invocation	6-25
6.3.2.3.1. Prerequisites	6-25
6.3.2.3.2. Restrictions	6-25
6.3.2.3.3. Invoking events	6-26
6.3.2.4. Termination	6-26
6.3.2.4.1. Orderly termination	6-26
6.3.2.4.2. Disruption	6-26
6.3.3. Procedures	6-27
6.3.3.1. EASE service primitives	6-27
6.3.3.1.1. Sequence	6-27
6.3.3.1.2. Parameter values	6-30
6.3.3.2. Error handling	6-33
6.3.3.2.1. FU disruption	6-33
6.3.3.2.2. Illegal invocation attempt	6-33
6.3.3.2.3. Incoming EASE service primitive out of context	6-34
6.3.3.2.4. Timing errors	6-35
6.3.3.2.5. Congestion error	6-35
6.3.3.2.6. EASE service primitive parameter errors	6-36
6.4. Group Readout FU	6-39
6.4.1. Function	6-39
6.4.2. Coordination rules	6-39
6.4.2.1. Association usage	6-39
6.4.2.2. Relation to other FUs	6-40
6.4.2.2.1. Invoking FUs	6-40
6.4.2.2.2. Invoked FUs	6-40
6.4.2.2.3. Disrupting FUs	6-40
6.4.2.2.4. Disrupted FUs	6-40
6.4.2.3. Invocation	6-41
6.4.2.3.1. Prerequisites	6-41
6.4.2.3.2. Restrictions	6-41
6.4.2.3.3. Invoking events	6-41
6.4.2.4. Termination	6-42
6.4.2.4.1. Orderly termination	6-42
6.4.2.4.2. Disruption	6-42
6.4.3. Procedures	6-43
6.4.3.1. EASE service primitives	6-43
6.4.3.1.1. Sequence	6-43
6.4.3.1.2. Parameter values	6-47
6.4.3.2. Error handling	6-51
6.4.3.2.1. FU disruption	6-51
6.4.3.2.2. Illegal invocation attempt	6-51
6.4.3.2.3. Incoming EASE service primitive out of context	6-52
6.4.3.2.4. Timing errors	6-52
6.4.3.2.5. Congestion error	6-53
6.4.3.2.6. EASE service primitive parameter errors	6-53
6.5. Group Configuration FU	6-55
6.5.1. Function	6-55
6.5.2. Coordination rules	6-55
6.5.2.1. Association usage	6-55
6.5.2.2. Relation to other FUs	6-56
6.5.2.2.1. Invoking FUs	6-56
6.5.2.2.2. Invoked FUs	6-56
6.5.2.2.3. Disrupting FUs	6-56
6.5.2.2.4. Disrupted FUs	6-56
6.5.2.3. Invocation	6-57

6.5.2.3.1. Prerequisites	6-57
6.5.2.3.2. Restrictions	6-57
6.5.2.3.3. Invoking events	6-57
6.5.2.4. Termination	6-58
6.5.2.4.1. Orderly termination	6-58
6.5.2.4.2. Disruption	6-58
6.5.3. Procedures	6-58
6.5.3.1. EASE service primitives	6-58
6.5.3.1.1. Action sequence	6-59
6.5.3.1.2. Parameter values	6-59
6.5.3.2. Error handling	6-60
6.5.3.2.1. FU disruption	6-60
6.5.3.2.2. Illegal invocation attempt	6-60

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 1¹ - 255 objects; see chapter 6.1.3 "Group size", below.

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

1. Via the Elcom interface, as specified by this document.
2. By any means not involving the Elcom interface, not specified by this document.

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

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

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

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

- **Object number**
- **Object identifier**

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 class**

¹A group containing 0 objects is created but not defined, and is not considered valid here.

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

- **Object set** (some or all objects)

Only the following² 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 class**

²[9], chapter 12.3.1.1.

6.1.1. Group type

The table below lists the group types that are standardised by the Elcom-90 specification³. 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>Measure-group</i>	Floating-point values	DRFU, DPRFU, DPFU, DUFU, DUMFU
<i>Status-group</i>	3-state values (on/off/between)	DRFU, DPRFU, DPFU, DUFU, DUMFU
<i>Discrete-group</i>	Two octet integer	DRFU, DPRFU, DPFU, DUFU, DUMFU
<i>Logical-breaker-status-group</i>	Composite 3-state values	DRFU, DPRFU, DPFU, DUFU, DUMFU
<i>Binary-command-group</i>	2-state commands (on/off)	DSFU
<i>Analog-setpoint-group</i>	Floating-point command values	DSFU
<i>Digital-setpoint-group</i>	Two octet integer command values	DSFU
<i>Text-message-group</i>	General ASCII character string	DRFU, DPRFU, DPFU, DUFU

Further information may be found in the individual FU descriptions, in Appendix A, and in relevant EASE documentation ([8] and [9]).

³Additional group types, defined for regional conventions are allowed.

6.1.2. Group number

Each group has an identifying associated integer, for use at the EAPI interface⁴. 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 allowed.⁵

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

The RESPONDER UE may accept or reject a proposed number, according to local conditions⁶. 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 an UE is not restricted by others 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 document. As a minimum, any RESPONDER UE shall support the following value of **Group size**:
20.

⁴For local use, Elcom groups may well be identified by other means. However, at the EAPI interface, the identifying quantity is always this group number.

⁵In the M-D-REQ PDU, for example, group number 0 terminates the data field.

⁶For 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 document.

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 document. As a minimum, any RESPONDER UE shall support the following value of **Object identifier size**: **12**.

6.1.5. Persistent

A boolean value, with the following semantics:

- true*: The group can be neither deleted/changed nor redefined via the Elcom interface: Any invocation of the *Group Management FU* or the *Group Definition FU* shall fail for such groups.⁷ Such groups are termed **permanent**, and if also of non-Elcom origin, **predefined**.
- false*: The group configuration may be manipulated at will via the Elcom interface. Such groups are termed **non-permanent**, or **dynamic**.

Note that if it is desirable to exclude a group configuration of non-Elcom origin from any manipulative access via the Elcom 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 indeed 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-Elcom means, if necessary.

The procedure for creating a persistent group is:

1. Create group with **Persistent**=false
2. Define group (Multiple ADGRQ's can follow)
3. Change the attribute **Persistent** to true

⁷Rules for manipulation by non-Elcom means are not within the scope of this document.

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.⁸
- false*: The object set of the group may be defined or redefined at will, with *Group Definition FU* invocations for the group.

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

The procedure for creating a static group with multiple ADGRQ's is:

1. Create group with **Static** = false.
2. Define group (Multiple ADGRQ's may follow).
3. Change the attribute **Static** to true.

⁸The purpose of this exception is to enable reconfiguration (invocation of *Restart Reconfigure FU*) of groups for which **Static** = *true*. This will allow the Initiator to send 1 ADGRQ, and this puts a limitation on the size of the static group.

6.1.7. Priority class

The **Priority class** is an integer value within the range 0 - 15, governing:

1. For Unsolicited Data Transfer:

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

2. For Periodic Data Transfer:

The period lengths in the RESPONDER UE for the individual *Periodic Data Transfer FU* invocations, in a somewhat indirect manner. See *Periodic Data Transfer FU* description.

Whether **Priority class** 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 class 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 mechanism.⁹ RESPONDER UEs that support the mechanism, shall accept all **Priority class** values within the legal range¹⁰. 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 document.

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 in the CS(R) into accordance with the CS(I) in the INITIATOR system.

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

⁹Support of this mechanism shall mean support of both output queue ordering and period length determination.

¹⁰However, the granularity of actual decisions may be less than 1/15: Local subranges with equal effective value of **Priority class** may be defined, but at least two effective values shall be the result.

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 *A-Def-Group ind.* service primitive; see the *Group Definition FU* description. 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 Elcom 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 identify¹¹ 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 document.

Object identifiers are copied by the RESPONDER UE from the Objid parameter of the *A-Def-Group ind.* service primitive into the CS(R); see the *Group Definition FU* description.

¹¹Within a given system or site, objects may be identified by other mechanisms, with local reference to the corresponding Elcom **Object identifier**.

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 that was created via the Elcom interface
<i>G-Delete</i>	Deletes a single group that was created via the Elcom interface: Deallocates the corresponding set of attributes.
<i>Delete-all-groups</i>	Deletes ALL groups: Deallocates the set of attributes for all groups in the CS(R) that were created via the Elcom interface.

6.2.2. Coordination rules

6.2.2.1. Association usage

All Elcom 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 chapter 6.2.2.3.1 "Prerequisites", below.

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*
- *Periodic Data Transfer FU*
- *Unsolicited Data Transfer FU*
- *Unsolicited Mixed Data Transfer FU*
- *Supervisory Control Data Transfer FU*

6.2.2.3. Invocation

6.2.2.3.1. Prerequisites

One¹² of the following FUs shall have been invoked preceding any invocation of the *Group Management FU*:

- *Permanent Association FU*
- *Dynamic Association FU*

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 chapter 5.1.2 "A-suffices".

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* must not be invoked while data transfer is active, for the group involved.

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

A *Group Management FU* must 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 Elcom partner.¹³

¹²Subject to local decision in the INITIATOR UE.

¹³The purpose of this rule is to ensure the integrity of the configuration integrity control parameter for the INITIATOR system in the RESPONDER system. This parameter is updated by both the *Group Management FU* and the *Group Definition FU*; see chapter 5.5 "Group configuration integrity control".

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 document.
- The *Group Configuration FU*.

Invocation of the RESPONDER part of the *Group Management FU* is attempted whenever a valid *A-Group-Mgmt ind.* primitive is received via an association with the characteristics as defined for the *Group Management FU*: See description of the *Permanent Association 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 relevant chapter, below.
- Reception of an *A-Group-Mgmt cnf.* service primitive after issuing an *A-Group-Mgmt req.* service primitive (normal termination).

The RESPONDER part of a *Group Management FU* invocation always terminates itself in an orderly manner (normal termination) after issuing an *A-Group-Mgmt res.* 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 chapter 6.2.2.2.3 "Disrupting FUs", above.
- A fatal error condition encountered during operation. See chapter 6.2.3.2 "Error handling", below.

6.2.3. Procedures

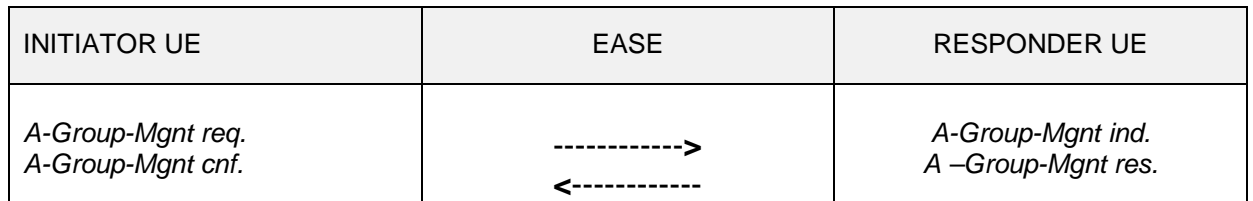
6.2.3.1. EASE service primitives

The following elementary EASE service is used by the *Group Management FU*:

- *A-Group-Mgmt*

6.2.3.1.1. Sequence

The normal sequence of primitives is as follows:



Rules:

1. Upon reception of a valid *A-Group-Mgmt ind.* service primitive, the RESPONDER UE shall:
 - Update group attributes in the CS(R) with information from the primitive:

Attribute	Parameter	Function
Group Type Group Number	Gtype Gnr	<i>G-Create</i> <i>G-Create, G-Delete</i> <i>Delete-all-groups</i>
Persistent	Persist	<i>G-Create, G-Change¹⁴</i>
Static	Static	<i>G-Create, G-Change</i>
Priority class	Priority Class	<i>G-Create, G-Change</i>
Group size	Gsize	<i>G-Create</i>
Object identifier size	Objlength	<i>G-Create</i>

- 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 system.¹⁵
 - Report the new value of the configuration integrity control parameter back to the INITIATOR UE, via the CF parameter in the *A-Group-Mgmt res.* service primitive.
2. The RESPONDER UE shall return "OK" status (parameter Result = *result-ok*) if and only if the configuration integrity control parameter has been updated OK.
 3. Upon reception of a valid *A-Group-Mgmt cnf.* 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 system.¹⁶
 4. The rules stated in chapter 6.1 "Group attributes", above, also apply here.
 5. The rules stated in the procedure for Configuration Set update, in chapter 5.5 "Group configuration integrity control", also apply.

¹⁴The attribute 'Persistent' is not allowed to change from true to false, only from false to true.

¹⁵See the APDU description, about the configuration integrity control parameter and associated functions.

¹⁶The INITIATOR UE will later check his value of the configuration integrity control parameter against the value conveyed by the *A-Connect cnf.* 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 section 5 "THE ASSOCIATION MANAGEMENT FUNCTION GROUP".

6.2.3.1.2. Parameter values

A-Group-Mgmt :

Parameter	req. (INITIATOR)	res. (RESPONDER)
Function	<i>G-Create, G-Change, G-Delete or Delete-all-groups, according to the function with which the FU is invoked</i>	Copy of value from <i>ind.</i>
Gtype	<p>Value <i>Measure-group, Status-group, Discrete-group, Logical-breaker status-group, Binary-command-group, Analog-setpoint-group, Digital-setpoint-group or Text-message-group.</i></p> <p>Function = <i>G-Create</i>: Value of attribute Group type to be included in the CS(R) for group no. Gnr.</p> <p>Function = <i>G-Change</i>: Value shall be equal to value of attribute Group type in the CS(R) for group no. Gnr., for the request to be legal.</p> <p>Function = <i>G-Delete or Delete-all-groups</i>: Value is irrelevant. Do not test for illegal values.</p>	Copy of value from <i>ind.</i>
Gnr	<p>Reference number for the group in question.</p> <p>Function = <i>G-Create</i>: Value of attribute Group number in set of attributes to be created in the CS(R).</p> <p>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</p> <p>Function = <i>G-Delete</i>: Value of attribute Group number in set of attributes to be removed from the CS(R).</p> <p>Function = <i>Delete-all-groups</i>: Value is irrelevant. Do not test for illegal values.</p>	Copy of value from <i>ind.</i>

Cont.

Parameter	reg. (INITIATOR)	res. (RESPONDER)
CF	(Not applicable)	Control Field value for the INITIATOR UE that invoked the <i>Group Management FU</i> . See chapter 5.5 "Group configuration integrity control", in section "THE ASSOCIATION MANAGEMENT FUNCTION GROUP".
Gsize	<p>Group size.</p> <p>Function <i>G-Create</i>: Value of attribute Group size to be included in the CS(R) for group no. Gnr.</p> <p>Function = <i>G-Change</i>: Value shall be equal to value of attribute Group size in the CS(R) for group no. Gnr., for the request to be legal.</p> <p>Function = <i>G-Delete</i> or <i>Delete-all-groups</i>: Value is irrelevant. Do not test for illegal values.</p>	(Not applicable)
Objlength	<p>Object identifier size.</p> <p>Function <i>G-Create</i>: Value of attribute Object identifier size to be included in the CS(R) for group no. Gnr.</p> <p>Function = <i>G-Change</i>: Value shall be equal to value of attribute Object identifier size in the CS(R) for group no. Gnr., for the request to be legal.</p> <p>Function = <i>G-Delete</i> or <i>Delete-all-groups</i>: Value is irrelevant. Do not test for illegal values.</p>	(Not applicable)
Persist	<p>Persistent flag.</p> <p>Function = <i>G-Create</i> or <i>G-Change</i>: Value of attribute Persistent to be included in the CS(R) for group no. Gnr.</p> <p>Function = <i>G-Delete</i> or <i>Delete-all-groups</i>: Value is irrelevant. Do not test for illegal values.</p>	(Not applicable)

cont.

Parameter	req. (INITIATOR)	res. (RESPONDER)
Static	Static flag. Function = <i>G-Create</i> or <i>G-Change</i> : Value of attribute Static to be included in the CS(R) for group no. Gnr. Function = <i>G-Delete</i> or <i>Delete-all-groups</i> : Value is irrelevant. Do not test for illegal values.	(Not applicable)
Priority Class	Priority class value. Function = <i>G-Create</i> or <i>G-Change</i> : Value of attribute Priority class to be included in the CS(R) for group no. Gnr. Function = <i>G-Delete</i> or <i>Delete-all-groups</i> : Value is irrelevant. Do not test for illegal values.	(Not applicable)
Result	(Not applicable)	Action performed as specified. = <i>result-ok</i> Action not performed, due to error condition: Other value. See chapter 6.2.3.2 "Error handling".
Note: With Function = <i>G-Change</i> , the RESPONDER UE shall be able to handle modification of one or more of the attributes Persistent , Static and Priority class during a single <i>A-Group-Mgmt</i> transaction.		

6.2.3.2. Error handling

6.2.3.2.1. FU disruption

Disruption by the *Permanent Association FU* or *Dynamic Association FU*:

Disruption of both the INITIATOR part and the RESPONDER part of the current invocation of the *Group Management FU* shall be triggered locally, as a part of the handling of incoming *A-R-Abort ind.* primitives in both the INITIATOR part and the RESPONDER part of the *Permanent Association FU* or *Dynamic Association FU* invocation handling the association where the *Group Management FU* invocation is running.

Both parts of the current invocation of the *Group Management FU* shall be terminated gracefully, neither part attempting to issue any primitive associated with the termination itself.¹⁷

6.2.3.2.2. Illegal invocation attempt

FU not present:

If the *Group Management FU* is not present in an INITIATOR UE:

The handling of this type of error resulting from an invocation attempt by the local Coordinating Function is a local issue, outside the scope of this document.

If this type of error occurs upon an invocation attempt by a *Group Configuration FU* invocation, the error is handled by the error handling mechanism within the *Group Configuration FU*. See the *Group Configuration FU* description.

If the *Group Management FU* is not present in an RESPONDER UE:

The RESPONDER UE shall respond to activation attempts in one of two ways:

Either:

- Ignoring the incoming *A-Group-Mgmt ind.* primitive altogether

or:

- Issuing an *A-Group-Mgmt res.* primitive with Result = *remote-service-user-unavailable*.

¹⁷Local clean-up procedures are not specified by this document.

FU present, but attempt illegal:

Invocation attempts violating the rules stated in chapter 6.2.2.3.2 "Restrictions" above in an INITIATOR UE is a local issue, outside the scope of this document.

In a RESPONDER UE, invocation attempts violating the same rules shall be handled by the RESPONDER UE in one of two ways:

Either:

- Ignoring the incoming *A-Group-Mgmt ind.* primitive altogether

or:

- Issuing an *A-Group-Mgmt res.* primitive with Result = *remote-service-user-unavailable* without actually (re-)invoking the *Group Management FU* for the group concerned, in the RESPONDER UE.

6.2.3.2.3. Incoming EASE service primitive out of context

INITIATOR part:

State	<i>A-Group-Mgmt cnf.</i>
FU not running	Ignore, or local error indication
FU running, waiting for <i>A-Group-Mgmt cnf.</i>	(Normal)

RESPONDER part: Not applicable: An *A-Group-Mgmt ind.* service primitive is never out of context in the RESPONDER.¹⁸

¹⁸Proper *ind./res.* primitive sequencing is enforced by the EASE.

6.2.3.2.4. Timing errors

Error in INITIATOR part: Not applicable: There are no timing restraints placed upon the INITIATOR part of the *Group Management FU*.

Error in RESPONDER part:

Error	Reaction from EASE	Specified action in FU
<p>UE too late responding to <i>A-Group-Mgmt ind.</i></p>	<p>In RESPONDER: Local error from eventual attempt at issuing <i>A-Group-Mgmt res.</i></p> <p>In INITIATOR: <i>A-Group-Mgmt cnf.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Terminate FU invocation locally.</p> <p>INITIATOR part: Terminate FU invocation locally.</p>

6.2.3.2.5. Congestion error

INITIATOR part of the FU:

When occurring with an *A-Group-Mgmt req.* attempt:
Terminate FU invocation locally.

RESPONDER part of the FU:

When occurring with an *A-Group-Mgmt res.* attempt:
Terminate FU invocation locally. (INITIATOR part of FU will eventually be terminated after time-out in the Elcom provider.)

6.2.3.2.6. EASE service primitive parameter errors

Errors in the **A-Group-Mgmt ind.** primitive (detected by the RESPONDER part):

The RESPONDER part shall in all these cases issue *A-Group-Mgmt res.* with Result = <Value from table below>, and the values of Gnr and Gtype as in the corresponding *ind.* The FU must not be invoked.

Error	Value of parameter Result
Function = G-Create , and Gtype value other than those supported by the Responder	<i>gtype-out-of-range</i>
Function = G-Create , and: - Gnr < 1, or - Gnr greater than greatest value of the group attribute Group number that is acceptable in the RESPONDER system	<i>gnr-out-of-range</i>
Function = G-Create , and group no. Gnr already exists in CS(R)	<i>group-exists</i>
Function = G-Create , and a new group descriptor cannot be allocated because of local resource limitations	<i>no-memory</i>
Function = G-Create , and: - Gsize < 1, or - Gsize greater than greatest value of the group attribute Group size that is acceptable in the RESPONDER system	<i>gsize-out-of-range</i>
Function = G-Create , and: - Objlength < 1, or - Objlength greater than greatest value of the attribute Object identifier size that is acceptable in the RESPONDER system	<i>objlength-out-of-range</i>
Function = G-Create or G-Change , and: - Priority Class < 0 or > 15, or - Priority Class not equal to 0, and priority mechanism not supported by the RESPONDER UE	<i>priority-class-out-of-range</i>

cont.

Error	Value of parameter Result
Function = G-Change , and Gtype value not equal to value of attribute Group type in CS(R) for group no. Gnr. (The case of non-existing group no. Gnr is considered below.)	<i>gtype-out-of-range</i>
Function = G-Change , and Gsize value not equal to value of attribute Group size in CS(R) for group no. Gnr. ¹⁹ (The case of non-existing group no. Gnr is considered below.)	<i>gsize-out-of-range</i>
Function = G-Change , and Objlength value not equal to value of attribute Object identifier size in CS(R) for group no. Gnr. ²⁰ (The case of non-existing group no. Gnr is considered below.)	<i>objlength-out-of-range</i>
Function = G-Change , and attribute Persistent = <i>true</i> for the group in CS(R) ²¹	<i>not-deletable</i>
Function = G-Change or G-Delete ²² , and group no. Gnr does not exist in the CS(R).	<i>gnr-out-of-range</i>
Function = G-Delete , and attribute Persistent = <i>true</i> for the group in CS(R)	<i>not-deletable</i>
Function = G-Change or G-Delete , and data transfer is active for group	<i>remote-service-user-unavailable</i>
RESPONDER UE not able to perform the requested function, due to any reason other than those listed above	<i>remote-service-user-unavailable</i>

Note that the RESPONDER UE is **not** allowed to reject a request for deleting ALL non-permanent groups in the CS(R): An *A-Group-Mgmt ind.* primitive with Function = *Delete-all-groups* shall always be valid, having the effect of deleting all groups in the CS(R), if any, for which **Persistent** = *false*. All data transfer FU's to that Elcom partner shall be terminated.

¹⁹Equivalent to requesting a change of attribute Object identifier size for a group within the CS(R). This is illegal; [9], chapter 12.3.1.1.

²⁰Equivalent to requesting a change of attribute Group size for a group within the CS(R). This is illegal; [9], chapter 12.3.1.1.

²¹Equivalent to requesting a change in value of attribute **Persistent** from *true* to *false*, which shall be illegal.

²²Non-existent group to be deleted is considered an error at this stage. However, if it is desirable to treat the erroneous result of attempting to delete a non-existing group as "OK", the INITIATOR UE may very well do so, in the course of handling this *Group Management FU* error at a higher level, not within the scope of this document.

Errors in the **A-Group-Mgmt cnf.** primitive (detected by the INITIATOR part):
In general, when **Result** = *result_ok*, CS(I) shall be updated as normal.²³

Error	Action in INITIATOR part of FU
<p>Mismatch between Gnr/Gtype/Function in primitive and Gnr/Gtype/Function in corresponding <i>A-Group-Mgmt req.</i>, and Result = <i>result-ok</i>.</p> <p>Result >< <i>result-ok</i></p>	<p>Terminate the FU invocation locally, registering the error. Update the configuration integrity control parameter, as normal.²³</p> <p>Terminate the FU invocation locally, registering the error. Do not update the configuration integrity control parameter.</p>

²³Because Result = *result-ok*, the configuration integrity control parameter will have been updated in the RESPONDER system.

6.3. Group Definition FU

Type: Primary.

6.3.1. Function

A *Group Definition FU* invocation allows the invoking INITIATOR UE to modify the attribute **Object set** (component attribute pairs **Object number** and **Object identifier**) within the group descriptor that contains a given value of attribute **Group number**, in the CS(R) of a RESPONDER UE. This is equivalent to defining or redefining objects in a group in the CS(R).

The group shall have been defined via the Elcom interface; not by non-Elcom means.

The FU may be invoked for either a **definition** or a **redefinition**:

In the case of a **definition**, an update of the whole **Object set** of the group shall be attempted.

In the case of a **redefinition**, an update of a specified number of **Object identifiers** within the group shall be attempted. The part which is to be updated is specified by an **Object number** range with a defined maximum span; see below for rules on using the service primitive parameters Index1 and Index2. **Object numbers** themselves cannot be modified during a redefinition.

The INITIATOR and RESPONDER parts of this FU are generally asymmetrical with regard to invocation and termination events, when invoked for **definition**:

The RESPONDER part has no way of knowing whether a completed *A-Def-Group* transaction is the last in the series of transactions constituting the *Group Definition FU* invocation; see chapter 6.3.3.1.1 "Sequence", below. Consequently, the RESPONDER part is generally invoked a number of times, once for each incoming *A-Def-Group* transaction. The INITIATOR part, on the other hand, is invoked only once per invocation of the *Group Definition FU* itself. The term "*Group Definition FU* invocation" should therefore strictly read "INITIATOR part of *Group Definition FU* invocation". However, "*Group Definition FU* invocation" is used throughout this document.

6.3.2. Coordination rules

6.3.2.1. Association usage

All Elcom interactions that are part of one invocation of the *Group Definition FU* are conveyed by one single association. The association shall have the characteristics as specified in chapter 6.3.2.3.1 "Prerequisites", below.

6.3.2.2. Relation to other FUs

6.3.2.2.1. Invoking FUs

The *Group Definition FU* may be invoked by the *Group Configuration FU*.

6.3.2.2.2. Invoked FUs

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

6.3.2.2.3. Disrupting FUs

The *Group Definition FU* may be disrupted by:

- *Permanent Association FU*
- *Dynamic Association FU*

6.3.2.2.4. Disrupted FUs

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

- *Group Configuration FU*
- *Requested Data Transfer FU*
- *Periodically Requested Data Transfer FU*
- *Periodic Data Transfer FU*
- *Unsolicited Data Transfer FU*
- *Unsolicited Mixed Data Transfer FU*
- *Supervisory Control Data Transfer FU*

6.3.2.3. Invocation

6.3.2.3.1. Prerequisites

One²⁴ of the following FUs shall have been invoked preceding any invocation of the *Group Definition FU*:

- *Permanent Association FU*
- *Dynamic Association FU*

The *Permanent Association FU* or *Dynamic Association FU* invocation shall still be running at the time a *Group Definition FU* is invoked. 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 Definition FU*, with the A-suffix pair specified by the table in chapter 5.1.2 "A-suffixes" in section "THE ASSOCIATION MANAGEMENT FUNCTION GROUP".

6.3.2.3.2. Restrictions

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

A *Group Definition FU* must not be invoked while data transfer is active, for the group involved.

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

A *Group Definition FU* must not be invoked while a *Group Definition FU invocation* or a *Group Management FU invocation* is running for another group on another association with the same Elcom partner.²⁵

²⁴Subject to local decision in the INITIATOR UE.

²⁵The purpose of this rule is to ensure the integrity of the configuration integrity control parameter for the INITIATOR system in the RESPONDER system. This parameter is updated by both the *Group Management FU* and the *Group Definition FU*; see chapter 5.5 "Group configuration integrity control" in section 5 "THE ASSOCIATION MANAGEMENT FUNCTION GROUP".

6.3.2.3.3. Invoking events

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

- Local request via the Coordinating Function, the original source of which is outside the scope of this document.
- The *Group Configuration FU*.

Invocation of the RESPONDER part of the *Group Definition FU* is attempted whenever a valid *A-Def-Group ind.* primitive is received via an association with the characteristics as defined for the *Group Definition FU*: See description of the *Permanent Association FU*.

6.3.2.4. Termination

6.3.2.4.1. Orderly termination

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

- Congestion error. See relevant chapter, below.
- Reception of an invalid *A-Def-Group cnf.* service primitive after issuing an *A-Def-Group req.* service primitive (termination on error).
- Reception of a valid *A-Def-Group cnf.* service primitive after issuing the last *A-Def-Group req.* service primitive of the current *Group Definition FU* invocation (normal termination).

The RESPONDER part of a *Group Definition FU* invocation always terminates itself in an orderly manner (normal termination) after issuing an *A-Def-Group res.* service primitive.

6.3.2.4.2. Disruption

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

- Disruption of another FU invocation. See chapter 6.3.2.2.3 "Disrupting FUs", above.

6.3.3. Procedures

6.3.3.1. EASE service primitives

The following elementary EASE service is used by the *Group Definition FU*:

- *A-Def-Group*

6.3.3.1.1. Sequence

A. If invoked for **redefinition**:

The **redefinition** sequence of primitives is a single *A-Def-Group* transaction:

INITIATOR UE	EASE	RESPONDER UE
<i>A-Def-Group req.</i> <i>A-Def-Group cnf.</i>	-----> <-----	<i>A-Def-Group ind.</i> <i>A-Def-Group res.</i>

1. Rules:

1. The **Object number** range, or **subgroup**, that the transaction covers, may span a number of objects from 1 up to a maximum. The maximum is a function of the individual lengths of the corresponding **Object identifiers**, and may vary between transactions.²⁶
2. **Object identifiers** for ALL **Object numbers** within the subgroup (see 1, above) that an individual transaction covers, shall be updated in the CS(R) during the transaction.
3. Upon reception of a valid *A-Def-Group ind.* service primitive, the RESPONDER UE shall:
 - Update group attribute **Object set** in the CS(R) with information from the primitive:
The **Object identifiers** fetched from the parameter Objid(I) at consecutive

²⁶The limiting factor is the maximum number of user octets that a pair of X.25 packets may carry: Normally 256. The number of octets not available for object identifiers is given by:

P-layer overhead:	1 octet
Non-Objid part of D-G-REQ PDU:	8 octets

Sum:	9 octets
------	----------

Thus, the set of **Object identifiers** for the subgroup, coded as defined in [8], chapter 5.6.1, normally must be accommodated within 256-9 = **247** octets.

- ascending index (I)²⁷ values shall be associated with consecutive ascending values of **Object number** in the CS(R), starting with the value of Index1.
- 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 system.²⁸
 - Report the new value of the configuration integrity control parameter back to the INITIATOR UE, via the CF parameter in the *A-Def-Group res.* service primitive.
4. The RESPONDER UE shall return "OK" status (parameter Result(i) = *result-ok* for all i's) if and only if the configuration integrity control parameter has been updated OK.
 5. Upon reception of a valid *A-Def-Group cnf.* 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 system.²⁹
 6. The rules stated in chapter 6.1 "Group attributes", above, also apply here.
 7. The rules stated in the procedure for Configuration Set update, in chapter 5.5 "Group configuration integrity control", also apply.
 8. If *A-Def-Group cnf.* Service primitive signals an error, the group definition sequence should not be continued with *A-Def-Group req.* primitives with higher indexes than those indexes in the *A-Def-Group req.* primitive causing the error. This means that the error(s) should be corrected before transmitting the next subgroup.

²⁷This index is not to be interpreted as an index into an array Objid(I) in the usual sense, but only as an integer ordering number for the elements coded into the Objid parameter. The elements (**Object identifiers**) are ASCII text strings, all consecutively coded into a one-dimensional array of octets; see [8], chapter 5.6.1. Increasing the value of "index I" shall correspond to moving to a greater index value within this octets array.

²⁸See the APDU description, about the configuration integrity control parameter and associated functions.

²⁹The INITIATOR UE will later check his value of the configuration integrity control parameter against the value conveyed by the *A-Connect cnf.* primitive, during each subsequent *Permanent Association FU* or *Dynamic Association FU* invocation 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 section 5 "THE ASSOCIATION MANAGEMENT FUNCTION GROUP".

B. If invoked for **definition**:

The primitive sequence is composed of consecutive *A-Def-Group* transactions (see **A**, above), each covering the following contiguous ranges of ascending **Object number** values:

Transaction no.	Object no. lower bound	Object no. upper bound
1	1	<max(1)> ³⁰
2	<max(1)>+1	<max(2)>
•	•	•
•	•	•
j-1	<max(j-2)>+1	<max(j-1)>
j	<max(j-1)>+1	N
N = total number of objects in group		

Note that if the **Object set** of a group is no bigger than allowing all **Object identifiers** to be coded into a single Objid parameter, the **definition** case may be collapsed into the **redefinition** case: One single *A-Def-Group* transaction.

Rules:

1. The individual transactions shall encompass as many consecutive **Object numbers** as allowed by the EAPI (see footnote at rule **1** for the **redefinition** case, above), except for the last transaction of the FU invocation, which typically will encompass a smaller remaining set of objects.
2. Rules **2** through **8** for the **redefinition** case, above, also apply here, **for each individual transaction**.

³⁰Notation: <max(i)> is the accumulated number of objects whose **Object identifier** has been updated in the CS(R) during the current FU invocation, at the completion of transaction no. i.

6.3.3.1.2. Parameter values

A-Def-Group:

Parameter	req. (INITIATOR)	res. (RESPONDER)
Gtype	<p>Value <i>Measure-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>. Value shall be equal to assumed value of attribute Group type in the CS(R) for group no. Gnr.</p>	Copy of value from <i>ind</i> .
Gnr	<p>Reference number for the group in question. Value of attribute Group number in set of attributes in the CS(R), for which attribute Object set is to be modified</p>	Copy of value from <i>ind</i> .
Index1	<p>Lower bound of contiguous range of values of component attribute Object number for which the corresponding value of component attribute Object identifier is to be updated, in the CS(R). Shall be ≥ 1 and \leq assumed value of attribute Group size for the group, in the CS(R). If whole Object set of group is contained within Objid parameter: Shall be = 1.</p>	Copy of value from <i>ind</i> .
Index2	<p>Upper bound of contiguous range of values of component attribute Object number for which the corresponding value of component attribute Object identifier is to be updated, in the CS(R). Shall be \geq Index1 and \leq assumed value of attribute Group size for the group, in the CS(R). If whole Object set of group is contained within Objid parameter: Shall be equal to the number of Object identifiers coded into Objid.</p>	Copy of value from <i>ind</i> .

cont.

Parameter	req. (INITIATOR)	res. (RESPONDER)
CF	(Not applicable)	Control Field value for the INITIATOR UE that invoked the <i>Group Definition FU</i> . See chapter 5.5 "Group configuration integrity control", in section "THE ASSOCIATION MANAGEMENT FUNCTION GROUP".
Objid(l)	<p>Set of values of component attributes Object identifier for the group.</p> <p>The values are to be placed at ascending consecutive indices (l)³¹, in ascending consecutive order of value of their attribute Object number from the CS(l).</p> <p>Shall contain the following number of values: (Index2 - Index1) + 1.</p> <p>Coding rules are to be found in [8], chapter 5.6.1.</p> <p>For security class 2: The authentication code is appended after the terminating zero octet. The authentication code is generated on the basis of the Objid(l) parameter excluding the terminating zero octet. The size of the Objid(l) plus the authentication code must not exceed the maximum size.</p> <p>For security class 3: The Objid(l) parameter excluding the terminating zero octet is enciphered. A checksum must be present. Possible ways to do this is described in the chapter 9.5 "Suggested use of the security mechanisms".</p>	(Not applicable)

cont.

³¹See footnote about "index l" at the **Rules** list for the **redefinition** case, above.

Parameter	<i>req.</i> (INITIATOR)	<i>res.</i> (RESPONDER)
Result(I)	(Not applicable)	<p>One integer error code per object in the Objid parameter. Lowest value of index, I, is 1. The code at index I concerns the object coded into parameter Objid at location reference number I (see about "index I" in footnote at the Rules list for the redefinition case, above)</p> <p>Result(I) may contain more than one error code at a time (simultaneous errors for more than one object).</p> <p>Action performed as specified: Value of first element (I=1): <i>result-ok.</i></p> <p>Action not performed, due to error condition: If error resulting in error code <i>gtype-out-of-range, gnr-out-of-range, config-buffer-overflow, not-reconfigurable</i> or <i>index-out-of-range</i>: Value of first element (I=1): The error code. If error resulting in error code <i>objlength-out-of-range</i> or <i>objid-unknown</i> for the object corresponding to element no. I: Value of element no. I: The error code.³²</p> <p>In the case of object related errors, the Responder UE shall always code one element per defined object into the Result (I) parameter, irrespective of the degree of success.</p> <p>Also see chapter 6.3.3.2 "Error handling", below. Coding rules are to be found in [8], chapter 5.6.3.</p>

³²Note that the whole transaction shall fail in the RESPONDER UE (no update of the CS(R)), if it fails for at least one object.

6.3.3.2. Error handling

6.3.3.2.1. FU disruption

Disruption by the *Permanent Association FU* or *Dynamic Association FU*:

Disruption of both the INITIATOR part and the RESPONDER part of the current invocation of the *Group Definition FU* shall be triggered locally, as a part of the handling of incoming *A-R-Abort ind.* primitives in both the INITIATOR part and the RESPONDER part of the *Permanent Association FU* or *Dynamic Association FU* invocation handling the association where the *Group Definition FU* is running.

Both parts of the current invocation of the *Group Definition FU* shall be terminated gracefully, neither part attempting to issue any primitive associated with the termination itself.³³

6.3.3.2.2. Illegal invocation attempt

FU not present:

If the *Group Definition FU* is not present in an INITIATOR UE:

The handling of this type of error resulting from an invocation attempt by the local Coordinating Function is a local issue, outside the scope of this document.

If this type of error occurs upon an invocation attempt by a *Group Configuration FU* invocation, the error is handled by the error handling mechanism within the *Group Configuration FU*. See the *Group Configuration FU* description.

If the *Group Definition FU* is not present in an RESPONDER UE:

The RESPONDER UE shall respond to activation attempts in one of two ways:

Either:

- Ignoring the incoming *A-Def-Group ind.* primitive altogether

or:

- Issuing an *A-Def-Group res.* primitive with Result = *remote-service-user-unavailable*

³³Local clean-up procedures are not specified by this document.

FU present, but attempt illegal:

Invocation attempts violating the rules stated in chapter 6.3.2.3.2 "Restrictions" above in an INITIATOR UE is a local issue, outside the scope of this document.

In a RESPONDER UE, invocation attempts violating the same rules shall be handled by the RESPONDER UE in one of two ways:

Either:

- Ignoring the incoming *A-Def-Group ind.* primitive altogether

or:

- Issuing an *A-Def-Group res.* primitive with Result = *remote-service-user-unavailable* without actually (re-)invoking the *Group Definition FU* for the group concerned, in the RESPONDER UE.

6.3.3.2.3. Incoming EASE service primitive out of context

INITIATOR part:

State	<i>A-Def-Group cnf.</i>
FU not running	Ignore, or local error indication
FU running, waiting for <i>A-Def-Group cnf.</i>	(Normal)

RESPONDER part: Not applicable: An *A-Def-Group ind.* service primitive is never out of context in the RESPONDER.

6.3.3.2.4. Timing errors

Error in INITIATOR part: Not applicable: There are no timing restraints placed upon the INITIATOR part of the *Group Definition FU*.

Error in RESPONDER part:

Error	Reaction from EASE	Specified action in FU
<p>UE too late responding to <i>A-Def-Group ind.</i></p>	<p>In RESPONDER: Local error from eventual attempt at issuing <i>A-Def-Group res.</i></p> <p>In INITIATOR: <i>A-Def-Group cnf.</i>, with Result = <i>remote-service-user-unavailable.</i></p>	<p>RESPONDER part: Terminate FU invocation locally.</p> <p>INITIATOR part: Terminate FU invocation locally.</p>

6.3.3.2.5. Congestion error

INITIATOR part of the FU:

When occurring with an *A-Def-Group req.* attempt:
Terminate FU invocation locally.

RESPONDER part of the FU:

When occurring with an *A-Def-Group res.* attempt:
Terminate FU invocation locally. (INITIATOR part of FU will eventually be terminated after time-out in the Elcom provider.)

6.3.3.2.6. EASE service primitive parameter errors

Errors in the **A-Def-Group ind.** primitive (detected by the RESPONDER part):

A. Group-related errors:

The RESPONDER part shall in all these cases issue *A-Def-Group res.* with $Result(1)^{34} = \langle \text{Value from table below} \rangle$, and the values of Gnr, Gtype, Index1 and Index2 as in the corresponding *ind.* The number of elements in Result shall be ≥ 1 . If a group-related error is detected, $Result(i), i > 1$, shall have the value *result-ok*, if present. The RESPONDER part of the FU must not be invoked.

Error	Value of parameter Result(1)
Gtype value not equal to value of attribute Group type in CS(R) for group no. Gnr. (The case of non-existing group no. Gnr is considered below.)	<i>gtype-out-of-range</i>
Value of Gnr is illegal, or group no. Gnr does not exist in the CS(R) or was not created via the Elcom interface	<i>gnr-out-of-range</i>
Index1 ≤ 0 or Index2 ≤ 0	<i>index-out-of-range</i>
Index1 outside legal range of values of component attribute Object number for defined objects in group no. Gnr in CS(R). Legal range: 1 - Group size .	<i>index-out-of-range</i>
Index2 $<$ Index1 or Index2 $>$ upper bound of values of component attribute Object number for defined objects in group no. Gnr in CS(R). Upper bound: Group size .	<i>index-out-of-range</i>
Number of Object identifiers in Objid outside legal range of values of component attribute Object number for group no. Gnr in CS(R). Legal range: 1 - Group size .	<i>index-out-of-range</i>
Number of Object identifiers in Objid not equal to Index2 - Index1 + 1.	<i>index-out-of-range</i>
Consecutive A-Def-Group ind. must contain consecutive Indexes (Index 1 in the nth A-Def_Group ind. = Index2+1 in the (n-1)th A-Def-Group ind.).	<i>index-out-of-range</i>
cont.	

³⁴The first value in the array Result.

Error	Value of parameter Result(1)
The Object set of group no. Gnr in CS(R) cannot be updated as specified, because of local resource limitations	<i>config-buffer-overflow</i>
Attribute Persistent = <i>true</i> , for group no. Gnr in CS(R).	<i>not-reconfigurable</i>
Attribute Static = <i>true</i> , and at least one Object identifier defined, for group no. Gnr in CS(R). ³⁵	<i>not-reconfigurable</i>
Data transfer is active for group no Gnr.	<i>remote-service-user-unavailable</i>
RESPONDER UE not able to perform the requested CS(R) update, due to any reason other than those listed above, and those in the table in sub-clause B, below.	<i>remote-service-user-unavailable</i>
For security class 2: The received authentication code >< the generated authentication code based in the received data.	<i>invalid-authentication-code-received</i>
The security class 3: The received checksum >< the generated checksum during the decipherment.	<i>decipherment-error</i>
<p>Note: Group-related errors shall have precedence over object-related errors (below): Object-related errors shall not be reported unless no group-related error is detected.</p>	

B. Object-related errors:

The RESPONDER part shall in all these cases issue *A-Def-Group res.* with the values of Gnr, Gtype, Index1 and Index2 as in the corresponding *ind. Result(i)*³⁶ shall contain:

- At all indices i corresponding to objects for which an error has been detected:
 <Value from table below>
- At all indices i corresponding to objects for which no error has been detected:
 Value *result-ok*.

The number of elements in Result (max value of index i) shall be sufficient to encompass all indices corresponding to objects for which an error has been detected. The last value in Result (at max i) is not required to be different from *result-ok*.³⁷
The RESPONDER part of the FU must not be invoked.

³⁵Note that if no **Object identifier** is defined (empty group), *A-Def-Group* transactions shall however be legal.

³⁶The value in the array Result at index value i equal to the integer ordering number of the corresponding **Object identifier** in the Objid parameter. (Integer ordering numbers in Objid are described in footnote about "index I" at the **Rules** list for the **redefinition** case, above.)

³⁷Meaning that the RESPONDER UE may choose to always send a full-sized Result(l) parameter: One value per object, irrespective of the degree of transaction success.

Error	Value of parameter Result(i)
Number of octets in Object identifier for object no. i in Objid greater than attribute Object identifier size for group no. Gnr in CS(R)	<i>objlength-out-of-range</i>
Format error in Objid, making decoding of Object identifier for object no. i impossible	<i>objlength-out-of-range</i>
Value of Object identifier for object no. i in Objid not a legal identifier for such objects in the RESPONDER system	<i>objid-unknown</i>

Errors in the **A-Def-Group cnf.** primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Mismatch between Gnr/Gtype/Index1/Index2 in primitive and Gnr/Gtype/Index1/Index2 in corresponding <i>A-Def-Group req.</i> , and Result(i) = <i>result-ok</i> for all i's.	Terminate the FU invocation locally, registering the error. Update the configuration integrity control parameter, as normal. ³⁸
Result(i) >< <i>result-ok</i> for at least one i.	Terminate the FU invocation locally, registering the error. Do not update the configuration integrity control parameter.
<p>Note: In general, when Result = <i>result-ok</i>, the CS(I) shall be updated as normal.</p>	

³⁸Because Result = *result-ok*, the configuration integrity control parameter will have been updated in the RESPONDER system.

6.4. Group Readout FU

Type: Primary.

6.4.1. Function

A *Group Readout FU* invocation allows the invoking INITIATOR UE to fetch the complete attribute set of which a specified value of **Group number** is part, from the CS(R) in the RESPONDER system. This is equivalent to obtaining a readout of a group configuration in the CS(R).

The function shall not be limited to groups that are configured via the EASE: **Groups configured in the CS(R) by any non-Elcom means shall also be included.**

Note that the operation of both the *Group Management FU* and the *Group Definition FU*, on the other hand, is restricted to the groups created via the Elcom interface.

The INITIATOR and RESPONDER parts of this FU are generally asymmetrical with regard to invocation and termination events:

The RESPONDER part has no way of knowing whether a completed *A-Get-Group* transaction is the last in the series of transactions constituting the *Group Readout FU* invocation; see chapter 6.4.3.1.1 "Sequence", below. Consequently, the RESPONDER part is generally invoked a number of times, once for each incoming *A-Get-Group* transaction. The INITIATOR part, on the other hand, is invoked only once per invocation of the *Group Readout FU* itself. The term "*Group Readout FU* invocation" should therefore strictly read "INITIATOR part of *Group Readout FU* invocation". However, "*Group Readout FU* invocation" is used throughout this document.

6.4.2. Coordination rules

6.4.2.1. Association usage

All Elcom interactions that are part of one invocation of the *Group Readout FU* are conveyed by one single association. The association shall have the characteristics as specified in chapter 6.4.2.3.1 "Prerequisites", below.

6.4.2.2. Relation to other FUs

6.4.2.2.1. Invoking FUs

The *Group Readout FU* shall not be invoked by any other FU.

6.4.2.2.2. Invoked FUs

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

6.4.2.2.3. Disrupting FUs

The *Group Readout FU* may be disrupted by:

- *Permanent Association FU*
- *Dynamic Association FU*

6.4.2.2.4. Disrupted FUs

The *Group Readout FU* shall not disrupt any other FU.

6.4.2.3. Invocation

6.4.2.3.1. Prerequisites

One³⁹ of the following FUs shall have been invoked preceding any invocation of the *Group Readout FU*:

- *Permanent Association FU*
- *Dynamic Association FU*

The *Permanent Association FU* or *Dynamic Association FU* invocation shall still be running at the time of invocation of the *Group Readout 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 Readout FU*, with the A-suffix pair specified by the table in chapter 5.1.2 "A-Suffices" in section "THE ASSOCIATION MANAGEMENT FUNCTION GROUP".

6.4.2.3.2. Restrictions

There are no restrictions on invocation of the *Group Readout FU*.

However, if the *Group Readout FU* is invoked while the *Group Management FU* or the *Group Definition FU* is running for the same group, the *Group Readout FU* may fail, or the INITIATOR UE may receive erroneous values. This depends on database implementation details (not covered by this document), as well as actual timing of FU invocations.

6.4.2.3.3. Invoking events

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

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

Invocation of the RESPONDER part of the *Group Readout FU* is attempted whenever a valid *A-Get-Group ind.* primitive is received via an association with the characteristics as defined for the *Group Readout FU*. See description of the *Permanent Association FU*.

³⁹Subject to local decision in the INITIATOR UE.

6.4.2.4. Termination

6.4.2.4.1. Orderly termination

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

- Congestion error. See relevant chapter, below.
- Reception of an invalid *A-Get-Group cnf.* service primitive after issuing an *A-Get-Group req.* service primitive (termination on error).
- Reception of a valid *A-Get-Group cnf.* service primitive after issuing the last *A-Get-Group req.* service primitive of the current *Group Readout FU* invocation (normal termination).

The RESPONDER part of a *Group Readout FU* invocation always terminates itself in an orderly manner (normal termination) after issuing an *A-Get-Group res.* service primitive.

6.4.2.4.2. Disruption

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

- Disruption of another FU invocation. See chapter 6.4.2.2.3 "Disrupting FUs", above.

6.4.3. Procedures

6.4.3.1. EASE service primitives

The following elementary EASE service is used by the *Group Readout FU*:

- *A-Get-Group*

6.4.3.1.1. Sequence

The basic sequence of primitives is the single *A-Get-Group* transaction:

INITIATOR UE	EASE	RESPONDER UE
<i>A-Get-Group req.</i> <i>A-Get-Group cnf.</i>	<p style="text-align: center;">-----></p> <p style="text-align: center;"><-----</p>	<i>A-Get-Group ind.</i> <i>A-Get-Group res.</i>

Rules:

1. The complete primitive sequence is composed of a number of successive basic *A-Get-Group* transactions, each covering a **subgroup**, or contiguous range of **Object number** values. The algorithm below is mandatory for the INITIATOR UE controlling the *Group Readout FU* invocation.

Not having any previous knowledge about group attribute values except for **Group type** and **Group number**, the INITIATOR UE shall first obtain the maximum amount of information via a single *A-Get-Group* transaction. Then there shall follow a number of additional transactions, always specifying the maximum possible remaining number of objects and successively registering the received information, until either:

- The maximum allowed accumulated number of **Object identifiers** for the group have been returned, or:
- A transaction returns no further **Object identifiers**.

Algorithm for INITIATOR UE in *Group Readout FU* (informal pseudocode).

BEGIN

Perform an *A-Get-Group* transaction, with **Object number** range 0 - 0 (parameters Index1 and Index2 both =0).⁴⁰

Register⁴¹, from transaction, the following group attributes:

- **Persistent** (parameter Persist)
- **Static** (parameter Static)
- **Priority class** (parameter Priority Class)
- **Group size** (parameter Gsize)
- **Object identifier size** (parameter Objlength)
- **N** pairs of
 - **Object number** (successive count of **Object identifier** values fetched from parameter Objid, starting with **1** for the value at the lowest index of the Objid parameter when that parameter is viewed as a one-dimensional array of octets, and ending with **N**, for the last value from Objid)
 - **Object identifier** (parameter Objid)

IF NOT communicating with an Elcom-83 based RESPONDER UE:⁴²

Set current object count = N, as determined from the first transaction.

LOOP:

IF current object count is less than value of **Group size**, as determined from the first transaction:

Perform new transaction, with:

Index1 = current object count, plus one

Index2 = value of **Group size**, as determined from the first transaction

IF Objid parameter is not empty (contains at least one **Object identifier**)⁴³:

Register, from transaction, component attribute **Object identifier** for each object, as decoded from the Objid parameter.⁴⁴

Update current object count, adding the new number of objects decoded from the Objid parameter to the previous value of current object count.

ELSE

Exit LOOP

END-IF

ELSE

Exit LOOP

END-IF

END-LOOP

END-IF

Terminate the FU invocation.

END

⁴⁰The purpose of this transaction is to establish a bound for the LOOP below, based on **Group size**, at the same time extracting as many **Object identifiers** as possible during the transaction. The semantics of the **Object number** range 0 - 0, which is legal within the context of this FU, is "return as many **Object identifiers** as possible".

⁴¹That is, report to the Coordinating Function.

⁴²In Elcom-83, there is no support for multi-transaction group readout. The distinction between Elcom-90 and Elcom-83 shall be made as part of the procedure for establishing the association supporting the *Group Readout FU*; see description of the *Permanent Association FU* or *Dynamic Association FU*.

⁴³Note that zero objects in Objid is legal within the context of the *Group Readout FU*.

⁴⁴**Object numbers** shall here be assigned to the decoded **Object identifiers** in the usual ascending numerical sequence, starting with the value of parameter Index1 from the transaction.

Note that the number of transactions required to obtain a complete group readout is normally one larger than the number of transactions required to carry the actual set of **Object identifiers**: The last transaction will usually contain zero **Object identifiers**.⁴⁵

This also applies to groups for which the value of attribute **Group size** is no larger than always allowing the G-G-C-RSP PDU to hold the complete configuration, and to groups for which the value of attribute **Group size** is larger than always allowing the G-G-C-RSP PDU to hold the complete configuration but the actual number of **Object identifiers** being small enough to allow it:

In both cases, the *Group Readout FU* invocation will encompass two *A-Get-Group* transactions.

The underlying reason is that an *A-Get-Group* transaction has no way of simultaneously reporting a complete configuration, and the fact that the configuration is complete.

- The RESPONDER UE shall use the following parameters from the *A-Get-Group ind.* service primitive, in order to identify the group attributes to be returned with the *A-Get-Group res.* service primitive (Rule 3, below):

Parameter	Attribute used for search in the CS(R)
Gtype Gnr	Group type Group number

- The RESPONDER UE shall copy the values of the following group attributes into the following parameters of the *A-Get-Group res.* service primitive:

Attribute	Parameter
Group type Group number Persistent Stavic Priority class Group size Object identifier size	Gtype Gnr Persist Static Priority Class Gsize Objlength

The whole set of component attributes **Object identifier** within the attribute **Object set**, or a subset thereof, shall be coded into the Objid(l) parameter. Consecutive ascending index (l)⁴⁶ values in Objid(l) shall correspond to consecutive ascending values of **Object number** in the CS(R), for the same objects. The first **Object number** value shall be:

⁴⁵However, with groups for which the number of defined **Object identifiers** is equal to the value of attribute **Group size** (groups having their max. number of objects), the additional transaction will not be performed.

⁴⁶This index is not to be interpreted as an index into an array Objid(l) in the usual sense, but only as an integer ordering number for the elements coded into the Objid parameter. The elements (**Object identifiers**) are ASCII text strings, all consecutively coded into a one-dimensional array of octets; see [8], chapter 5.6.1. Increasing the value of "index l" shall correspond to moving to a greater index value within this octets array.

If parameters Index1 = Index2 = 0:1. If not: Value of Index 1.

4. If Index1 >> 0 and Index2 >> 0 and the INITIATOR UE is requesting **Object identifiers** for more objects⁴⁷ than the number currently defined in the CS(R), the RESPONDER UE shall respond with the values of Index1 and Index2 contained in the request, but with the Objid(I) parameter containing no more **Object identifiers** than the number currently defined in the CS(R), starting at the **Object number** value in Index1. As a special case, Objid(I) shall actually contain no **Object identifiers** at all, if Index1 itself is too large. This is also the case when the group is created, but not defined. The RESPONDER UE shall never return a value different from *result-ok* in parameter Result because of the CS(R) not containing the requested number of objects, if the Index1 and Index2 parameters are otherwise valid.
5. The RESPONDER UE shall always pack the maximum possible number of **Object identifiers** into the Objid parameter, when Index1 = Index2 = 0.
6. The case described in rule 4 above shall be the only case in which the number of **Object identifiers** returned in the Objid(I) parameter does not match the values of Index1 and Index2, while parameter Result = *result-ok*. For other values of Result, see "Error handling", below.
7. The rules stated in chapter 6.1 "Group attributes", above, also apply here.

⁴⁷Requested number of objects is given by Index2 - Index1 + 1, if Index1 and Index2 both are not equal to zero.

6.4.3.1.2. Parameter values

A-Get-Group:

Parameter	req. (INITIATOR)	res. (RESPONDER)
Gtype	<p>Value <i>Measure-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>. Value shall be equal to assumed value of attribute Group type in the CS(R) for group no. Gnr.</p>	Copy of value from <i>ind</i> .
Gnr	<p>Reference number for the group in question. Value of attribute Group number in set of attributes in the CS(R), for which attribute values are to be returned, if possible</p>	Copy of value from <i>ind</i> .
Persist	(Not applicable)	Value of attribute Persistent for group no. Gnr in CS(R)
Static	(Not applicable)	Value of attribute Static for group no. Gnr in CS(R)
Priority Class	(Not applicable)	Value of attribute Priority class for group no. Gnr in CS(R)
Gsize	(Not applicable)	Value of attribute Group size for group no. Gnr in CS(R)
Index1	<p>Lower bound of contiguous range of values of component attribute Object number for which the corresponding value of component attribute Object identifier in the CS(R) is to be returned via parameter <i>Objid(1)</i>. Shall be = 0, if <i>Index2=0</i>, signifying <i>return as many Object identifiers as possible</i>. If $>< 0$: Shall be ≥ 1 and \leq assumed value of attribute Group size for the group, in the CS(R).</p>	Copy of value from <i>ind</i> .

cont.

Parameter	<i>req.</i> (INITIATOR)	<i>res.</i> (RESPONDER)
Index2	<p>Upper bound of contiguous range of values of component attribute Object number for which the corresponding value of component attribute Object identifier in the CS(R) is to be returned via parameter Objid(l). Shall be = 0, if Index1=0, signifying <i>return as many Object identifiers as possible.</i> If >< 0: Shall be >= Index1 and <= assumed value of attribute Group size for the group, in the CS(R)</p>	Copy of value from <i>ind.</i>
Objlength	(Not applicable)	Value of attribute Object identifier size for group no. Gnr in CS(R)

cont.

Parameter	req. (INITIATOR)	res. (RESPONDER)
Objlength	(Not applicable)	<p>The set of component attributes Object identifier for group no. Gnr in CS(R), or a subset thereof. The values are to be placed at ascending consecutive indices, I,⁴⁸, in ascending consecutive order of value of their attribute Object number from the CS(R).</p> <p>If Index1 = Index2 = 0: Shall contain as many values as possible, starting with Object number 1. The limiting factor shall be the maximum size of the Objid parameter or the number of values contained in the CS(R), for the group.</p> <p>If not: Shall contain the following number of values: (Index2 - Index1) + 1, or fewer, starting with Object number = Index1. If fewer, the limiting factor shall be the maximum value of Object number in the CS(R), for the group.</p> <p>Coding rules are to be found in [8], chapter 5.6.1.</p> <p>For security class 2: The authentication code is appended after the terminating zero octet. The authentication code is generated on the basis of the Objid(I) parameter excluding the terminating zero octet. The size of the Objid(I) plus the authentication code must not exceed the maximum size.</p> <p>For security class 3: The Objid(I) parameter excluding the terminating zero octet is enciphered. A checksum must be present. Possible ways to do this is described in chapter 9.5 "Suggested use of the security mechanisms".</p>

cont.

⁴⁸See footnote about "index I" at the **Rules** list, above.

Parameter	<i>req.</i> (INITIATOR)	<i>res.</i> (RESPONDER)
Result	(Not applicable)	<p>Action performed as specified (parameters Persist, Static, Priority Class, Gsize, Objlength and Objid(I) are valid): <i>result-ok.</i></p> <p>Action not performed, due to error condition (parameters Persist, Static, Priority Class, Gsize, Objlength and Objid(I) are not valid): Error code: Value >< <i>result-</i> <i>ok.</i> See chapter 6.4.3.2 "Error handling", below.</p>

6.4.3.2. Error handling

6.4.3.2.1. FU disruption

Disruption by the *Permanent Association FU* or *Dynamic Association FU*:

Disruption of both the INITIATOR part and the RESPONDER part of the current invocation of the *Group Readout FU* shall be triggered locally, as a part of the handling of incoming *A-P-Abort ind.* primitives in both the INITIATOR part and the RESPONDER part of the *Permanent Association FU* or *Dynamic Association FU* invocation handling the association where the *Group Readout FU* is running.

Both parts of the current invocation of the *Group Readout FU* shall be terminated gracefully, neither part attempting to issue any primitive associated with the termination itself.⁴⁹

6.4.3.2.2. Illegal invocation attempt

FU not present:

If the *Group Readout FU* is not present in an INITIATOR UE:

The handling of this type of error resulting from an invocation attempt by the local Coordinating Function is a local issue, outside the scope of this document.

If this type of error occurs upon an invocation attempt by a *Group Configuration FU* invocation, the error is handled by the error handling mechanism within the *Group Configuration FU*. See the *Group Configuration FU* description.

If the *Group Readout FU* is not present in an RESPONDER UE:

The RESPONDER UE shall respond to activation attempts in one of two ways:

Either:

- Ignoring the incoming *A-Get-Group ind.* primitive altogether

or:

- Issuing an *A-Get-Group res.* primitive with Result = *remote-service-user-unavailable*

FU present, but attempt illegal:

Invocation attempts are always legal; see chapter 6.4.2.3.2 "Restrictions", above.

⁴⁹Local clean-up procedures are not specified by this document.

6.4.3.2.3. Incoming EASE service primitive out of context

INITIATOR part:

State	<i>A-Get-Group cnf.</i>
FU not running	Ignore, or local error indication
FU running, waiting for <i>A-Get-Group cnf.</i>	(Normal)

RESPONDER part:

Not applicable: An *A-Get-Group ind.* service primitive is never out of context in the RESPONDER.⁵⁰

6.4.3.2.4. Timing errors

Error in INITIATOR part:

Not applicable: There are no timing restraints placed upon the INITIATOR part of the *Group Readout FU*.

Error in RESPONDER part:

Error	Reaction from EASE	Specified action in FU
UE too late responding to <i>A-Get-Group ind.</i>	<p>In RESPONDER: Local error from eventual attempt at issuing <i>A-Get-Group res.</i></p> <p>In INITIATOR: <i>A-Get-Group cnf.</i>, with Result = <i>remote-service-user-unavailable.</i></p>	<p>RESPONDER part: Terminate FU invocation locally.</p> <p>INITIATOR part: Terminate FU invocation locally.</p>

⁵⁰Proper *ind./res.* primitive sequencing is enforced by the EASE. This document does not specify any enforcement of subgroup sequencing across individual *A-Get-Group* transactions, on the part of the RESPONDER UE.

6.4.3.2.5. Congestion error

INITIATOR part of the FU:

When occurring with an *A-Get-Group req.* attempt:
Terminate FU invocation locally.

RESPONDER part of the FU:

When occurring with an *A-Get-Group res.* attempt:
Terminate FU invocation locally. (INITIATOR part of FU will eventually be terminated after time-out in the Elcom provider.)

6.4.3.2.6. EASE service primitive parameter errors

Errors in the *A-Get-Group ind.* primitive (detected by the RESPONDER part):

The RESPONDER part shall in all these cases issue *A-Get-Group res.* with Result = <Value from table below>, and the values of Gnr, Gtype, Index1 and Index2 as in the corresponding *ind.*

The RESPONDER part of the FU must not be invoked.

Error	Value of parameter Result
Gtype value not equal to value of attribute Group type in CS(R) for group no. Gnr. (The case of non-existing group no. Gnr is considered below.)	<i>gtype-out-of-range</i>
Value of Gnr is illegal, or group no. Gnr does not exist in the CS(R).	<i>gnr-out-of-range</i>
Index1 = 0 and Index2 >< 0, or: Index2 = 0 and Index1 >< 0, or: Index1 < 0 or Index2 < 0	<i>index-out-of-range</i>
Index1 >< 0 and Index2 < Index1	<i>index-out-of-range</i>
Index1 > number of objects in group no. Gnr	<i>index-out-of-range</i>
The Object set of group no. Gnr in CS(R) cannot be returned as specified, because of local resource limitations, or any other reason than those listed above	<i>remote-service-user-unavailable</i>

Errors in the **A-Get-Group cnf.** primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
<p>Result = <i>result-ok</i>, and mismatch between Gnr/Gtype/Index1/Index2 in primitive and Gnr/Gtype/Index1/Index2 in corresponding <i>A-Get-Group req.</i>.</p>	<p>Terminate the FU invocation locally, registering the error.</p>
<p>Result = <i>result-ok</i>, and format error in Objid</p>	<p>Terminate the FU invocation locally, registering the error.</p>
<p>Result = <i>result-ok</i>, and Index1 >< 0 and Index2 >< 0 and the number of Object identifiers coded into Objid greater than the value Index2 - Index1 + 1</p>	<p>Terminate the FU invocation locally, registering the error.</p>
<p>Result = <i>result-ok</i>, and the number of Object identifiers accumulated during the current FU invocation greater than the value of attribute Group size for group no. Gnr in CS(l)</p>	<p>Terminate the FU invocation locally, registering the error.</p>
<p>Result >< <i>result-ok</i></p>	<p>Terminate the FU invocation locally, registering the error.</p>
<p>For security class 2: The received authentication code >< the generated authentication code based in the received data.</p>	<p>Terminate the FU invocation locally, registering the error.</p>
<p>For security class 3: The received checksum >< the generated checksum during the decipherment.</p>	<p>Terminate the FU invocation locally, registering the error.</p>

6.5. Group Configuration FU

Type: Composite.

6.5.1. Function

A *Group Configuration FU* invocation adheres to the following procedure, for a single specified group:

1. The INITIATOR UE invokes the *Group Management FU* for the group, with function *G-Create*, in order to establish a descriptor for the group in the CS(R) in the RESPONDER system.
2. The INITIATOR UE invokes the *Group Definition FU* for the group with the specified object definition set, in order to define the group in the CS(R).

6.5.2. Coordination rules

6.5.2.1. Association usage

The Elcom interactions that are part of the *Group Management FU* and *Group Definition FU* invocations invoked by the *Group Configuration FU* are conveyed by associations with the characteristics as specified for the *Group Management FU* and *Group Definition FU*.

6.5.2.2. Relation to other FUs

6.5.2.2.1. Invoking FUs

The *Group Configuration FU* may be invoked by the following FU:
- *Restart Reconfigure FU*

6.5.2.2.2. Invoked FUs

The *Group Configuration FU* may invoke the following FUs:
- *Group Management FU*
- *Group Definition FU*

6.5.2.2.3. Disrupting FUs

The *Group Configuration FU* may be disrupted by disruption of a supporting FU invocation.

6.5.2.2.4. Disrupted FUs

The *Group Configuration FU* shall not disrupt any other FU.

The FU's invoked by the *Group Configuration FU* may, however, disrupt any other FU.

6.5.2.3. Invocation

6.5.2.3.1. Prerequisites

No previous or concurrent FU invocations are required⁵¹.

6.5.2.3.2. Restrictions

A *Group Configuration FU* invocation as such is allowed at any time. However, it will fail if violating rules stated in the "Restrictions" section of the descriptions of the FUs that the *Group Configuration FU* invokes: See chapter 6.5.2.2.2 "Invoked FUs", above.

6.5.2.3.3. Invoking events

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

- Local request via the Coordinating Function, the original source of which is outside the scope of this document.
- The *Restart Reconfigure FU*.

The *Group Configuration FU* is of the composite type. Consequently, there exists no specific RESPONDER part of the *Group Configuration FU*, with an associated specific invoking event.

⁵¹ *Permanent Association FU* or *Dynamic Association FU* invocations are a requirement of the *Group Management FU* and *Group Definition FU* invoked by the *Group Configuration FU*, not of the *Group Configuration FU* itself.

6.5.2.4. Termination

6.5.2.4.1. Orderly termination

Orderly termination of the *Group Configuration FU* shall take place after orderly termination of the last FU invocation invoked by the *Group Configuration FU*.

Unexpected orderly termination of any FU that the *Group Configuration FU* invokes is considered an error within the *Group Configuration FU* invocation, and shall also lead to orderly termination of the *Group Configuration FU*.

6.5.2.4.2. Disruption

The INITIATOR part of a *Group Configuration FU* invocation may be disrupted by:
- Disruption of another FU invocation. See chapter 6.5.2.2.3 "Disrupting FUs", above.

The *Group Configuration FU* is of the composite type. Consequently, there exists no specific RESPONDER part of the *Group Configuration FU*, to be disrupted.

6.5.3. Procedures

6.5.3.1. EASE service primitives

The *Group Configuration FU* is of the composite type, having no specific EASE service primitive sequence associated with it.

6.5.3.1.1. Action sequence

The normal action sequence has 2 phases:

Phase 1: Creating the group in the CS(R) of the RESPONDER system.

If the prerequisites and restrictions that apply to a **Group Management FU** invocation for the group can be met:

The INITIATOR UE invokes the **Group Management FU** with function *G-Create* for the group, with specified values of the following attributes.

- **Group type**
- **Group number**
- **Group size**
- **Object identifier size**
- **Persistent**
- **Static**
- **Priority class**

If the **Group Management FU** is NOT running OK or the **Group Management FU** invocation does NOT complete OK:

The **Group Configuration FU** invocation terminates, with error indication.

If not:

The **Group Configuration FU** invocation terminates, with error indication.

Phase 2: Defining the group in the CS(R) of the RESPONDER system.

If the prerequisites and restrictions that apply to a **Group Definition FU** invocation for the group can be met:

The INITIATOR UE invokes the **Group Definition FU** for **definition** of the group, with specified values of the following attributes.

- **Group type**
- **Group number**
- **Object set**

If the **Group Definition FU** is NOT running OK or the **Group Definition FU** invocation does NOT complete OK:

The **Group Configuration FU** invocation terminates, with error indication.

If not:

The **Group Configuration FU** invocation terminates, with error indication.

6.5.3.1.2. Parameter values

The FU is of the composite type. See description of component FUs.

6.5.3.2. Error handling

Handling of errors occurring with EASE service primitives are not considered here, since the *Group Configuration FU* is of the composite type.

Errors occurring in FUs invoked by the *Group Configuration FU* are described in chapter 6.5.3.1.1 "Action sequence", above. Further processing of such errors is a local matter in the INITIATOR UE, not specified by this document. However, errors shall be reported to the invoking FU, if any.

6.5.3.2.1. FU disruption

Disruption of the *Group Configuration FU* shall be further processed by the INITIATOR UE as for other *Group Configuration FU* errors; see chapter "Error handling", above.

6.5.3.2.2. Illegal invocation attempt

FU not present:

If the *Group Configuration FU* is not present in an INITIATOR UE:

Invocation requests are always generated locally; see chapter 6.5.2.3 "Invocation", above. Consequently, the handling of this type of error is a local issue, outside the scope of this document. However, such errors shall be reported to the invoking FU, if any.

No specific RESPONDER part is defined for the *Group Configuration FU*; since it is of the composite type.

FU present, but attempt illegal:

Group Configuration FU invocation attempts as such are always legal. See chapter 6.5.2.3.2 "Restrictions", above.

Section 7 : THE DATA TRANSFER FUNCTION GROUP

TABLE OF CONTENTS

7. THE DATA TRANSFER FUNCTION GROUP	7-1
7.1. Requested Data Transfer FU	7-1
7.1.1. Function	7-1
7.1.2. Coordination rules	7-3
7.1.2.1. Association usage	7-3
7.1.2.2. Relation to other FUs	7-3
7.1.2.2.1. Invoking FUs	7-3
7.1.2.2.2. Invoked FUs	7-3
7.1.2.2.3. Disrupting FUs	7-3
7.1.2.2.4. Disrupted FUs	7-3
7.1.2.3. Invocation	7-4
7.1.2.3.1. Prerequisites	7-4
7.1.2.3.2. Restrictions	7-4
7.1.2.3.3. Invoking events	7-5
7.1.2.4. Termination	7-5
7.1.2.4.1. Orderly termination	7-5
7.1.2.4.2. Disruption	7-5
7.1.3. Procedures	7-6
7.1.3.1. EASE service primitives	7-6
7.1.3.1.1. Sequence	7-6
7.1.3.1.2. Parameter values	7-9
7.1.3.2. Error handling	7-12
7.1.3.2.1. FU disruption	7-12
7.1.3.2.2. Illegal invocation attempt	7-12
7.1.3.2.3. Incoming EASE service primitive out of context	7-13
7.1.3.2.4. Timing errors	7-14
7.1.3.2.5. Congestion error	7-15
7.1.3.2.6. EASE service primitive parameter errors	7-16
7.2. Periodically Requested Data Transfer FU	7-19
7.2.1. Function	7-19
7.2.2. Coordination rules	7-19
7.2.2.1. Association usage	7-19
7.2.2.2. Relation to other FUs	7-20
7.2.2.2.1. Invoking FUs	7-20
7.2.2.2.2. Invoked FUs	7-20
7.2.2.2.3. Disrupting FUs	7-20
7.2.2.2.4. Disrupted FUs	7-20
7.2.2.3. Invocation	7-21
7.2.2.3.1. Prerequisites	7-21
7.2.2.3.2. Restrictions	7-21
7.2.2.3.3. Invoking events	7-21
7.2.2.4. Termination	7-22
7.2.2.4.1. Orderly termination	7-22
7.2.2.4.2. Disruption	7-22
7.2.3. Procedures	7-22
7.2.3.1. EASE service primitives	7-22
7.2.3.1.1. Action sequence	7-23
7.2.3.1.2. Parameter values	7-23
7.2.3.2. Error handling	7-23
7.2.3.2.1. FU disruption	7-24

7.2.3.2.2. Illegal invocation attempt	7-24
7.3. Periodic Data Transfer FU	7-25
7.3.1. Function	7-25
7.3.2. Coordination rules	7-26
7.3.2.1. Association usage	7-26
7.3.2.2. Relation to other FUs	7-26
7.3.2.2.1. Invoking FUs	7-26
7.3.2.2.2. Invoked FUs	7-26
7.3.2.2.3. Disrupting FUs	7-26
7.3.2.2.4. Disrupted FUs	7-26
7.3.2.3. Invocation	7-27
7.3.2.3.1. Prerequisites	7-27
7.3.2.3.2. Restrictions	7-27
7.3.2.3.3. Invoking events	7-28
7.3.2.4. Termination	7-28
7.3.2.4.1. Orderly termination	7-28
7.3.2.4.2. Disruption	7-28
7.3.3. Procedures	7-29
7.3.3.1. EASE service primitives	7-29
7.3.3.1.1. Sequence	7-29
7.3.3.1.2. Parameter values	7-32
7.3.3.2. Error handling	7-34
7.3.3.2.1. FU disruption	7-34
7.3.3.2.2. Illegal invocation attempt	7-34
7.3.3.2.3. Incoming EASE service primitive out of context	7-35
7.3.3.2.4. Timing errors	7-36
7.3.3.2.5. Congestion error	7-38
7.3.3.2.6. EASE service primitive parameter errors	7-39
7.4. Unsolicited Data Transfer FU	7-42
7.4.1. Function	7-42
7.4.2. Coordination rules	7-43
7.4.2.1. Association usage	7-43
7.4.2.2. Relation to other FUs	7-43
7.4.2.2.1. Invoking FUs	7-43
7.4.2.2.2. Invoked FUs	7-43
7.4.2.2.3. Disrupting FUs	7-43
7.4.2.2.4. Disrupted FUs	7-43
7.4.2.3. Invocation	7-44
7.4.2.3.1. Prerequisites	7-44
7.4.2.3.2. Restrictions	7-44
7.4.2.3.3. Invoking events	7-45
7.4.2.4. Termination	7-45
7.4.2.4.1. Orderly termination	7-45
7.4.2.4.2. Disruption	7-45
7.4.3. Procedures	7-46
7.4.3.1. EASE service primitives	7-46
7.4.3.1.1. Sequence	7-46
7.4.3.1.2. Parameter values	7-48
7.4.3.2. Error handling	7-51
7.4.3.2.1. FU disruption	7-51
7.4.3.2.2. Illegal invocation attempt	7-51
7.4.3.2.3. Incoming EASE service primitive out of context	7-52
7.4.3.2.4. Timing errors	7-53
7.4.3.2.5. Congestion error	7-55
7.4.3.2.6. EASE service primitive parameter errors	7-56
7.5. Unsolicited Mixed Data Transfer FU	7-59

7.5.1. Function	7-59
7.5.2. Coordination rules	7-61
7.5.2.1. Association usage	7-61
7.5.2.2. Relation to other FUs	7-61
7.5.2.2.1. Invoking FUs	7-61
7.5.2.2.2. Invoked FUs	7-61
7.5.2.2.3. Disrupting FUs	7-61
7.5.2.2.4. Disrupted FUs	7-61
7.5.2.3. Invocation	7-62
7.5.2.3.1. Prerequisites	7-62
7.5.2.3.2. Restrictions	7-62
7.5.2.3.3. Invoking events	7-63
7.5.2.4. Termination	7-63
7.5.2.4.1. Orderly termination	7-63
7.5.2.4.2. Disruption	7-63
7.5.3. Procedures	7-64
7.5.3.1. EASE service primitives	7-64
7.5.3.1.1. Sequence	7-64
7.5.3.1.2. Parameter values	7-65
7.5.3.2. Error handling	7-66
7.5.3.2.1. FU disruption	7-66
7.5.3.2.2. Illegal invocation attempt	7-66
7.5.3.2.3. Incoming EASE service primitive out of context	7-66
7.5.3.2.4. Timing errors	7-66
7.5.3.2.5. Congestion error	7-67
7.5.3.2.6. EASE service primitive parameter errors	7-68
7.6. Supervisory Control Data Transfer FU	7-69
7.6.1. Function	7-69
7.6.2. Coordination rules	7-70
7.6.2.1. Association usage	7-70
7.6.2.2. Relation to other FUs	7-70
7.6.2.2.1. Invoking FUs	7-70
7.6.2.2.2. Invoked FUs	7-70
7.6.2.2.3. Disrupting FUs	7-70
7.6.2.2.4. Disrupted FUs	7-70
7.6.2.3. Invocation	7-71
7.6.2.3.1. Prerequisites	7-71
7.6.2.3.2. Restrictions	7-71
7.6.2.3.3. Invoking events	7-72
7.6.2.4. Termination	7-72
7.6.2.4.1. Orderly termination	7-72
7.6.2.4.2. Disruption	7-72
7.6.3. Procedures	7-73
7.6.3.1. EASE service primitives	7-73
7.6.3.1.1. Sequence	7-73
7.6.3.1.2. Parameter values	7-75
7.6.3.2. Error handling	7-79
7.6.3.2.1. FU disruption	7-79
7.6.3.2.2. Illegal invocation attempt	7-79
7.6.3.2.3. Incoming EASE service primitive out of context	7-80
7.6.3.2.4. Timing errors	7-80
7.6.3.2.5. Congestion error	7-81
7.6.3.2.6. EASE service primitive parameter errors	7-82

7. THE DATA TRANSFER FUNCTION GROUP

7.1. Requested Data Transfer FU

Type: Primary.

7.1.1. Function

A *Requested Data Transfer FU* invocation shall adhere to the following procedure:

1. The INITIATOR UE instructs the RESPONDER UE to return a specified set of data, subject to the following general rules:

- The data shall be associated with one specified Elcom group only, but shall comprise the requested number of incarnations. The whole group or a subset thereof may be requested.
- The same number of data values shall be returned per invocation.
- Each transmission shall contain data for one incarnation only.
- The sequence of the data values in a transmission shall be the same as in the group definition.
- The sequence of the data values in a transmission may be a subset of the requested set of data for one incarnation. In this case, the data conveyed by a number of successive transmissions shall together constitute the complete set of requested data for the incarnation. Also, the individual transmissions associated with one incarnation shall be ordered in time according to the value of the lower bound of object numbers conveyed; lowest value first.
- For groups of type *Text-message-group*, each transmission shall contain the data value of exactly one object. In other words, for groups of this type each object shall constitute a data subset.
- One single incarnation of the data value of any one object shall always be contained within a single transmission.¹
- The sequence of incarnations shall be ordered according to time, oldest incarnation first.
- The RESPONDER UE shall, for a given invocation, return the set of data values that were currently valid in the RESPONDER system at the point in time computed from the parameters in the request, for that invocation.
- The RESPONDER UE shall always return the number of incarnations specified by the parameter *Periods* of the request, except for the case when the first² element in

¹The limiting factor is the maximum number of user octets in a A-PDU. An A-PDU normally carries 256 octets.

The number of octets not available for data values is given by:

P-layer overhead:	1 octet
Non-Data value part of D-T-REQ PDU:	19 octets

Sum:	20 octets
------	-----------

Thus, the set of **Data** for the subgroup, coded as defined in [8], section 5.8.3, normally must be accommodated within 256-20 = **236** octets.

parameter T0 is equal to -1, when only one incarnation (latest set of values) shall be returned.

- The RESPONDER UE shall transmit with minimal delay between individual transmissions.
- The RESPONDER UE shall specify non-acknowledged operation with each transmission, except for the last transmission, for which acknowledged operation shall be specified.

2. The RESPONDER UE returns the requested data, in the course of a number of successive transmissions.

3. Following the last data transmission, which signals acknowledged operation, the INITIATOR UE acknowledges the data, terminating the FU invocation.

²At lowest index value.

7.1.2. Coordination rules

7.1.2.1. Association usage

All Elcom interactions that are part of one invocation of the *Requested Data Transfer FU* are conveyed by one single association. The association shall have the characteristics as specified in the chapter 7.1.2.3.1 "Prerequisites", below.

7.1.2.2. Relation to other FUs

7.1.2.2.1. Invoking FUs

The *Requested Data Transfer FU* may be invoked by the *Periodically Requested Data Transfer FU*.

7.1.2.2.2. Invoked FUs

The *Requested Data Transfer FU* shall not invoke any other FU.

7.1.2.2.3. Disrupting FUs

The *Requested Data Transfer FU* may be disrupted by:

- *Permanent Association FU*
- *Dynamic Association FU*
- *Group Management FU*
- *Group Definition FU*
- *Restart Reconfigure FU*

7.1.2.2.4. Disrupted FUs

The *Requested Data Transfer FU* shall not disrupt any other FU.

7.1.2.3. Invocation

7.1.2.3.1. Prerequisites

The following FUs shall have been invoked preceding any invocation of the *Requested Data Transfer FU*:

- The *Permanent Association FU* or *Dynamic Association FU*
- The *Group Configuration FU*³

The *Permanent Association FU* or *Dynamic Association FU* invocation shall still be running at the time the *Requested Data Transfer FU* is invoked, and in the case of the *Permanent Association FU*, the association maintained by the *Permanent Association FU* invocation shall be running (not temporarily broken) at that time.

The *Group Configuration FU* invocation shall be terminated when the *Requested Data Transfer FU* is invoked.

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 *Requested Data Transfer FU*, with the A-suffix pair as specified in the table in chapter 5.1.2 "A-Suffixes" in section "THE ASSOCIATION MANAGEMENT FUNCTION GROUP".

The *Group Configuration FU* shall have been invoked in order to define and configure the group that is to be transmitted, prior to the current invocation of the *Requested Data Transfer FU*.⁴

7.1.2.3.2. Restrictions

For any given INITIATOR/RESPONDER system combination, multiple simultaneous invocations of the *Requested Data Transfer FU* on the same association are not allowed. This means that only one group at a time may be requested.

The *Requested Data Transfer FU* must not be invoked while at least one of the following FUs are running, for the group involved:

- *Group Configuration FU*
- *Group Management FU*
- *Group Definition FU*

Neither shall the *Requested Data Transfer FU* be invoked on a given association, if the *Periodic Data Transfer FU* is already running on the same association.

An implementation of the DRFU may be restricted to only support present information (no archive information) on an association where an Unsolicited Data Transfer FU is already running.

³Alternatively, the *Group Management FU* and *Group Definition FU*, which are invoked by the *Group Configuration FU*, may have been invoked directly by the Coordinating Function.

⁴The use of the *Group Configuration FU* is, however, not required if the group is defined in the RESPONDER AP by a non-Elcom mechanism and also made known to the INITIATOR AP by a non-Elcom mechanism.

7.1.2.3.3. Invoking events

The INITIATOR part of the *Requested Data Transfer FU* may be invoked by:

- Local request via the Coordinating Function, the original source of which is outside the scope of this document.
- The *Periodically Requested Data Transfer FU*.

Invocation of the RESPONDER part of the *Requested Data Transfer FU* is attempted whenever a valid *A-Init-Transfer ind.* primitive is received via an association with the characteristics as defined for the *Requested Data Transfer FU*: See description of the *Permanent Association FU*.

7.1.2.4. Termination

7.1.2.4.1. Orderly termination

Orderly termination of the INITIATOR part of a *Requested Data Transfer FU* invocation may only be triggered by:

- Local request via the Coordinating Function, the original source of which is outside the scope of this document (premature termination).
- Local accumulated error count becoming too great. See "Error handling", below, on timing errors.
- Congestion error. See relevant chapter, below.
- Reception of an *A-Data(init⁵, F⁶) ind.* service primitive after issuing an *A-Init-Transfer req.service* primitive (normal termination).

During orderly termination, except for the case of congestion error, the INITIATOR UE shall try to issue an *A-Conf-Data req.* primitive.

The RESPONDER part of a *Requested Data Transfer FU* invocation always terminates itself in an orderly manner (normal termination) upon reception of an *A-Conf-Data ind.* primitive.

7.1.2.4.2. Disruption

Both the INITIATOR and the RESPONDER part of a *Requested Data Transfer FU* invocation may be disrupted by:

- Disruption of another FU invocation. See chapter 7.1.2.2.3 "Disrupting FUs", above.
- A fatal error condition encountered during operation. See chapter 7.1.3.2 "Error handling", below.

⁵Parameter Transmod = *initiated*.

⁶Parameter More-D = *false*.

7.1.3. Procedures

7.1.3.1. EASE service primitives

The following elementary EASE services are used by the *Requested Data Transfer FU*:

- *A-Init-Transfer*
- *A-Data (init.)*
- *A-Conf-Data (init.)*

7.1.3.1.1. Sequence

The normal sequence of primitives is partitioned into **3** phases:

Phase 1: Requesting data.

INITIATOR UE	EASE	RESPONDER UE
<i>A-Init-Transfer req.</i>	----->	<i>A-Init-Transfer ind.</i>

Phase 2: Data transmission.⁷

INITIATOR UE	EASE	RESPONDER UE
<i>A-Data (init., T) ind.</i> <i>A-Data (init., T) ind.</i>	<----- <----- . .	<i>A-Data (init., T) req.</i> <i>A-Data (init., T) req.</i>
<i>A-Data (init., T) ind.</i> <i>A-Data (init., F) ind.</i>	<----- <-----	<i>A-Data (init., T) req.</i> <i>A-Data (init., F) req.</i>

In this phase, the following rules apply:

1. This document places no restriction on the number of consecutive *A-Data (init., T) req.* primitives that may be issued before the terminating *A-Data (init., F) req.* is issued. However, if too much time elapses between any *A-Data (init., T)* primitive and the succeeding *A-Data (init., T)* or *A-Data (init., F)* primitive, an error situation occurs in the EASE. See chapter 7.1.3.2 "Error handling", below.
2. The RESPONDER UE is responsible for determining the timing of the individual *A-Data (init.) req.* primitives. However, the primitives shall be issued in without undue delay.
3. The RESPONDER UE shall report the number of data incarnations specified by the parameter Periods from the request (*A-Init-Transfer* transaction). Each incarnation shall contain the data values that were (will be) valid **in the RESPONDER system at the points in time defined by the parameters T0, Dt, T-Unit and Periods** from the request.
4. If at least one of the requested incarnations does exist in the RESPONDER system, the RESPONDER UE shall return the requested number of incarnations, setting the quality code = *not-ok* for all data values in all incarnations that do not exist. The case of no existing incarnation is considered an error; see below.
5. The case of parameters Dt and T-Unit in the request specifying a time interval **unequal** than the archiving time interval in the RESPONDER system, is considered an error; see below.
6. The rules stated in chapter 7.1.1 "Function", above, also apply here.

⁷Notation: *T* and *F* signifies, for the parameter More-D, the values *true* and *false*, respectively.

Phase 3: Orderly termination: Data acknowledgement.

INITIATOR UE	EASE	RESPONDER UE
<i>A-Conf-Data (init.) req.</i>	----->	<i>A-Conf-Data (init) ind.</i>

The following rules apply:

1. The transition from phase 2 to phase 3 shall be triggered by the INITIATOR UE receiving an *A-Data (init., F) ind.* primitive: The RESPONDER UE flags the fact that all requested data are transferred, by setting parameter More-D = *false* with the last transmission.

7.1.3.1.2. Parameter values

A-Init-Transfer:

Parameter	req. (INITIATOR)
Gtype	Value <i>Measure-group, Status-group, Discrete-group, Logical-breaker status-group, or Text-message-group</i> , depending on the type of the group in question, as assumed in the CS(R).
Gnr	Reference number for the group in question, as assumed in the CS(R).
Index1	Lower bound of contiguous range of values of component attribute Object numbers for which data are to be returned, for each requested incarnation. Shall be = 0, if Index2=0, signifying <i>return data for all defined Object identifiers within the group</i> . If >< 0: Shall be >= 1 and <= value of attribute Group size for the group, in the CS(R).
Index2	Upper bound of contiguous range of values of component attribute Object numbers for which data are to be returned, for each requested incarnation. Shall be = 0, if Index1=0, signifying <i>return data for all defined Object identifiers within the group..</i> If >< 0: Shall be >= Index1 and <= value of attribute Group size for the group, in the CS(R).
T0	Time stamp for oldest requested data incarnation, coded into an integer array according to [8], chapter 5.8.1. The use of UTC is recommended, please refer Appendix F for ELCOM-83 compatibility. First element in array (at lowest index) equal to -1 shall correspond to requesting latest incarnation only.
Dt	Requested time interval between successive data incarnations Can have any value which the EAPI accepts, if first element in T0 = -1.
T-Unit	Unit for parameter Dt Can have any value which the EAPI accepts, if first element in T0 = -1.
Periods	Requested number of data incarnations. Can have any value which the EAPI accepts, if first element in T0 = -1.

A-Data (init.,) req. (RESPONDER):

Parameter	req.
Gtype	Copy of value from request (<i>A-Init-Transfer</i> of Phase 1)
Gnr	Copy of value from request (<i>A-Init-Transfer</i> of Phase 1)
Transmod	= <i>initiated</i>
Index1 ⁸	Shall be > 0: Equal to the lowest Object number , in accordance with the CS(R), of the range of Object numbers whose data values are contained in the Data parameter (below). Shall be equal to Index2, if Gtype = <i>Text-message-group</i> . ⁹
Index2	Shall be >= value of Index1: Equal to the highest Object number , in accordance with the CS(R), of the range of Object numbers whose data values are contained in the Data parameter (below). Shall be equal to Index1, if Gtype = <i>Text-message-group</i> .
T	Time stamp, applying to the whole set of values contained in the Data parameter (below), and determined by the RESPONDER UE. The use of UTC is recommended, please refer Appendix F for ELCOM-83 compatibility.
More-D	If current <i>A-Data (init.)</i> primitive is not the last carrying requested data, so that another will follow shortly: = <i>true</i> . If current <i>A-Data (init.)</i> primitive is the last carrying requested data: = <i>false</i> .
Data	The actual Elcom data transferred. The structure is defined in Appendix A of this document
Length	Length of Data in octets. Shall be computed according to the length of the actual datatype, Index1 and Index2.
Result	= <i>result-ok</i>

⁸The number of values that shall be contained in the Data parameter can be computed as: Index2 - Index1 + 1.

⁹Only one object per transmission, for such groups.

A-Conf-Data (init.) req. (INITIATOR):

Parameter	<i>req.</i>
Gtype	Copy of value from request (<i>A-Init-Transfer</i> of Phase 1)
Gnr	Copy of value from request (<i>A-Init-Transfer</i> of Phase 1)
Transmod	= <i>initiated</i>
Result	If no error detected by the INITIATOR UE: = <i>result-ok</i> . If error detected by the INITIATOR UE: Other value. See "Error handling" (7.1.3.2).

7.1.3.2. Error handling

7.1.3.2.1. FU disruption

Disruption by the *Permanent Association FU* or the *Dynamic Association FU*:

Disruption of both the INITIATOR part and the RESPONDER part of the current invocation of the *Requested Data Transfer FU* shall be triggered locally, as a part of the handling of incoming *A-P-Abort ind.* primitives in both the INITIATOR part and the RESPONDER part of the FU invocation handling the association on which the *Requested Data Transfer FU* is running.

Both parts of the current invocation of the *Requested Data Transfer FU* shall be terminated gracefully, neither part attempting to issue any primitive associated with the termination itself.¹⁰

7.1.3.2.2. Illegal invocation attempt

FU not present:

If the *Requested Data Transfer FU* is not present in an INITIATOR UE:

Invocation requests are always generated locally; see chapter 7.1.2.3 "Invocation", above. Consequently, the handling of this type of error is a local issue, outside the scope of this document.

If the *Requested Data Transfer FU* is not present in an RESPONDER UE:

The RESPONDER User Element shall respond to activation attempts in one of two ways:
Either:

- Ignoring the incoming *A-Init-Transfer ind.* primitive altogether
- or:
- Issuing an *A-Data req.* primitive with Result = *remote-service-user-unavailable*

¹⁰Local clean-up procedures are not specified by this document.

FU present, but attempt illegal:

Attempts of illegal invocations of the *Requested Data Transfer FU* for any given group in an INITIATOR UE is a local issue, outside the scope of this document.

Attempts of illegal invocations of the *Requested Data Transfer FU* for any given group in a RESPONDER UE shall be handled by the RESPONDER UE in one of two ways:

Either:

- Ignoring the incoming *A-Init-Transfer ind.* primitive altogether

or:

- Issuing an *A-Data req.* primitive with Result = *remote-service-user-unavailable* without actually (re-)invoking the *Requested Data Transfer FU* for the group concerned, in the RESPONDER UE.

7.1.3.2.3. Incoming EASE service primitive out of context

The *A-Data (init) ind.* service primitive is never out of context in the INITIATOR UE. Likewise, in the RESPONDER UE the *A-Init-Transfer ind.* and the *A-Conf-Data (init) ind.* service primitives are never out of context.

7.1.3.2.4. Timing errors

Error in INITIATOR part:

Error	Reaction from EASE	Specified action in FU
UE too late issuing <i>A-Conf-Data (init.) req.</i> after receiving <i>A-Data (init., F) ind.</i> :	<p>In INITIATOR: Local error from eventual attempt at issuing <i>A-Conf-Data (init.) req.</i></p> <p>In RESPONDER: <i>A-Conf-Data (init.) ind.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>INITIATOR part: Ignore, or local error indication/logging, then proceed as normal.</p> <p>RESPONDER part: Terminate FU invocation locally, then: local error indication/logging</p>

Error in RESPONDER part:

Error	Reaction from EASE	Specified action in FU
UE too late responding to <i>A-Init-Transfer ind.</i>	<p>In RESPONDER: <i>A-Conf-Data (init.) ind.</i>, with Result = <i>misbehaviour-of-local-service-user</i>.</p> <p>In INITIATOR: <i>A-Data (init.) ind.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Terminate FU invocation locally, then: local error indication/logging</p> <p>INITIATOR part: Ignore, or local error indication/logging, then proceed as normal.</p>
UE too late issuing next <i>A-Data (init.) req.</i> after latest <i>A-Data (init., T) req.</i> issued	<p>In RESPONDER: <i>A-Conf-Data (init.) ind.</i>, with Result = <i>misbehaviour-of-local-service-user</i>.</p> <p>In INITIATOR: <i>A-Data (init.) ind.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Terminate FU invocation locally.</p> <p>INITIATOR part: Terminate FU invocation locally.</p>

7.1.3.2.5. Congestion error

INITIATOR part of the FU:

When occurring with an *A-Init-Transfer req.* attempt:
 Terminate FU invocation locally.

When occurring with an *A-Conf-Data (init) req.* attempt:
 Terminate FU invocation locally.

RESPONDER part of the FU:

When occurring with an *A-Data req.* attempt:
 Terminate FU invocation locally. (INITIATOR part of FU will eventually be terminated upon
 A-Conf-Data (init) ind. with Result $\gt\lt$ *result-ok*, after time-out in the Elcom provider.)

7.1.3.2.6. EASE service primitive parameter errors

Errors in the *A-Init-Transfer ind.* primitive (detected by the RESPONDER part):

The RESPONDER part shall in all these cases issue one *A-Data req.* with Result = <Value from table below>, Transmod = *initiated*, More-D = *false*, empty Data, and the values of Gnr, Gtype, Index1 and Index2 as in the *A-Init-Transfer ind.*
The RESPONDER part of the FU must not be running.

Error	Value of parameter Result in <i>A-Data req.</i>
Gtype value not equal to value of attribute Group type in CS(R) for group no. Gnr. (The case of non-existing group no. Gnr is considered below.)	<i>gtype-out-of-range</i>
Value of Gnr is illegal, or group no. Gnr does not exist in the CS(R).	<i>gnr-out-of-range</i>
Index1 = 0 and Index2 >< 0, or: Index2 = 0 and Index1 >< 0, or: Index1 < 0 or Index2 < 0	<i>index-out-of-range</i>
Index1 >< 0 and Index2 >< 0 and Index2 < Index1	<i>index-out-of-range</i>
Index2 >number of objects configured	<i>index-out-of-range</i>
Index1 = 0 and Index2 = 0, and group created, but not defined (empty group).	<i>index-out-of-range</i>
Format error in T0	<i>T0-out-of-range</i>
None of the requested data incarnations exist in the RESPONDER system	<i>T0-out-of-range</i>
Dt and T-Unit specify a period unequal than the archive period in the RESPONDER system	<i>Dt-out-of-range</i>
Data cannot be returned as specified, because of local resource limitations, or any other reason than those listed above	<i>remote-service-user-unavailable</i>

Errors in the **A-Conf-Data (init) ind.** primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Gtype value different from the value in CS(R) for the group associated with the current FU invocation	Error indication/logging, then terminate FU invocation, as normal
Result >< <i>result-ok</i>	Error indication/logging, then terminate FU invocation, as normal
For security class 2: The received authentication code >< the generated authentication code based in the received data.	<i>invalid-authentication-code-received</i>
The security class 3: The received checksum >< the generated checksum during the decipherment.	<i>decipherment-error</i>

Errors in the **A-Data (init.) ind.** primitive (detected by the INITIATOR part):

The INITIATOR part shall in all cases, except for the case of syntax error in data parameter:

1. Terminate the current FU invocation in the RESPONDER by issuing an *A-Conf-Data (init) req.* with Result = <Value from table below>, Transmod = *initiated*, and the values of Gnr and Gtype as in the *A-Data (init.) ind.* The data shall be ignored.
2. Terminate the INITIATOR part of the FU itself, optionally incrementing local error count and/or reporting or logging the error.

In the case of syntax error in Data parameter, "*invalid-authentication-code-received*" or "*decipherment-error*"¹¹, the INITIATOR part shall ignore only the invalid data, but otherwise proceed as normal, optionally incrementing local error count and/or reporting or logging the error.

Error	Value of parameter Result in <i>A-Conf-Data (init) req.</i>
Result = <i>result-ok</i> , and mismatch between Gnr in primitive and Gnr in corresponding request (<i>A-Init-Transfer req.</i>)	<i>gnr-out-of-range</i>
Result = <i>result-ok</i> , and mismatch between Gtype in primitive and Gtype in corresponding request (<i>A-Init-Transfer req.</i>)	<i>gtype-out-of-range</i>
Result = <i>result-ok</i> , and Index1 < 1 or Index2 < Index1 in corresponding request (<i>A-Init-Transfer req.</i>)	<i>index-out-of-range</i>

¹¹See Appendix A in this document for syntax definition.

Error	Value of parameter Result in <i>A-Conf-Data (init) req.</i>
Result = <i>result-ok</i> , and mismatch between Length and Index1/Index2	<i>index-out-of-range</i>
Result = <i>result-ok</i> , More=false, and requested number of values >< returned number of values	<i>index-out-of-range</i>
Result = <i>result-ok</i> , and index2 > number of objects in CS(I)	<i>index-out-of-range</i>
Result = <i>result-ok</i> , and format error in T	<i>T-out-of-range</i>
Result >< <i>result-ok</i>	Same value as received parameter Result
Invalid Data (see appendix A for validity conditions).	Ignore the data, but otherwise proceed as normal, optionally incrementing local error count.
<p>The security class 2: The received authentication code >< the generated authentication code based in the received data.</p> <p>For security class 3: The received checksum >< the generated checksum during the decipherment.</p>	Ignore the data, but otherwise proceed as normal, optionally incrementing local error count.
	Ignore the data, but otherwise proceed as normal, optionally incrementing local error count.

7.2. Periodically Requested Data Transfer FU

Type: Composite.

7.2.1. Function

A *Periodically Requested Data Transfer FU* invocation adheres to the following procedure, for a single specified group:

1. The INITIATOR UE invokes the *Requested Data Transfer FU* for the group, in order to fetch the specified set of data invocations from the RESPONDER system once.
2. The INITIATOR UE waits for a locally determined amount of time, and then repeats step 1, with exactly the same parameters.

The repetitions continue until the *Periodically Requested Data Transfer FU* invocation is terminated.

7.2.2. Coordination rules

7.2.2.1. Association usage

The Elcom interactions that are part of the *Requested Data Transfer FU* invocations invoked by the *Periodically Requested Data Transfer FU* are conveyed by associations with the characteristics as specified for the *Requested Data Transfer FU*.

7.2.2.2. Relation to other FUs

7.2.2.2.1. Invoking FUs

The *Periodically Requested Data Transfer FU* shall not be invoked by another FU.

7.2.2.2.2. Invoked FUs

The *Periodically Requested Data Transfer FU* may invoke the following FU:
- *Requested Data Transfer FU*

7.2.2.2.3. Disrupting FUs

The *Periodically Requested Data Transfer FU* may be disrupted by disruption of a supporting FU invocation.

7.2.2.2.4. Disrupted FUs

The *Periodically Requested Data Transfer FU* shall not disrupt any other FU.

The FU's invoked by the *Periodically Requested Data Transfer FU* may, however, disrupt any other FU.

7.2.2.3. Invocation

7.2.2.3.1. Prerequisites

No previous or concurrent FU invocations are required¹².

7.2.2.3.2. Restrictions

A *Periodically Requested Data Transfer FU* invocation as such is allowed at any time. However, it will fail if violating rules stated in the "Restrictions" chapter of the descriptions of the FU that the *Periodically Requested Data Transfer FU* invokes: See chapter 7.2.2.2.2 "Invoked FUs", above.

7.2.2.3.3. Invoking events

The INITIATOR part of the *Periodically Requested Data Transfer FU* may be invoked by:

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

The *Periodically Requested Data Transfer FU* is of the composite type. Consequently, there exists no specific RESPONDER part of the *Periodically Requested Data Transfer FU*, with an associated specific invoking event.

¹²*Permanent Association FU* or *Dynamic Association FU* invocations are a requirement of the *Requested Data Transfer FU* invoked by the *Periodically Requested Data Transfer FU*, not of the *Periodically Requested Data Transfer FU* itself.

7.2.2.4. Termination

7.2.2.4.1. Orderly termination

Orderly termination of the *Periodically Requested Data Transfer FU* shall take place after orderly termination of the last FU invocation invoked by the *Periodically Requested Data Transfer FU*, at a point in time determined by a termination request from the local Coordinating Function.

Unexpected orderly termination of any FU that the *Periodically Requested Data Transfer FU* invokes is considered an error within the *Periodically Requested Data Transfer FU* invocation. It may or may not lead to orderly termination for the *Periodically Requested Data Transfer FU*, subject to local decision within the INITIATOR UE.

7.2.2.4.2. Disruption

The INITIATOR part of a *Periodically Requested Data Transfer FU* invocation may be disrupted by:

- Disruption of another FU invocation. See chapter 7.2.2.2.3 "Disrupting FUs", above.

The *Periodically Requested Data Transfer FU* is of the composite type. Consequently, there exists no specific RESPONDER part of the *Periodically Requested Data Transfer FU*, to be disrupted.

7.2.3. Procedures

7.2.3.1. EASE service primitives

The *Periodically Requested Data Transfer FU* is of the composite type, having no specific EASE service primitive sequence associated with it.

7.2.3.1.1. Action sequence

The normal action sequence has 2 phases that are repeated until the FU is terminated by the INITIATOR UE:

Phase 1: Fetching the specified data from RESPONDER system.

If the prerequisites and restrictions that apply to a *Requested Data Transfer FU* invocation for the group can be met:

The INITIATOR UE invokes the *Requested Data Transfer FU* for the group, with specified parameters defining the desired data incarnations per *Requested Data Transfer FU* invocation.

If the *Requested Data Transfer FU* is NOT running OK or the *Requested Data Transfer FU* invocation does NOT complete OK:

The *Periodically Requested Data Transfer FU* invocation terminates, with error indication, if the INITIATOR UE finds this appropriate¹³.

If not:

The *Periodically Requested Data Transfer FU* invocation terminates, with error indication, if the INITIATOR UE finds this appropriate.

Phase 2: Pause.

The pause length is entirely determined by the INITIATOR UE. After pause expiry, the sequence is restarted at Phase 1.

7.2.3.1.2. Parameter values

The FU is of the composite type. See description of component FUs.

7.2.3.2. Error handling

Handling of errors occurring with EASE service primitives are not considered here, since the *Periodically Requested Data Transfer FU* is of the composite type.

Errors occurring in FUs invoked by the *Periodically Requested Data Transfer FU* are described in chapter 7.2.3.1.1 "Action sequence", above. Further processing of such errors is a local matter in the INITIATOR UE, not specified by this document.

¹³For example, because of an accumulated error count within the FU invocation becoming too great.

7.2.3.2.1. FU disruption

Disruption of the *Periodically Requested Data Transfer FU* shall be further processed by the INITIATOR UE as for other *Periodically Requested Data Transfer FU* errors; see chapter 7.2.3.2 "Error handling", above.

7.2.3.2.2. Illegal invocation attempt

FU not present:

If the *Periodically Requested Data Transfer FU* is not present in an INITIATOR UE:

Invocation requests are always generated locally; see chapter 7.2.2.3 "Invocation", above. Consequently, the handling of this type of error is a local issue, outside the scope of this document.

No specific RESPONDER part is defined for the *Periodically Requested Data Transfer FU*; since it is of the composite type.

FU present, but attempt illegal:

Periodically Requested Data Transfer FU invocation attempts as such are always legal. See chapter 7.2.2.3.2 "Restrictions", above.

7.3. Periodic Data Transfer FU

Type: Primary.

7.3.1. Function

A *Periodic Data Transfer FU* invocation adheres to the following procedure:

1. The INITIATOR UE grants the RESPONDER UE permission to transmit data periodically, subject to the following general rules:
 - The data are associated with one specified Elcom group only.
 - The sequence of the data values in any transmission is the same as in the group definition.
 - The sequence of the data shall, if possible, comprise the whole group. If this is not possible due to the size of the group, data subsets that together comprise the whole group shall be transmitted individually.
 - The RESPONDER UE determines when to transmit, for each transmission, based on a local best-fit to the period defined by a timer for the group. If data subsets are necessary (above), the timer shall define the transmission instant of the first subset. The rest of the subsets shall then follow in sequence, with minimal delay between subsets.
 - For groups of type *Text-message-group*, each transmission shall contain the data value of exactly one object. In other words, for groups of this type each object shall constitute a data subset.
 - One single incarnation of the data value of any one object shall always be contained within a single transmission.¹⁴
 - The RESPONDER UE may transmit any number of times, until the INITIATOR UE withdraws the permission to send.
 - The RESPONDER UE shall specify **non**-acknowledged operation with each transmission.
2. The RESPONDER UE transmits data periodically, a number of times.
3. The INITIATOR UE withdraws the permission to send.
4. The RESPONDER UE stops the periodical transmissions.

¹⁴The limiting factor is the maximum number of user octets in a A-PDU. An A-PDU normally carries 256 octets. The number of octets not available for data values is given by:

P-layer overhead:	1 octet
Non-Data value part of D-T-REQ PDU:	19 octets

Sum:	20 octets
------	-----------

Thus, the set of **Data** for the subgroup, coded as defined in [8] section 5.8.3, normally must be accommodated within $256-20 = 236$ octets.

7.3.2. Coordination rules

7.3.2.1. Association usage

All Elcom interactions that are part of one invocation of the *Periodic Data Transfer FU* are conveyed by one single association. The association shall be permanent, and have the characteristics as specified in the chapter 7.3.2.3.1 "Prerequisites", below.

7.3.2.2. Relation to other FUs

7.3.2.2.1. Invoking FUs

The *Periodic Data Transfer FU* may be invoked by the *Restart Reactivate FU*.

7.3.2.2.2. Invoked FUs

The *Periodic Data Transfer FU* may invoke the *Unsolicited Mixed Data Transfer FU*. Any such invocation is the responsibility of the RESPONDER UE.

7.3.2.2.3. Disrupting FUs

The *Periodic Data Transfer FU* may be disrupted by:

- *Permanent Association FU*
- *Group Management FU*
- *Group Definition FU*
- *Restart Reconfigure FU*

7.3.2.2.4. Disrupted FUs

The *Periodic Data Transfer FU* shall not disrupt any other FU.

7.3.2.3. Invocation

7.3.2.3.1. Prerequisites

The following FUs shall have been invoked preceding any invocation of the *Periodic Data Transfer FU*:

- The *Permanent Association FU*
- The *Group Configuration FU*¹⁵

The *Permanent Association FU* invocation shall still be running at the time of invocation of the *Periodic Data Transfer FU*, and the association maintained by the *Permanent Association FU* invocation shall be running (not temporarily broken) at that time.

The *Permanent 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 *Periodic Data Transfer FU*, with the following characteristics:

- A-suffix pair as specified in the table in chapter 5.1 "Addressing" in section "THE ASSOCIATION MANAGEMENT (A) FUNCTION GROUP".
- Spontaneous mode code as specified in the table in chapter 5.3 "Spontaneous mode codes" in section "THE ASSOCIATION MANAGEMENT (A) FUNCTION GROUP".

The *Group Configuration FU* shall have been invoked in order to define and configure the group that is to be transmitted, prior to the current invocation of the *Periodic Data Transfer FU*.¹⁶

The *Group Configuration FU* invocation of that group must not be running at the time of invocation of the *Periodic Data Transfer FU*.

7.3.2.3.2. Restrictions

For any given INITIATOR/RESPONDER system combination, multiple simultaneous invocations of the *Periodic Data Transfer FU* for any given group are not allowed on the same association.

The *Periodic Data Transfer FU* must not be invoked while at least one of the following FUs are running, for the group involved:

- *Group Configuration FU*
- *Group Management FU*
- *Group Definition FU*

Neither shall the *Periodic Data Transfer FU* be running on a given association, if the *Requested Data Transfer FU* is currently running on the same association.

¹⁵Alternatively, the *Group Management FU* and *Group Definition FU*, which are invoked by the *Group Configuration FU*, may have been invoked directly by the Coordinating Function.

¹⁶The use of the *Group Configuration FU* or *Group Management FU* and *Group Definition FU* is, however, not required if the group is defined in the RESPONDER AP by a non-Elcom mechanism and also made known to the INITIATOR AP by a non-Elcom mechanism.

7.3.2.3.3. Invoking events

The INITIATOR part of the *Periodic Data Transfer FU* may be invoked by:

- Local request via the Coordinating Function, the original source of which is outside the scope of this document.
- The *Restart Reactivate FU*.

Invocation of the RESPONDER part of the *Periodic Data Transfer FU* is attempted whenever a valid *A-Spont-Mgmt (start) ind.* primitive is received via an association with the characteristics as defined for the *Periodic Data Transfer FU*: See description of the *Permanent Association FU*.

7.3.2.4. Termination

7.3.2.4.1. Orderly termination

Orderly termination of the INITIATOR part of a *Periodic Data Transfer FU* invocation may only be triggered by:

- Local request via the Coordinating Function, the original source of which is outside the scope of this document.
- Local accumulated error count becoming too great. See "Error handling", below, on timing errors.
- Congestion error. See relevant chapter, below.

Orderly termination of the RESPONDER part of a *Periodic Data Transfer FU* invocation may only be triggered by reception of a valid *A-Spont-Mgmt (stop) ind.* primitive.

7.3.2.4.2. Disruption

Both the INITIATOR and the RESPONDER part of a *Periodic Data Transfer FU* invocation may be disrupted by:

- Disruption of another FU invocation. See chapter 7.3.2.2.3 "Disrupting FUs", above.
- A fatal error condition encountered during operation. See chapter 7.3.3.2 "Error handling", below.

7.3.3. Procedures

7.3.3.1. EASE service primitives

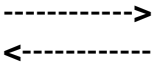
The following elementary EASE services are used by the *Periodic Data Transfer FU*:

- *A-Spont-Mgnt*
- *A-Data (spont.)*
- *A-Conf-Data (spont.)* (Only in case of error)

7.3.3.1.1. Sequence

The normal sequence of primitives is partitioned into **3** phases:

Phase 1: Granting permission to send.

INITIATOR UE	EASE	RESPONDER UE
<i>A-Spont-Mgnt (start) req.</i> <i>A-Spont-Mgnt (start) cnf.</i>	<div style="text-align: center;">  </div>	<i>A-Spont-Mgnt (start) ind.</i> <i>A-Spont-Mgnt (start) res.</i>

Phase 2: Data transmission.¹⁷

INITIATOR UE	EASE	RESPONDER UE
<i>A-Data (spont., T) ind.</i>	←----- · ·	<i>A-Data (spont., T) req</i>
<i>A-Data (spont., T) ind.</i>	←----- · ·	<i>A-Data (spont., T) req.</i>

In this phase, the following rules apply:

1. The period length, or time span between successive *A-Data (spont., T) ind.* primitives, shall be kept constant by the RESPONDER during the lifetime of the FU invocation. The RESPONDER UE shall determine period length based on the value of attribute **Priority class** for the group¹⁸ for which the FU is invoked:
 - If **Priority class** = 0, the RESPONDER UE shall determine period length exclusively based on local rules, outside the scope of this document.
 - For all other values of **Priority class**, the value shall be interpreted in the RESPONDER UE as a request for a period length defined by an entry in a RESPONDER specific table of supported period lengths. This table is implementation dependent, but shall conform to the following 3 rules:
 1. Moving towards smaller **Priority class** values shall correspond to moving to shorter period lengths.
 2. **Priority class** value 1 shall correspond to the shortest period length that the RESPONDER UE implementation is able to support.
 3. **Priority class** value 15 shall correspond a period length of 50 seconds.

The number of entries in the period length table may be less than 15, effectively collapsing a number of **Priority class** values into one. However, the table shall contain at least two entries, if the mechanism is at all supported by a given RESPONDER UE. These two entries shall then correspond to values 1 and 15, respectively.

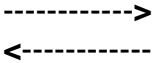
2. The EASE imposes a limit on the time allowed between any *A-Data (spont., T)* primitive and the succeeding *A-Data (spont., T)* primitive¹⁹. If too much time elapses between these primitives, an error situation occurs in the EASE. See the chapter 7.3.3.2 "Error handling", below.
3. The rules stated in chapter 7.3.1 "Function", above, also apply here.

¹⁷Notation: *T* and *F* signifies, for the parameter More-D, the values *true* and *false*, respectively.

¹⁸If data belong to more than one group, each occurrence of the data in the outgoing queue shall be handled independently, according to their individual **Priority Class** values.

¹⁹Defined by the timer DCUT1 within the Elcom provider; see [10], chapter 4 and appendix C.2.5. (Default value: Normally 60 seconds).

Phase 3: Orderly termination: Withdrawing permission to send.

INITIATOR UE	EASE	RESPONDER UE
<i>A-Spont-Mgmt (stop) req.</i> <i>A-Spont-Mgmt (stop) cnf.</i>		<i>A-Spont-Mgmt (stop) ind.</i> <i>A-Spont-Mgmt (stop) res.</i>

The following rules apply:

1. The transition from phase 2 to phase 3 always following an *A-Data (spont., T)* primitive, the error condition described in rule 2 of phase 2 above will eventually occur. However, this error shall be considered a part of normal operation, and be suppressed by the INITIATOR UE. See "Error handling", below.
2. Data transmissions shall always be non-acknowledged (parameter More-D = T). However, the INITIATOR UE shall be prepared to issue an *A-Conf-Data (spont) req.* service primitive with error code, upon reception of any *A-Data (spont, F)* while waiting for next *A-Data (spont, T)*. See "Error handling", below.
3. After a *Periodic Data Transfer FU* invocation for a given group has been terminated, the RESPONDER UE must not try to invoke the *Unsolicited Mixed Data Transfer FU* for that group.

7.3.3.1.2. Parameter values

A-Spont-Mgmt :

Parameter	req. (INITIATOR)	res. (RESPONDER)
Function	Phase 1: = <i>start</i> Phase 3: = <i>stop</i>	Copy of value from <i>ind.</i>
Gtype	Value <i>Measure-group, Status-group, Discrete-group, Logical-breaker status-group, or Text-message-group</i> , depending on the type of the group in question, as defined in the CS.	Copy of value from <i>ind.</i>
Gnr	Reference number for the group in question, as defined in the CS.	Copy of value from <i>ind.</i>
Result	(Not applicable)	Action performed as specified. = <i>result-ok</i> Action not performed, due to error condition: Other value. See chapter 7.3.3.2 "Error handling". (If Function = <i>start</i> , this means that the RESPONDER part of the <i>Periodic Data Transfer FU</i> has not been invoked, for the group in question).

A-Data (spont., T) req. (RESPONDER):

Parameter	req.
Gtype	Copy of value from <i>A-Spont-Mgmt (start) ind.</i> of Phase 1
Gnr	Copy of value from <i>A-Spont-Mgmt (start) ind.</i> of Phase 1
Transmod	= <i>spontaneous</i>
Index1 ²⁰	Shall be > 0: Equal to the lowest Object number , in accordance with the CS(R), of the range of Object numbers whose data values are contained in the Data parameter (below). Shall be equal to Index2, if Gtype = <i>Text-message-group</i> . ²¹
Index2	Shall be >= value of Index1: Equal to the highest Object number , in accordance with the CS(R), of the range of Object numbers whose data values are contained in the Data parameter (below). Shall be equal to Index1, if Gtype = <i>Text-message-group</i> .
T	Time stamp, applying to the whole set of values contained in the Data parameter (below), and determined by the RESPONDER UE. The use of UTC is recommended, please refer Appendix F for ELCOM-83 compatibility.
More-D	= <i>true</i>
Data	The actual Elcom data transferred. The structure is defined in Appendix A of this document
Length	Length of Data in octets. Shall be computed according to the length of the actual datatype, Index1 and Index2.
Result	= <i>result-ok</i>

A-Conf-Data (spont.) req. (INITIATOR; in case of error, only):

Parameter	req.
Gtype	Copy of value from <i>A-Spont-Mgmt (start) ind.</i> of Phase 1
Gnr	Copy of value from <i>A-Spont-Mgmt (start) ind.</i> of Phase 1
Transmod	= <i>spontaneous</i>
Result	Error code, >< <i>result-ok</i> . See "Error handling" (below).

²⁰The number of values that shall be contained in the Data parameter can be computed as: Index2 - Index1 + 1.

²¹Only one object per transmission, for such groups.

7.3.3.2. Error handling

7.3.3.2.1. FU disruption

Disruption by the *Permanent Association FU*:

Disruption of both the INITIATOR part and the RESPONDER part of the current invocation of the *Periodic Data Transfer FU* shall be triggered locally, as a part of the handling of incoming *A-P-Abort ind.* primitives in both the INITIATOR part and the RESPONDER part of the *Permanent Association FU* invocation handling the association over which the *Periodic Data Transfer FU* is running.

Both parts of the current invocation of the *Periodic Data Transfer FU* shall be terminated gracefully, neither part attempting to issue any primitive associated with the termination itself.²²

Following the termination, the *Permanent Association FU* will enter a state in which it will signal *Restart, spontaneous management lost* or *Restart, group management lost*, the next time an association with the characteristics for Periodic Data Transfer is established between the same INITIATOR and RESPONDER UE pair.²³

7.3.3.2.2. Illegal invocation attempt

FU not present:

If the *Periodic Data Transfer FU* is not present in an INITIATOR UE:

Invocation requests are always generated locally; see chapter 7.3.2.3 "Invocation", above. Consequently, the handling of this type of error is a local issue, outside the scope of this document.

If the *Periodic Data Transfer FU* is not present in an RESPONDER UE:

The RESPONDER User Element shall respond to activation attempts in one of two ways:

Either:

- Ignoring the incoming *A-Spont-Mgmt (start) ind.* primitive altogether

or:

- Issuing an *A-Spont-Mgmt (start) res.* primitive with Result = *remote-service-user-unavailable*

²²Local clean-up procedures are not specified by this document.

²³The RESPONDER UE may be programmed to always signal the restart code "*Restart, spontaneous management lost*" during creation of associations for Periodic Data Transfer, regardless of previous history.

FU present, but attempt illegal:

Attempts of illegal invocations of the *Periodic Data Transfer FU* for any given group in an INITIATOR UE is a local issue, outside the scope of this document.

Attempts of illegal invocations of the *Periodic Data Transfer FU* for any given group in a RESPONDER UE shall be handled by the RESPONDER UE in one of two ways:

Either:

- Ignoring the incoming *A-Spont-Mgmt (start) ind.* primitive altogether

or:

Issuing an *A-Spont-Mgmt (start) res.* primitive with Result = *remote-service-user-unavailable* without actually (re-)invoking the *Periodic Data Transfer FU* for the group concerned, in the RESPONDER UE.

7.3.3.2.3. Incoming EASE service primitive out of context

INITIATOR part:

State	<i>A-Spont-Mgmt (start) cnf.</i>	<i>A-Spont-Mgmt (stop) cnf.</i>	<i>A-Data (spont., T) ind.</i>
FU not running	Ignore, or local error indication/logging	Ignore, or local error indication/logging	Ignore, or: Simulate the entry of Phase 3 of a <i>Periodic Data Transfer FU</i> invocation for the group in question, thus terminating the (supposed) current invocation of the <i>Periodic Data Transfer FU</i> in the RESPONDER UE., or: Issue <i>A-Conf-Data (spont) req.</i> , with Result = <i>spontaneous-transfer-not-initiated</i> .
FU running, waiting for <i>A-Spont-Mgmt (start) cnf.</i>	(Normal)	(parameter error)	Ignore, or local error indication/logging
Waiting for <i>A-Data (spont., T) ind.</i>	Ignore, or local error indication/logging	Ignore, or local error indication/logging	(Normal)
Waiting for <i>A-Spont-Mgmt (stop) cnf.</i>	(parameter error)	(Normal)	Ignore, or local error indication/logging

RESPONDER part:

State	<i>A-Spont-Mgmt (start) ind.</i>	<i>A-Spont-Mgmt (stop) ind.</i>	<i>A-Conf-Data (spont) ind.</i>
FU not running	(Normal)	Ignore, or: Issue an <i>A-Spont-Mgmt (stop) res.</i> primitive with Result = <i>result-ok</i> , without terminating any <i>Periodic Data Transfer FU</i> invocation in the RESPONDER UE.	Always an error: Ignore, or decode, then report/log.
FU running, waiting for DATA TIMER ²⁴ expiry	Illegal invocation attempt; see relevant chapter.	(Normal) If Gnr is not active for spont. transfer, issue an <i>A-Spont-Mgmt (stop) res.</i> Primitive with Result = <i>spontaneous-transfer-not-initiated</i>	Always an error: Ignore, or decode, then report/log.

7.3.3.2.4. Timing errors

Error in INITIATOR part:

Error	Reaction from EASE	Specified action in FU
UE too late issuing <i>A-Conf-Data (spont.) req.</i> after receiving <i>A-Data (spont., F) ind.</i> :	In INITIATOR: Local error from eventual attempt at issuing <i>A-Conf-Data (spont) req.</i> In RESPONDER: <i>A-Conf-Data (spont) ind.</i> , with Result = <i>remote-service-user-unavailable</i> .	INITIATOR part: Ignore RESPONDER part: Local issue, since the issuing of an <i>A-Data (spont., F) req.</i> was an error in the first place.

²⁴The local timer governing the periodicity of the current *Periodic data Transfer FU* invocation.

Error in RESPONDER part:

Error	Reaction from EASE	Specified action in FU
<p>UE too late responding to <i>A-Spont-Mgmt (start) ind.</i></p>	<p>In RESPONDER: Local error from eventual attempt at issuing <i>A-Spont-Mgmt (start) res.</i></p> <p>In INITIATOR: <i>A-Spont-Mgmt (start) cnf.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Terminate FU invocation locally.</p> <p>INITIATOR part: Terminate FU invocation locally.</p>
<p>UE too late issuing next <i>A-Data (spont., T) req.</i> after latest <i>A-Data (spont., T) req.</i> issued</p>	<p>In RESPONDER: <i>A-Conf-Data (spont) ind.</i>, with Result = <i>misbehaviour-of-local-service-user</i>.</p> <p>In INITIATOR: <i>A-Data (spont.) ind.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Ignore, or local error indication/logging</p> <p>INITIATOR part: If occurring during phase 3 of the service primitive sequence (after issuing <i>A-Spont-Mgmt (stop) req.</i>): Ignore. If not: Local error indication/logging, then proceed as normal. Optionally: If accumulated error count during current FU invocation becomes too great (by local decision) Enter Phase 3 (Orderly Termination) of the FU, issuing <i>A-Spont-Mgmt (stop) req.</i>, with Result = <i>misbehaviour-of-local-service-user</i></p>
<p>UE too late responding to <i>A-Spont-Mgmt (stop) ind.</i></p>	<p>In RESPONDER: Local error from eventual attempt at issuing <i>A-Spont-Mgmt (stop) res.</i></p> <p>In INITIATOR: <i>A-Spont-Mgmt (stop) cnf.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Terminate FU invocation locally.</p> <p>INITIATOR part: Terminate FU invocation locally.</p>

7.3.3.2.5. Congestion error

INITIATOR part of the FU:

When occurring with an *A-Spont-Mgmt (start) req.* attempt:
Terminate FU invocation locally.

When occurring with an *A-Conf-Data (spont) req.* attempt:
Ignore.

When occurring with an *A-Spont-Mgmt (stop) req.* attempt:
Ignore, or notify operator

RESPONDER part of the FU:

When occurring with an *A-Spont-Mgmt (start) res.* attempt:
Terminate FU invocation locally. (INITIATOR part of FU will eventually be terminated after time-out in the Elcom provider.)

When occurring with an *A-Data req.* attempt:
Report the error locally, then abandon the attempt, reinitialising the DATA TIMER as normal.

When occurring with an *A-Spont-Mgmt (stop) res.* attempt:
Terminate FU invocation locally. (INITIATOR part of FU will eventually be terminated after time-out in the Elcom provider.)

7.3.3.2.6. EASE service primitive parameter errors

For all primitives except the *A-Spont-Mgmt (start) ind.* primitive, the parameter Gnr will always be valid within the context of this FU, *provided the FU is running*²⁵ for that group: It serves as identification of the FU invocation to which the incoming primitive shall be directed.

Errors in the *A-Spont-Mgmt (start) ind.* primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Gtype value not equal to value of attribute Group type in CS(R) for group no. Gnr. (The case of non-existing group no. Gnr is considered below.)	Issue <i>A-Spont-Mgmt (start) res.</i> with Result = <i>gtype-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not invoke the FU.
Value of Gnr is illegal, or group no. Gnr does not exist in the CS(R).	Issue <i>A-Spont-Mgmt (start) res.</i> with Result = <i>gnr-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not invoke the FU.
Value of Gtype is illegal.	Issue <i>A-Spont-Mgmt (start) res.</i> with Result = <i>gtype-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not invoke the FU.
Group is created, but not defined	Issue <i>A-Spont-Mgmt (start) res.</i> with Result = <i>index-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not invoke the FU.

Errors in the *A-Spont-Mgmt (stop) ind.* primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Gtype value different from the value in CS(R) for the group associated with the current FU invocation	Issue <i>A-Spont-Mgmt (stop) res.</i> with Result = <i>gtype-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not terminate the FU.

²⁵The case of the FU **not** being invoked for the group shall be handled outside the FU (by the Coordinating Function): The natural replying primitive shall be issued, with parameter Result = *gnr-out-of-range*.

Errors in the **A-Spont-Mgmt (start) cnf.** primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
<p>Mismatch between Gnr/Gtype/Function in primitive and Gnr/Gtype/Function in corresponding <i>A-Spont-Mgmt (start) req.</i>, and Result = <i>result-ok</i>.</p> <p>Result >< <i>result-ok</i></p>	<p>Trigger Orderly Termination of the current FU invocation, issuing <i>A-Spont-Mgmt (stop) req.</i> for the group in question, with value of Gnr/Gtype/Function as in the corresponding <i>A-Spont-Mgmt (start) req.</i></p> <p>Terminate the FU invocation locally</p>

Errors in the **A-Data (spont.) ind.** primitive (detected by the INITIATOR part):

The INITIATOR may supervise that all values are received within a certain time limit. This is a local matter in the INITIATOR.

Error	Action in INITIATOR part of FU
<p>More-D = F</p> <p>Mismatch between Gtype in primitive and Gtype in corresponding <i>A-Spont-Mgmt (start) req.</i></p> <p>Invalid Index1 or Index2 (Validity conditions are given above, in chapter 7.3.3.1.2 "Parameter values".)</p> <p>Invalid Data (See Appendix A in this document for validity conditions)</p> <p>Result = <i>result-ok</i>, and mismatch between Length and Index1/index2</p> <p>Result = <i>result-ok</i>, and format error in T</p> <p>Result >< <i>result-ok</i></p> <p>The security class 2: The received authentication code >< the generated authentication code based in the received data.</p> <p>For security class 3: The received checksum >< the generated checksum during the decipherment.</p>	<p>Issue <i>A-Conf-Data req.</i> with Result = <i>remote-service-user-unavailable</i>, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count.</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count.</p>

Errors in the ***A-Spont-Mgmt (stop) cnf.*** primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Mismatch between Gtype in primitive and Gtype in corresponding <i>A-Spont-Mgmt (stop) req.</i> , and Result = <i>result-ok</i> .	Trigger abrupt termination of supporting association
Result \neq <i>result-ok</i>	Trigger abrupt termination of supporting association

7.4. Unsolicited Data Transfer FU

Type: Primary.

7.4.1. Function

An *Unsolicited Data Transfer FU* invocation adheres to the following procedure:

1. The INITIATOR UE grants the RESPONDER UE permission to transmit unsolicited data, subject to the following general rules:
 - The data are associated with one specified Elcom group only.
 - The sequence of the data values is the same as in the group definition.
 - The sequence of the data values may be a subset of the complete sequence, and the subset may vary from transmission to transmission.
 - For groups of type *Text-message-group*, each transmission shall contain the data value of exactly one object. In other words, for groups of this type each object shall constitute a data subset.
 - One single incarnation of the data value of any one object shall always be contained within a single transmission.²⁶
 - The RESPONDER UE may transmit any number of times, until the INITIATOR UE withdraws the permission to send.
 - The RESPONDER UE determines when to transmit, for each transmission.
 - The RESPONDER UE must specify acknowledged or non-acknowledged operation, on a per transmission basis.
2. The RESPONDER UE transmits data, a number of times.
3. The INITIATOR UE acknowledges reception of data, whenever such acknowledgement has been specified by the RESPONDER UE.
4. The INITIATOR UE withdraws the permission to send.
5. The RESPONDER UE stops the transmissions.

²⁶The limiting factor is the maximum number of user octets in a A-PDU. An A-PDU normally carries 256 octets. The number of octets not available for data values is given by:

P-layer overhead:	1 octet
Non-Data value part of D-T-REQ PDU:	19 octets

Sum:	<u>20 octets</u>
------	------------------

Thus, the set of **Data** for the subgroup, coded as defined in [8], chapter 5.8.3, normally must be accommodated within 256-20 = **236** octets.

7.4.2. Coordination rules

7.4.2.1. Association usage

All Elcom interactions that are part of one invocation of the *Unsolicited Data Transfer FU* are conveyed by one single association. The association shall be permanent, and have the characteristics as specified in chapter 7.4.2.3.1 "Prerequisites", below.

7.4.2.2. Relation to other FUs

7.4.2.2.1. Invoking FUs

The *Unsolicited Data Transfer FU* may be invoked by the *Restart Reactivate FU*.

7.4.2.2.2. Invoked FUs

The *Unsolicited Data Transfer FU* may invoke the *Unsolicited Mixed Data Transfer FU*. Any such invocation is the responsibility of the RESPONDER UE.

7.4.2.2.3. Disrupting FUs

The *Unsolicited Data Transfer FU* may be disrupted by:

- *Permanent Association FU*
- *Group Management FU*
- *Group Definition FU*
- *Restart Reconfigure FU*

7.4.2.2.4. Disrupted FUs

The *Unsolicited Data Transfer FU* shall not disrupt any other FU.

7.4.2.3. Invocation

7.4.2.3.1. Prerequisites

The following FUs shall have been invoked preceding any invocation of the *Unsolicited Data Transfer FU*:

- The *Permanent Association FU*
- The *Group Configuration FU*²⁷

The *Permanent Association FU* invocation shall still be running at the time of invocation of the *Unsolicited Data Transfer FU*, and the association maintained by the *Permanent Association FU* invocation shall be running (not temporarily broken) at that time.

The *Permanent 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 *Unsolicited Data Transfer FU*, with the following characteristics:

- A-suffix pair as specified in the table in chapter 5.1.2 "A-Suffixes" in section "THE ASSOCIATION MANAGEMENT (A) FUNCTION GROUP".
- Spontaneous mode code as specified in the table in chapter 5.3 "Spontaneous mode codes" in section "THE ASSOCIATION MANAGEMENT (A) FUNCTION GROUP".

The *Group Configuration FU* shall have been invoked in order to define and configure the group that is to be transmitted, prior to the current invocation of the *Unsolicited Data Transfer FU*.²⁸ The *Group Configuration FU* invocation of that group must not be running at the time of invocation of the *Unsolicited Data Transfer FU*.

7.4.2.3.2. Restrictions

For any given INITIATOR/RESPONDER system combination, multiple simultaneous invocations of the *Unsolicited Data Transfer FU* for any given group on the same association are not allowed.

The *Unsolicited Data Transfer FU* must not be running while at least one of the following FUs are running, for the group involved:

- *Group Configuration FU*
- *Group Management FU*
- *Group Definition FU*

²⁷Alternatively, the *Group Management FU* and *Group Definition FU*, which are invoked by the *Group Configuration FU*, may have been invoked directly by the Coordinating Function.

²⁸The use of the *Group Configuration FU* is, however, not required if the group is defined in the RESPONDER AP by a non-Elcom mechanism and also made known to the INITIATOR AP by a non-Elcom mechanism.

7.4.2.3.3. Invoking events

The INITIATOR part of the *Unsolicited Data Transfer FU* may be invoked by:

- Local request via the Coordinating Function, the original source of which is outside the scope of this document.
- The *Restart Reactivate FU*.

Invocation of the RESPONDER part of the *Unsolicited Data Transfer FU* is attempted whenever a valid *A-Spont-Mgmt (start) ind.* primitive is received via an association with the characteristics as defined for the *Unsolicited Data Transfer FU*: See description of the *Permanent Association FU*.

7.4.2.4. Termination

7.4.2.4.1. Orderly termination

Orderly termination of the INITIATOR part of an *Unsolicited Data Transfer FU* invocation may only be triggered by:

- Local request via the Coordinating Function, the original source of which is outside the scope of this document.
- Local accumulated error count becoming too great. See "Error handling", below, on timing errors.
- Congestion error. See relevant chapter, below.

Orderly termination of the RESPONDER part of an *Unsolicited Data Transfer FU* invocation may only be triggered by reception of a valid *A-Spont-Mgmt (stop) ind.* primitive.

7.4.2.4.2. Disruption

Both the INITIATOR and the RESPONDER part of an *Unsolicited Data Transfer FU* invocation may be disrupted by:

- Disruption of another FU invocation. See chapter 7.4.2.2.3 "Disrupting FUs", above.
- A fatal error condition encountered during operation. See chapter 7.4.3.2 "Error handling", below.

7.4.3. Procedures

7.4.3.1. EASE service primitives

The following elementary EASE services are used by the *Unsolicited Data Transfer FU*:

- *A-Spont-Mgmt*
- *A-Data (spont.)*
- *A-Conf-Data (spont.)*

7.4.3.1.1. Sequence

The normal sequence of primitives is partitioned into **3** phases:

Phase 1: Granting permission to send.

INITIATOR UE	EASE	RESPONDER UE
<i>A-Spont-Mgmt (start) req.</i> <i>A-Spont-Mgmt (start) cnf.</i>	-----> <-----	<i>A-Spont-Mgmt (start) ind.</i> <i>A-Spont-Mgmt (start) res.</i>

Phase 2: Data transmission.²⁹

INITIATOR UE	EASE	RESPONDER UE
<i>A-Data (spont., T) ind.</i> <i>A-Data (spont., T) ind.</i>	<----- <----- . .	<i>A-Data (spont., T) req.</i> <i>A-Data (spont., T) req.</i>
<i>A-Data (spont., T) ind.</i> <i>A-Data (spont., T) ind.</i> <i>A-Data (spont., T) ind.</i>	<----- <----- <----- . .	<i>A-Data (spont., T) req.</i> <i>A-Data (spont., T) req.</i> <i>A-Data (spont., T) req.</i>
<i>A-Data (spont.,) ind.</i> <i>A-Conf-Data (spont.) req.</i>	<----- -----> . .	<i>A-Data (spont., F) req.</i> <i>A-Conf-Data (spont.) ind.</i>

²⁹Notation: *T* and *F* signifies, for the parameter More-D, the values *true* and *false*, respectively.

In this phase, the following rules apply:

1. For the current invocation of this FU, the *A-Conf-Data (spont.)* primitive is to be functionally interpreted as acknowledgement of the data carried by all preceding *A-Data (spont.)* primitives since the last *A-Conf-Data (spont.)* primitive.
2. This document places no restriction on the number of consecutive *A-Data (spont., T) req.* primitives that may be issued before an *A-Data (spont., F) req.* is issued. However, if too much time elapses between any *A-Data (spont., T)* primitive and the succeeding *A-Data (spont., T)* or *A-Data (spont., F)* primitive, an error situation occurs in the EASE. See chapter 7.4.3.2 "Error handling", below.
3. The RESPONDER UE is responsible for determining the timing and ordering of the *A-Data (spont.) req.* primitives for the different *Unsolicited Data Transfer FU* invocations. The RESPONDER UE shall order its outgoing data queue according to the attribute **Priority Class** of the group³⁰ to which the data belongs:
 - Data of a given **Priority Class** value may be sent prior to all pending data of greater **Priority Class** value.
 - Data of equal **Priority Class** value may be sent in order of occurrence.
 - **Priority Class** equal to zero shall be interpreted as "priority function off for these data", disabling priority check for the data concerned and always appending these at the end of the outgoing data queue.³¹
4. The RESPONDER UE may at any time choose to report data for any number of groups for which the *Unsolicited Data Transfer FU* is invoked, via a local *Unsolicited Mixed Data Transfer FU* invocation instead of, or in addition to, the normal *A-Data (spont.) req.* primitives. See the *Unsolicited Mixed Data Transfer FU* description.
5. The rules stated in chapter 7.4.1 "Function", above, also apply here.

Phase 3: Orderly termination: Withdrawing permission to send.

INITIATOR UE	EASE	RESPONDER UE
<i>A-Spont-Mgmt (stop) req.</i> <i>A-Spont-Mgmt (stop) cnf.</i>	-----> <-----	<i>A-Spont-Mgmt (stop) ind.</i> <i>A-Spont-Mgmt (stop) res.</i>

The following rules apply:

1. The transition from phase 2 to phase 3 may follow either an *A-Data (spont., T)* primitive or an *A-Conf-Data (spont.)* primitive. If it follows an *A-Data (spont., T)* primitive, the error condition described in rule 2 of phase 2 above will eventually occur.
2. After an *Unsolicited Data Transfer FU* invocation for a given group has been terminated, the RESPONDER UE must not try to invoke the *Unsolicited Mixed Data Transfer FU* for that group.

³⁰If data belong to more than one group, each occurrence of the data in the outgoing queue shall be handled independently, according to their individual **Priority Class** values.

³¹A RESPONDER UE altogether lacking support for the priority mechanism shall report the fact as an error whenever an INITIATOR UE tries to create or change a group into one for which **Priority Class** is different from zero. See the *Group Management FU* description.

7.4.3.1.2. Parameter values

A-Spont-Mgmt :

Parameter	req. (INITIATOR)	res. (RESPONDER)
Function	Phase 1: = <i>start</i> Phase 3: = <i>stop</i>	Copy of value from <i>ind</i> .
Gtype	Value <i>Measure-group, Status-group, Discrete-group, Logical-breaker status-group, or Text-message-group</i> , depending on the type of the group in question, as defined in the CS.	Copy of value from <i>ind</i> .
Gnr	Reference number for the group in question, as defined in the CS.	Copy of value from <i>ind</i> .
Result	(Not applicable)	Action performed as specified. = <i>result-ok</i> Action not performed, due to error condition: Other value. See chapter 7.4.3.2 "Error handling". (If Function = <i>start</i> , this means that the RESPONDER part of the <i>Unsolicited Data Transfer FU</i> has not been invoked, for the group in question).

A-Data (spont.,) req. (RESPONDER):

Parameter	<i>req.</i>
Gtype	Copy of value from <i>A-Spont-Mgmt (start) ind.</i> of Phase 1
Gnr	Copy of value from <i>A-Spont-Mgmt (start) ind.</i> of Phase 1
Transmod	= <i>spontaneous</i>
Index1 ³²	Shall be > 0: Equal to the lowest Object number , in accordance with the CS(R), of the range of Object numbers whose data values are contained in the Data parameter (below). Shall be equal to Index2, if Gtype = <i>Text-message-group</i> . ³³
Index2	Shall be >= value of Index1: Equal to the highest Object number , in accordance with the CS(R), of the range of Object numbers whose data values are contained in the Data parameter (below). Shall be equal to Index1, if Gtype = <i>Text-message-group</i> .
T	Time stamp, applying to the whole set of values contained in the Data parameter (below), and determined by the RESPONDER UE. The use of UTC is recommended, please refer Appendix F for ELCOM-83 compatibility.
More-D	If another <i>A-Data (spont.)</i> primitive for the group specified by the Gnr parameter (above) will follow shortly, so that data acknowledgement (<i>A-Conf-Data</i> primitive, issued by the INITIATOR) can be postponed: = <i>true</i> . If another <i>A-Data (spont.)</i> primitive for the group specified by the Gnr parameter (above) will NOT follow shortly, so that data acknowledgement (<i>A-Conf-Data</i> primitive) shall be issued by the INITIATOR: = <i>false</i> .
Data	The actual Elcom data transferred. The structure is defined in Appendix A of this document
Length	Length of Data in octets. Shall be computed according to the length of the actual datatype, Index1 and Index2.
Result	= <i>result-ok</i>

³²The number of values that shall be contained in the Data parameter can be computed as: Index2 - Index1 + 1.

³³Only one object per transmission, for such groups.

A-Conf-Data (spont.) req. (INITIATOR):

Parameter	<i>req.</i>
Gtype	Copy of value from <i>A-Spont-Mgmt (start) ind.</i> of Phase 1
Gnr	Copy of value from <i>A-Spont-Mgmt (start) ind.</i> of Phase 1
Transmod	= <i>spontaneous</i>
Result	If no error detected by the INITIATOR UE: = <i>result-ok</i> . If error detected by the INITIATOR UE: Other value. See "Error handling" (below).

7.4.3.2. Error handling

7.4.3.2.1. FU disruption

Disruption by the *Permanent Association FU*:

Disruption of both the INITIATOR part and the RESPONDER part of the current invocation of the *Unsolicited Data Transfer FU* shall be triggered locally, as a part of the handling of incoming *A-P-Abort ind.* primitives in both the INITIATOR part and the RESPONDER part of the *Permanent Association FU* invocation handling the association on which the *Unsolicited Data Transfer FU* is running.

Both parts of the current invocation of the *Unsolicited Data Transfer FU* shall be terminated gracefully, neither part attempting to issue any primitive associated with the termination itself.³⁴

Following the termination, the *Permanent Association FU* will enter a state in which it will signal *Restart, spontaneous management lost* or *Restart, group management lost*, the next time an association with the characteristics for Unsolicited Data Transfer is established between the same INITIATOR and RESPONDER UE pair.³⁵

7.4.3.2.2. Illegal invocation attempt

FU not present:

If the *Unsolicited Data Transfer FU* is not present in an INITIATOR UE:

Invocation requests are always generated locally; see chapter 7.4.2.3 "Invocation", above. Consequently, the handling of this type of error is a local issue, outside the scope of this document.

If the *Unsolicited Data Transfer FU* is not present in an RESPONDER UE:

The RESPONDER User Element shall respond to activation attempts in one of two ways:

Either:

- Ignoring the incoming *A-Spont-Mgmt (start) ind.* primitive altogether

or:

- Issuing an *A-Spont-Mgmt (start) res.* primitive with Result = *remote-service-user-unavailable*

³⁴Local clean-up procedures are not specified by this document.

³⁵The RESPONDER UE may be programmed to always signal the restart code "*Restart, spontaneous management lost*" during creation of associations for Unsolicited Data Transfer, regardless of previous history.

FU present, but attempt illegal:

Attempts of multiple simultaneous invocations of the *Unsolicited Data Transfer FU* for any given group in an INITIATOR UE is a local issue, outside the scope of this document.

Attempts of multiple simultaneous invocations of the *Unsolicited Data Transfer FU* for any given group in a RESPONDER UE shall be handled by the RESPONDER UE in one of two ways:

Either:

- Ignoring the incoming *A-Spont-Mgmt (start) ind.* primitive altogether

or:

- Issuing an *A-Spont-Mgmt (start) res.* primitive with Result = *remote-service-user-unavailable* without actually (re-)invoking the *Unsolicited Data Transfer FU* for the group concerned, in the RESPONDER UE.

7.4.3.2.3. Incoming EASE service primitive out of context

INITIATOR part:

State	<i>A-Spont-Mgmt (start) cnf.</i>	<i>A-Spont-Mgmt (stop) cnf.</i>	<i>A-Data (spont) ind.</i>
FU not running	Ignore, or local error indication/logging	Ignore, or local error indication/logging	Ignore, or: Simulate the entry of Phase 3 of an <i>Unsolicited Data Transfer FU</i> invocation for the group in question, thus terminating the (supposed) current invocation of the <i>Unsolicited Data Transfer FU</i> in the RESPONDER UE., or: Issue <i>A-Conf-Data (spont) req.</i> , with Result = <i>spontaneous-transfer-not-initiated</i> .
FU running, waiting for <i>A-Spont-Mgmt (start) cnf.</i>	(Normal)	(Parameter error)	Ignore, or local error indication/logging
Waiting for <i>A-Data (spont) ind.</i>	Ignore, or local error indication/logging	Ignore, or local error indication/logging	(Normal)
Waiting for <i>A-Spont-Mgmt (stop) cnf.</i>	(Parameter error)	(Normal)	Ignore, or local error indication/logging

RESPONDER part:

State	<i>A-Spont-Mgmt (start) ind.</i>	<i>A-Spont-Mgmt (stop) ind.</i>	<i>A-Conf-Data (spont) ind.</i>
FU not running	(Normal)	Ignore, or: Issue an <i>A-Spont-Mgmt (stop) res.</i> primitive with Result = <i>result-ok</i> , without terminating any <i>Unsolicited Data Transfer FU</i> invocation in the RESPONDER UE.	Ignore, or local error indication/logging
FU running, not waiting for <i>A-Conf-Data (spont.) ind.</i>	Illegal invocation attempt; see relevant chapter.	(Normal) If Gnr is not active for spont. transfer, issue an <i>A-Spont-Mgmt (stop) res.</i> primitive with Result = <i>spontaneous-transfer-not-initiated</i>	(Normal)
FU running, waiting for <i>A-Conf-Data (spont.) ind.</i>	Illegal invocation attempt; see relevant chapter.	(Normal)	(Normal)

7.4.3.2.4. Timing errors

Error in INITIATOR part:

Error	Reaction from EASE	Specified action in FU
UE too late issuing <i>A-Conf-Data (spont.) req.</i> after receiving <i>A-Data (spont., F) ind.</i> :	In INITIATOR: Local error from eventual attempt at issuing <i>A-Conf-Data (spont) req.</i> In RESPONDER: <i>A-Conf-Data (spont) ind.</i> , with Result = <i>remote-service-user-unavailable</i> .	INITIATOR part: Ignore, or local error indication/logging, then proceed as normal. RESPONDER part: Enter procedure for Missing data acknowledgement in RESPONDER. See chapter on parameter errors, below.

Error in RESPONDER part:

Error	Reaction from EASE	Specified action in FU
<p>UE too late responding to <i>A-Spont-Mgmt (start) ind.</i></p>	<p>In RESPONDER: Local error from eventual attempt at issuing <i>A-Spont-Mgmt (start) res.</i></p> <p>In INITIATOR: <i>A-Spont-Mgmt (start) cnf.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Terminate FU invocation locally.</p> <p>INITIATOR part: Terminate FU invocation locally.</p>
<p>UE too late issuing next <i>A-Data (spont.) req.</i> after latest <i>A-Data (spont., T) req.</i> issued</p>	<p>In RESPONDER: <i>A-Conf-Data (spont) ind.</i>, with Result = <i>misbehaviour-of-local-service-user</i>.</p> <p>In INITIATOR: <i>A-Data (spont.) ind.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Enter procedure for Missing data acknowledgement in RESPONDER. See chapter on parameter errors, below.</p> <p>INITIATOR part: Local error indication/logging, then proceed as normal.</p>
<p>UE too late responding to <i>A-Spont-Mgmt (stop) ind.</i></p>	<p>In RESPONDER: Local error from eventual attempt at issuing <i>A-Spont-Mgmt (stop) res.</i></p> <p>In INITIATOR: <i>A-Spont-Mgmt (stop) cnf.</i>, with Result = <i>remote-service-user-unavailable</i>.</p>	<p>RESPONDER part: Terminate FU invocation locally.</p> <p>INITIATOR part: Terminate FU invocation locally.</p>

7.4.3.2.5. Congestion error

INITIATOR part of the FU:

When occurring with an *A-Spont-Mgmt (start) req.* attempt:
Terminate FU invocation locally.

When occurring with an *A-Conf-Data (spont) req.* attempt:
Enter Phase 3 (Orderly Termination) of the FU, attempting to issue *A-Spont-Mgmt (stop) req.*, with Result = *misbehaviour-of-local-service-user*.

When occurring with an *A-Spont-Mgmt (stop) req.* attempt:
Ignore, or notify operator

RESPONDER part of the FU:

When occurring with an *A-Spont-Mgmt (start) res.* attempt:
Terminate FU invocation locally. (INITIATOR part of FU will eventually be terminated after time-out in the Elcom provider.)

When occurring with an *A-Data req.* attempt:
Trigger local abrupt termination of supporting association³⁶

When occurring with an *A-Spont-Mgmt (stop) res.* attempt:
Terminate FU invocation locally. (INITIATOR part of FU will eventually be terminated after time-out in the Elcom provider.)

³⁶Such abrupt termination is effected by a local "detach" call (ADET) against the EAPI, followed by a local "attach" call (AATT).

7.4.3.2.6. EASE service primitive parameter errors

For all primitives except the *A-Spont-Mgmt (start) ind.* primitive, the parameter Gnr will always be valid within the context of this FU, *provided the FU is running*³⁷ for that group: It serves as identification of the FU invocation to which the incoming primitive shall be directed.

Errors in the *A-Spont-Mgmt (start) ind.* primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Gtype value not equal to value of attribute Group type in CS(R) for group no. Gnr. (The case of non-existing group no. Gnr is considered below.)	Issue <i>A-Spont-Mgmt (start) res.</i> with Result = <i>gtype-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not invoke the FU.
Value of Gnr is illegal, or group no. Gnr does not exist in the CS(R).	Issue <i>A-Spont-Mgmt (start) res.</i> with Result = <i>gnr-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not invoke the FU.
Value of Gtype is illegal.	Issue <i>A-Spont-Mgmt (start) res.</i> with Result = <i>gtype-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not invoke the FU.
Group is created, but not defined	Issue <i>A-Spont-Mgmt (start) res.</i> with Result = <i>index-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not invoke the FU.

Errors in the *A-Conf-Data (spont) ind.* primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Gtype value different from the value in CS(R) for the group associated with the current FU invocation	Enter procedure for Missing data acknowledgement (immediately below).
Result >< <i>result-ok</i>	Enter procedure for Missing data acknowledgement (immediately below).

³⁷The case of the FU **not** being invoked for the group shall be handled outside the FU (by the Coordinating Function): The natural replying primitive shall be issued, with parameter Result = *gnr-out-of-range*.

Procedure for **Missing data acknowledgement**:

- If Gtype OK and Result = *spontaneous-transfer-not-initiated* (INITIATOR part not running):
 Terminate FU invocation locally
- If Gtype OK and Result = *misbehaviour-of-local-service-user* (RESPONDER data rate too slow, for More-D=T):
 Proceed as normal
- If Gtype not OK or Result any other value than *result-ok*, *spontaneous-transfer-not-initiated* and *misbehaviour-of-remote-service-user*:
 Retry, a locally determined number of times, including 0 (no retry) and infinite (always retry)
 - Wait, for a locally determined time span
 - Repeat all un-acknowledged *A-Data (spont) ind.*'s
 Terminate retry loop on any of the conditions (and actions) above, as well as Result = *result-ok*, in which case:
 Proceed as normal.
 If all retrials failed (Result ><*result-ok*, for all retrials), or local decision not to retry at all:
 Proceed as normal, or:
 Trigger abrupt termination of supporting association

Errors in the ***A-Spont-Mgmt (stop) ind.*** primitive (detected by the RESPONDER part):

Error	Action in RESPONDER part of FU
Gtype value different from the value in CS(R) for the group associated with the current FU invocation	Issue <i>A-Spont-Mgmt (stop) res.</i> with Result = <i>gtype-out-of-range</i> , and the values of Gnr and Gtype as in the corresponding <i>ind.</i> Do not terminate the FU.

Errors in the ***A-Spont-Mgmt (start) cnf.*** primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Mismatch between Gnr/Gtype/Function in primitive and Gnr/Gtype/Function in corresponding <i>A-Spont-Mgmt (start) req.</i> , and Result = <i>result-ok</i> .	Trigger Orderly Termination of the current FU invocation, issuing <i>A-Spont-Mgmt (stop) req.</i> for the group in question, with value of Gnr/Gtype/Function as in the corresponding <i>A-Spont-Mgmt (start) req.</i>
Result >< <i>result-ok</i>	Terminate the FU invocation locally

Errors in the ***A-Data (spont.) ind.*** primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
<p>Mismatch between Gtype in primitive and Gtype in corresponding <i>A-Spont-Mgmt (start) req.</i>.</p> <p>Invalid Index1 or Index2 (Validity conditions are given above, in chapter 7.4.3.1.2 "Parameter values".)</p> <p>Invalid Data (See Appendix A in this document for validity conditions)</p> <p>Result = result-ok, and mismatch between Length and Index1/index2</p> <p>Result = <i>result-ok</i>, and format error in T</p> <p>Result >< <i>result-ok</i></p> <p>The security class 2: The received authentication code >< the generated authentication code based in the received data.</p> <p>For security class 3: The received checksum >< the generated checksum during the decipherment.</p>	<p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count.</p> <p>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count.</p>

Errors in the *A-Spont-Mgmt (stop) cnf.* primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
<p>Mismatch between Gtype in primitive and Gtype in corresponding <i>A-Spont-Mgmt (stop) req.</i>, and Result = <i>result-ok</i>.</p> <p>Result >< <i>result-ok</i></p>	<p>Trigger abrupt termination of supporting association</p> <p>Trigger abrupt termination of supporting association</p>

7.5. Unsolicited Mixed Data Transfer FU

Type: Secondary.

7.5.1. Function

The purpose of the *Unsolicited Mixed Data Transfer FU* is to allow buffered spontaneous data reporting by a RESPONDER UE: Data belonging to different group types may be put into a common buffer prior to transmission. Transmission will then be effected by invoking the *Unsolicited Mixed Data Transfer FU* for data transfer (below) with the buffer contents as data. The FU invocation may be triggered by for example the buffer becoming full, a timer, or another local RESPONDER system event. The existence of a data buffering mechanism is not mandatory for this FU.

The *Unsolicited Mixed Data Transfer FU* is of type Secondary, meaning that invocation of the FU is triggered exclusively by the RESPONDER UE. The RESPONDER part of an *Unsolicited Mixed Data Transfer FU* may be invoked for two functions: **data transfer** and **error reporting**.

The *Unsolicited Mixed Data Transfer FU* is invoked for **data transfer** by request from the local Co-ordination Function in the RESPONDER UE.

When invoked for data transfer, the time of invocation, as well as the size and contents of the data part is determined by the RESPONDER UE, subject to the following rules:

- The *Unsolicited Data Transfer FU* and/or the *Periodic Data transfer FU* must currently be running for each group for which the *Unsolicited Mixed Data Transfer FU* is to be invoked.³⁸
- *Unsolicited Mixed Data Transfer FU* shall only be invoked for data from groups of the following types: *Measure-group*, *Status-group*, *Discrete-group* and *Logical-breaker-status-group*.
- The data may be associated with any number of Elcom groups, subject only to two limitations:
 1. A maximum length of the Data parameter of the *A-Mixed-Data req.* primitive must not be exceeded.³⁹
 2. The time span between values must not be too great; see Appendix A on the "Delta T" field of the Data parameter.
- Each single data value is identified by the group configuration attributes **Group number** (coded into the Gnr parameter) and **Object number** (coded into the Index parameter).

³⁸Such preceding FU invocations can be considered a "priming" mechanism for invocations of the *Unsolicited Mixed Data Transfer FU* in the RESPONDER UE, for the individual groups.

³⁹ The limiting factor is the maximum number of user octets in a A-PDU. An A-PDU normally carries 256 octets. The number of octets not available for data values is given by:

P-layer overhead:	1 octet
Non-Data value part of M-D-REQ PDU:	9 octets

Sum: 10 octets

Thus, the set of **Data** for the subgroup, coded as defined in [8], chapter 5.8.15, normally must be accommodated within 256-10 = **246** octets.

- The amount of data transferred may vary from invocation to invocation of the *Unsolicited Mixed Data Transfer FU*.

- If data buffering is used, as describe above, data values shall always be ordered according to value of attribute **Priority class** for the group to which each data value belongs:
 - If **Priority class** value = 0:
The RESPONDER UE may order the data values in any manner.
 - If **Priority class** value > 0:
Data values with lower **Priority class** value shall be given precedence over data values with higher **Priority class** value.

The INITIATOR part is always invoked for data transfer, but may decide to report errors detected with the received data by issuing a number of *A-Mixed-Data-Error req.* primitives on the same association as the one over which the data were received, before terminating the current invocation.

The *Unsolicited Mixed Data Transfer FU* is invoked for **error reporting** in the RESPONDER UE upon reception of an *A-Mixed-Data-Error ind.* service primitive.

Each *A-Mixed-Data-Error ind.* primitive contains an error report from the INITIATOR UE, concerning one⁴⁰ of the previous *Unsolicited Mixed Data Transfer FU* invocations for data transfer in the RESPONDER UE. One FU invocation for data transfer may trigger a number of subsequent invocations for error reporting; one invocation per *A-Mixed-Data-Error ind.* service primitive issued by the INITIATOR UE during the single invocation of the INITIATOR part of the FU. Processing of such errors on the part of the RESPONDER UE is not specified by this document.⁴¹

In order to transfer one batch of data from the RESPONDER UE to the INITIATOR UE using the *Unsolicited Mixed Data Transfer FU*, a single invocation is normally sufficient, both of the RESPONDER part and the INITIATOR part.

If the INITIATOR UE finds it necessary to report N errors back to the RESPONDER UE concerning the received data, the data transfer and error reporting sequence will encompass N+1 invocations in the RESPONDER UE and one in the INITIATOR UE:

- Invocation of the RESPONDER part for data transfer.
- Invocation of the INITIATOR part, including issuing of N error primitives.
- Invocation no. 1 of the RESPONDER part for error reporting
- Invocation no. 2 of the RESPONDER part for error reporting
-
- Invocation no. N of the RESPONDER part for error reporting

⁴⁰Which invocation the error report actually concerns, cannot be identified. The primitive only indicates the group number concerned.

⁴¹Typically, the RESPONDER UE may report/log the error, or try to re-invoke the *Unsolicited Mixed Data Transfer FU* for data transfer.

7.5.2. Coordination rules

7.5.2.1. Association usage

All Elcom interactions that are part of one invocation of the *Unsolicited Mixed Data Transfer FU* are conveyed by one single association. The association shall be permanent, and have the characteristics as specified in chapter 7.5.2.3.1 "Prerequisites", below.

7.5.2.2. Relation to other FUs

7.5.2.2.1. Invoking FUs

The *Unsolicited Mixed Data Transfer FU* may be invoked by:

- *Periodic Data Transfer FU*
- *Unsolicited Data Transfer FU*.

7.5.2.2.2. Invoked FUs

The *Unsolicited Mixed Data Transfer FU* shall not invoke any other FU.

7.5.2.2.3. Disrupting FUs

The *Unsolicited Mixed Data Transfer FU* may be disrupted by:

- *Permanent Association FU*
- *Group Management FU*
- *Group Definition FU*
- *Restart Reconfigure FU*

7.5.2.2.4. Disrupted FUs

The *Unsolicited Mixed Data Transfer FU* shall not disrupt any other FU.

7.5.2.3. Invocation

7.5.2.3.1. Prerequisites

The following FUs shall have been invoked preceding any invocation of the *Unsolicited Mixed Data Transfer FU*:

- The *Permanent Association FU*
- A number of *Unsolicited Data Transfer FU* and/or *Periodic Data Transfer FU* invocations

The *Permanent Association FU* invocation shall still be running at the time of invocation of the *Unsolicited Mixed Data Transfer FU*, and the association maintained by the *Permanent Association FU* invocation shall be running (not temporarily broken) at that time.

The *Unsolicited Data Transfer FU* and/or *Periodic Data Transfer FU* invocations shall still be running at the time of invocation of the *Unsolicited Mixed Data Transfer FU*.

The *Permanent 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 *Unsolicited Mixed Data Transfer FU*, with the following characteristics:

- A-suffix pair as specified in the table in chapter 5.1.2 "A-Suffixes" in section 5 "THE ASSOCIATION MANAGEMENT (A) FUNCTION GROUP".

The *Unsolicited Data Transfer FU* and/or *Periodic Data Transfer FU* shall have been invoked **for each group** for which data is to be transferred by the *Unsolicited Mixed Data Transfer FU*.

7.5.2.3.2. Restrictions

For any given INITIATOR/RESPONDER system combination, multiple simultaneous invocations of the *Unsolicited Mixed Data Transfer FU* are not allowed.

The *Unsolicited Mixed Data Transfer FU* must not be invoked while at least one of the following FUs are running, for at least one of the groups involved:

- *Group Configuration FU*
- *Group Management FU*
- *Group Definition FU*

Neither shall the *Unsolicited Mixed Data Transfer FU* be invoked on a given association, if the *Requested Data Transfer FU* is currently running on that association.

7.5.2.3.3. Invoking events

The INITIATOR part of the *Unsolicited Mixed Data Transfer FU* is always invoked by reception of an *A-Mixed-Data ind.* service primitive via an association with the characteristics as defined for the *Unsolicited Mixed Data Transfer FU*.

Invocation of the RESPONDER part of the *Unsolicited Mixed Data Transfer FU* for **data transfer** is triggered by request from the local Coordination Function, the details of which are outside the scope of this document.

Invocation of the RESPONDER part of the *Unsolicited Mixed Data Transfer FU* for **error reporting** is triggered by reception of an *A-Mixed-Data-Error ind.* service primitive via an association with the characteristics as defined for the *Unsolicited Mixed Data Transfer FU*.

7.5.2.4. Termination

7.5.2.4.1. Orderly termination

The INITIATOR part of an *Unsolicited Mixed Data Transfer FU* invocation is always terminated after reception of the *A-Mixed-Data ind.* primitive, or, in case of error, after issuing an *A-Mixed-Data-Error req.* primitive.

The RESPONDER part of an *Unsolicited Mixed Data Transfer FU* invocation for data transfer is always terminated in an orderly manner after issuing an *A-Mixed-Data req.* service primitive containing the data to be transferred. If invoked for error reporting, the invocation is orderly terminated after reception of the *A-Mixed-Data-Error ind.* primitive.

7.5.2.4.2. Disruption

Both the INITIATOR and the RESPONDER part of an *Unsolicited Mixed Data Transfer FU* invocation may be disrupted by:

- Disruption of another FU invocation. See chapter 7.5.2.2.3 "Disrupting FUs", above.
- A fatal error condition encountered during operation. See chapter 7.5.3.2 "Error handling", below.

7.5.3. Procedures

7.5.3.1. EASE service primitives

The following elementary EASE service is used by the *Unsolicited Mixed Data Transfer FU*:

- *A-Mixed-Data*
- *A-Mixed-Data-Error* (Only in case of error)

7.5.3.1.1. Sequence

The normal sequence of primitives is dependent on the function for which the FU is invoked:

Invoked for *data transfer* (in RESPONDER by local Coordinating Function).

INITIATOR UE	EASE	RESPONDER UE
<i>A-Mixed-Data ind.</i>	←-----	<i>A-Mixed-Data req.</i>

The transaction carries values for a number of data objects.

The rules stated in chapter 7.5.1 "Function", above, shall apply.

Invoked for *error reporting* (in RESPONDER by incoming Error service primitive).

INITIATOR UE	EASE	RESPONDER UE
<i>A-Mixed-Data-Error req.</i>	----->	<i>A-Mixed-Data-Error ind.</i>
	:	
<i>A-Mixed-Data-Error req.</i>	----->	<i>A-Mixed-Data-Error ind.</i>

Each transaction concerns a single group.

The rules stated in chapter 7.5.1 "Function", above, shall apply.

7.5.3.1.2. Parameter values

A-Mixed-Data :

Parameter	<i>req. (RESPONDER)</i>
T	The point in time to which all relative time values coded into the Data parameter relate. The use of UTC is recommended, please refer Appendix F for ELCOM-83 compatibility.
Data	For each data value: - Group number: Value of attribute Group number . - Index: Value of attribute Object number - Delta T: Time, relative to parameter T - Quality code and value. Coding is described in Appendix A of this document.
Length	Length of Data in octets

A-Mixed-Data-Error :

Parameter	<i>req. (INITIATOR)</i>
Gnr	Attribute Group number for the group for which an error has been detected in a received <i>A-Mixed-Data ind.</i> primitive
Result	>< <i>result-ok</i> , indicating error type; see "Error handling", below.

7.5.3.2. Error handling

7.5.3.2.1. FU disruption

When disrupted, both the INITIATOR and the RESPONDER part of the current invocation of the *Unsolicited Mixed Data Transfer FU* shall be terminated gracefully, neither part attempting to issue any primitive associated with the termination itself.⁴²

7.5.3.2.2. Illegal invocation attempt

FU not present, or FU present but attempt illegal:

In a RESPONDER UE:

Invocation requests for data transfer are always generated locally; see chapter 7.5.2.3 "Invocation", above. Consequently, the handling of this type of error is a local issue, outside the scope of this document.

Erroneous invocation requests for error reporting shall be handled by the RESPONDER UE ignoring the incoming *A-Mixed-Data-Error ind.* primitive, optionally reporting and/or logging the event.

The INITIATOR User Element shall respond to erroneous activation attempts by ignoring the incoming *A-Mixed-Data ind.* primitive altogether

7.5.3.2.3. Incoming EASE service primitive out of context

The *A-Mixed-Data ind.* service primitive is never out of context in the INITIATOR UE. Likewise, in the RESPONDER UE the *A-Mixed-Data-Error ind.* primitive is never out of context.

7.5.3.2.4. Timing errors

There is no requirement for a specific timing of service primitives in this FU.

⁴²Local clean-up procedures are not specified by this document.

7.5.3.2.5. Congestion error

INITIATOR part of the FU:

Ignore, and optionally report/log the error.

RESPONDER part of the FU:

Trigger local abrupt termination of supporting association⁴³

⁴³Such abrupt termination is effected by a local "detach" call (ADET) against the EAPI, followed by a local "attach" call (AATT).

7.5.3.2.6. EASE service primitive parameter errors

Errors in the **A-Mixed-Data ind.** primitive (detected by the INITIATOR part):

The INITIATOR part shall in all these cases issue *A-Mixed-Data-Error req.* with Result = <Value from table below> and the value of Gnr equal to Gnr for the data value which is in error, as decoded from the Data parameter of the *A-Mixed-data ind.* primitive.

Errors that are not described in the table below shall be handled by the INITIATOR UE ignoring the incoming *A-Mixed-data ind.* primitive, optionally reporting/logging the event.

Error	Value of parameter Result in <i>A-Mixed-Data-Error req.</i>
<p>INITIATOR UE has not currently invoked the <i>Unsolicited Data Transfer FU</i> or the <i>Periodic Data Transfer FU</i> for group no. Gnr in the CS(I)</p> <p>Value of Gnr is illegal, or there exists no group with attribute value Group number equal to Gnr in the CS(I)</p> <p>Result = <i>result-ok</i>, and format error in T</p> <p>At least one of the values of Index field in Data parameter was outside legal set of values for attribute Object number within group no. Gnr in the CS(I), and the Gnr/Index pair was associated with the same data value in the primitive.</p> <p>Invalid Data (see appendix A for validity conditions).</p> <p>For security class 2: The received authentication code >< the generated authentication code based in the received data.</p> <p>The security class 3: The received checksum >< the generated checksum during the decipherment.</p>	<p><i>spontaneous-transfer-not-initiated</i></p> <p><i>gnr-out-of-range</i></p> <p><i>T-out-of-range</i></p> <p><i>index-out-of-range</i></p> <p><i>Ignore the data, but otherwise proceed as normal, optionally incrementing local error count.</i></p> <p><i>invalid-authentication-code-received</i></p> <p><i>decipherment-error</i></p>

Errors in the **A-Mixed-Data-Error ind.** primitive (detected by the RESPONDER part):

Handling of parameter errors shall be subject to local rules in the RESPONDER UE, not specified by this document. The RESPONDER part of the *Unsolicited Mixed Data Transfer FU* invocation shall be terminated locally, prior to such local error handling.

7.6. Supervisory Control Data Transfer FU

Type: Primary.

7.6.1. Function

A *Supervisory Control Data Transfer FU* invocation allows the invoking INITIATOR UE to exert a certain amount of functional control over the associated RESPONDER UE, by way of passing setpoint values or previously agreed-upon binary commands to it.

Binary commands and setpoint values shall be defined as Elcom groups of the relevant types.

Actual semantics (functional interpretation) of binary commands are not specified by this document, and is therefore subject to separate agreement between the INITIATOR/RESPONDER UE pairs involved.

The FU may be invoked for either a **one-phase** or a **two-phase** action. A one-phase action comprises a single Elcom transaction, while a two-phase action comprises a pair of consecutive Elcom transactions.

In the case of a **one-phase** action, the RESPONDER UE shall attempt to execute the command passed to it, at once.

In the case of a **two-phase** action, the first Elcom transaction instructs the RESPONDER UE to prepare for execution of a command (optionally in the primitive) passed to it. The second Elcom transaction then instructs the RESPONDER UE to either execute the command that has been prepared for execution, or to abort the two-phase action, i.e. undo the effect of the first transaction of the current two-phase action. The second transition shall only be entered (by the INITIATOR UE) if the first transaction returned Result = *result-ok*.

7.6.2. Coordination rules

7.6.2.1. Association usage

All Elcom interactions that are part of one invocation of the *Supervisory Control Data Transfer FU* are conveyed by one single association. The association shall have the characteristics as specified in the chapter 7.6.2.3.1 "Prerequisites", below.

7.6.2.2. Relation to other FUs

7.6.2.2.1. Invoking FUs

The *Supervisory Control Data Transfer FU* shall not be invoked by any other FU.

7.6.2.2.2. Invoked FUs

The *Supervisory Control Data Transfer FU* shall not invoke any other FU.

7.6.2.2.3. Disrupting FUs

The *Supervisory Control Data Transfer FU* may be disrupted by:

- *Permanent Association FU*
- *Dynamic Association FU*
- *Group Management FU*
- *Group Definition FU*
- *Restart Reconfigure FU*

7.6.2.2.4. Disrupted FUs

The *Supervisory Control Data Transfer FU* shall not disrupt any other FU.

7.6.2.3. Invocation

7.6.2.3.1. Prerequisites

The following FUs shall have been invoked preceding any invocation of the *Supervisory Control Data Transfer FU*:

- The *Permanent Association FU* or *Dynamic Association FU*⁴⁴
- The *Group Configuration FU*⁴⁵

The *Permanent Association FU* or *Dynamic Association FU* invocation shall still be active at the time of invocation of the *Supervisory Control Data Transfer FU*. In the case of *Permanent Association FU* the association maintained by the invocation shall be active (not temporarily broken) at that time. The *Group Configuration FU* invocation shall be terminated at the time of invocation of the *Supervisory Control Data Transfer FU*.

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 *Supervisory Control Data Transfer FU*, with the A-suffix pair specified by the table in chapter 5.1.2 "A-Suffixes" in section 5 "THE ASSOCIATION MANAGEMENT FUNCTION GROUP".

The *Group Configuration FU* shall have been invoked in order to define and configure the group defining the command or setpoint in question, prior to the current invocation of the *Supervisory Control Data Transfer FU*.⁴⁶

7.6.2.3.2. Restrictions

For any given INITIATOR/RESPONDER system combination, multiple simultaneous invocations of the *Supervisory Control Data Transfer FU* are not allowed.

The *Supervisory Control Data Transfer FU* must not be invoked while at least one of the following FUs are active, for the group involved:

- *Group Configuration FU*
- *Group Management FU*
- *Group Definition FU*

⁴⁴Subject to local decision in the INITIATOR UE.

⁴⁵Alternatively, the *Group Management FU* and *Group Definition FU*, which are invoked by the *Group Configuration FU*, may have been invoked directly by the Coordinating Function.

⁴⁶The use of the *Group Configuration FU* is, however, not required if the group is defined in the RESPONDER AP by a non-Elcom mechanism and also made known to the INITIATOR AP by a non-Elcom mechanism.

7.6.2.3.3. Invoking events

The INITIATOR part of the *Supervisory Control Data Transfer FU* may be invoked by:

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

Invocation of the RESPONDER part of the *Supervisory Control Data Transfer FU* is attempted whenever a valid *A-Command-Transfer ind.* primitive with parameter *Com.type = CBXC* or *= IXC* is received via an association with the characteristics as defined for the *Supervisory Control Data Transfer FU*: See description of the *Permanent Association FU*.

If parameter *Com.type = CBXC*, the RESPONDER part shall be invoked for a two-phase action. The RESPONDER part shall be invoked for a one-phase action if parameter *Com.type = IXC*.

7.6.2.4. Termination

7.6.2.4.1. Orderly termination

Orderly termination of the INITIATOR part of a *Supervisory Control Data Transfer FU* invocation may be triggered by:

- Congestion error. See relevant chapter, below.
- Reception of an invalid *A-Command-Transfer cnf.* service primitive after issuing an *A-Command-Transfer req.* service primitive (termination on error).
- Reception of a valid *A-Command-Transfer cnf.* service primitive after issuing the last *A-Command-Transfer req.* service primitive of the current *Supervisory Control Data Transfer FU* invocation (normal termination).

The RESPONDER part of a *Supervisory Control Data Transfer FU* invocation for one-phase action always terminates itself in an orderly manner (normal termination) after issuing an *A-Command-Transfer res.* service primitive.

In the case of a two-phase action, the RESPONDER part of a *Supervisory Control Data Transfer FU* invocation terminates itself in an orderly manner either after issuing the second *A-Command-Transfer res.* service primitive (normal termination), or after issuing the first *A-Command-Transfer res.* service primitive with parameter *Result >< result-ok* (termination on error).

7.6.2.4.2. Disruption

Both the INITIATOR and the RESPONDER part of a *Supervisory Control Data Transfer FU* invocation may be disrupted by:

- Another FU invocation. See chapter 7.6.2.2.3 "Disrupting FUs", above.
- A fatal error condition encountered during operation. See chapter 7.6.3.2 "Error handling", below.

7.6.3. Procedures

7.6.3.1. EASE service primitives

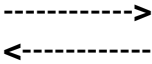
The following elementary EASE service is used by the *Supervisory Control Data Transfer FU*:

- *A-Command-Transfer*

7.6.3.1.1. Sequence

A. If invoked for **one-phase** action:

The **one-phase** action sequence of primitives is a single *A-Command-Transfer* transaction:

INITIATOR UE	EASE	RESPONDER UE
<i>A-Command-Transfer req.</i> <i>A-Command-Transfer cnf.</i>		<i>A-Command-Transfer ind.</i> <i>A-Command-Transfer res.</i>

Rules:

1. Upon reception of a valid *A-Command-Transfer ind.* service primitive with parameter Time mode = *T-argument-not-used*, the RESPONDER UE shall immediately try to execute the command embedded in the primitive. Only in case of successful⁴⁷ command execution shall the Result parameter in the *A-Command-Transfer res.* service primitive have the value *result-ok*.
2. Upon reception of a valid *A-Command-Transfer ind.* service primitive with parameter Time mode = *latest-point-of-time-when-command-can-be-issued-at-the-RTU-side*, the RESPONDER UE shall register the command and schedule it for execution as soon as possible, but not later than the point in time given by the parameter T of the primitive⁴⁸. The Result parameter in the *A-Command-Transfer res.* service primitive shall have the value *result-ok* only if the command has been successfully registered.

⁴⁷Evaluation of the successful command transmission to the process is subject to local decision in the RESPONDER UE.

⁴⁸Note that if the command eventually fails, the RESPONDER UE has no immediate way of reporting this back to the INITIATOR UE.

3. Upon reception of a valid *A-Command-Transfer ind.* service primitive with parameter Time mode = *point-of-time-when-command-shall-be-issued-at-the-RTU-side*, the RESPONDER UE shall register the command and schedule it for execution as soon as possible after (but not before) the point in time given by the parameter T of the primitive⁴⁹. The Result parameter in the *A-Command-Transfer res.* service primitive shall have the value *result-ok* only if the command has been successfully registered.
4. For groups of type *Binary-command-group*, the command carried by *A-Command-Transfer* primitives shall contain exactly one object, i.e. the value of parameters Index1 and Index2 shall always be identical in this case.
5. Setpoint values must be treated as one unit. This means that all setpoints must be accepted before any of them can be executed.

B. If invoked for **two-phase** action:

The primitive sequence is composed a pair of consecutive *A-Command-Transfer* transactions (see **A**, above), implementing a "check before execute" mechanism:

During the **first** transaction, the INITIATOR UE effectively requests the RESPONDER UE to register the command carried by the *A-Command-Transfer ind.* service primitive and prepare it for execution at some indefinite future time, to be decided upon later by the INITIATOR UE. The RESPONDER UE may impose a time limit between the two transactions, by locally terminating the FU invocation if no EXC/IHC is received within a time-out.⁵⁰ The RESPONDER UE indicates acceptance of the request by returning parameter Result = *result-ok*, and rejection by returning any other Result value.

The **second** transaction shall only be entered (by the INITIATOR UE) if the first transaction returned Result = *result-ok*. If Result = *result-ok*, and EASE service primitive parameter errors are discovered, this is regarded as Result >< *result-ok*. During the second transaction, the INITIATOR UE may instruct the RESPONDER UE to either:

- Actually execute the command registered during the first transaction, with parameter Com.type = *EXC*,

or:

- Undo the action of the first transaction, effectively aborting the two-phase action, with parameter Com.type = *IHC*.

The RESPONDER UE shall always return Result = *result_ok* when Com.type = *IHC*.

Rules **1** through **5** for the one-phase action also apply for the second transaction of the two-phase action. The two-phase action must be completed before a new two-phase action can be invoked.

⁴⁹Note that if the command eventually fails, the RESPONDER UE has no immediate way of reporting this back to the INITIATOR UE.

⁵⁰ The responder UE can supervise that the second transaction of a two phase command is received within acceptable time limits by invoking a local timer when a successful first transaction of a two phase command is received. The second transaction is only accepted within the timer limit.

After the timer expires the second transaction is rejected with result code = *CBXC-not-received-before-EXC*. Local cleanup after expired timer, and timer length is a decision for the responder UE.

7.6.3.1.2. Parameter values

A-Command-Transfer:

Parameter	req. (INITIATOR)	res. (RESPONDER)
Gtype	<p>One-phase action, or first transaction of two-phase action: Value <i>Binary-command-group</i>, <i>Analog-setpoint-group</i> or <i>Digital-setpoint-group</i>. Value shall be equal to assumed value of attribute Group type in the CS(R) for group no. Gnr.</p> <p>Second transaction of two-phase action: Copy of value from first transaction.</p>	Copy of value from <i>ind</i> .
Gnr	<p>One-phase action, or first transaction of two-phase action: Reference number for the group in question. Value of attribute Group number in set of attributes in the CS(R), which together with its attribute Object set defines the command.</p> <p>Second transaction of two-phase action: Copy of value from first transaction.</p>	Copy of value from <i>ind</i> .
Index1	<p>One-phase action, or first transaction of two-phase action: Lower bound of contiguous range of values of component attribute Object number taking part in defining the command. Shall be ≥ 1 and \leq assumed value of attribute Group size for the group, in the CS(R). If Gtype = <i>Binary-command-group</i>: Shall be = Index2.</p> <p>Second transaction of two-phase action: Copy of value from first transaction.</p>	Copy of value from <i>ind</i> .

cont.

Parameter	req. (INITIATOR)	res. (RESPONDER)
Index2	<p>One-phase action, or first transaction of two-phase action: Upper bound of contiguous range of values of component attribute Object number taking part in defining the command. Shall be \geq Index1 and \leq assumed value of attribute Group size for the group, in the CS(R). If Gtype = <i>Binary-command-group</i>: Shall be = Index1.</p> <p>Second transaction of two-phase action: Copy of value from first transaction.</p>	Copy of value from <i>ind</i> .
T	<p>One-phase action, or first transaction of two-phase action: If Time mode = <i>T-argument-not-used</i>: Any value. If Time mode = <i>latest-point-of-time-when-command-can-be-issued-at-the-RTU-side</i>: The time before which the command shall be executed. If Time mode = <i>point-of-time-when-command-shall-be-issued-at-the-RTU-side</i>: The time before which the command shall not be executed.</p> <p>Second transaction of two-phase action: Copy of value from first transaction.</p>	<p>If Time mode = <i>T-argument-not-used</i>: Copy of value from <i>ind</i>. If Time mode = <i>time-of-issuance</i> (= +1): The time at which the command has been executed, or at which the command is scheduled to execute.</p>

cont.

Parameter	req. (INITIATOR)	res. (RESPONDER)
Time mode	<p>One-phase action, or first transaction of two-phase action: Shall be = <i>T-argument-not-used</i> or <i>latest-point-of-time-when-command-can-be-executed-at-the-RTU-side</i> or <i>point-of-time-when-command-shall-be-issued-at-the-RTU-side</i>.</p> <p>Second transaction of two-phase action: Copy of value from first transaction.</p>	<p>One-phase action: If Time mode in <i>ind.</i> = <i>T-argument-not-used</i>: Copy of value from <i>ind.</i> If Time mode in <i>ind.</i> = <i>latest-point-of-time-when-command-can-be-issued-at-the-RTU-side</i> or = <i>point-of-time-when-command-shall-be-issued-at-the-RTU-side</i>: Shall be = <i>time-of-issuance</i> (= +1).</p> <p>First transaction of two-phase action: Copy of value from <i>ind.</i></p> <p>Second transaction of two-phase action: If Time mode in <i>ind.</i> = <i>T-argument-not-used</i>: Copy of value from <i>ind.</i> If Time mode in <i>ind.</i> = <i>latest-point-of-time-when-command-can-be-issued-at-the-RTU-side</i> or = <i>point-of-time-when-command-shall-be-issued-at-the-RTU-side</i>: Shall be = <i>time-of-issuance</i> (= +1).</p>
Com.type	<p>One-phase action: Shall be = <i>IXC</i> (immediate execute): Defines action as one-phase.</p> <p>First transaction of two-phase action: Shall be = <i>CBXC</i> (check before execute command): Defines action as two-phase.</p> <p>Second transaction of two-phase action: Shall be = <i>EXC</i> (execute command), or = <i>IHC</i> (inhibit command: Undo first phase).</p>	<p>If Com.type in <i>ind.</i> = <i>IXC</i>: Shall be = <i>EXR</i>.</p> <p>If Com.type in <i>ind.</i> = <i>CBXC</i>: Shall be = <i>CBR</i>.</p> <p>If Com.type in <i>ind.</i> = <i>EXC</i>: Shall be = <i>EXR</i>.</p> <p>If Com.type in <i>ind.</i> = <i>IHC</i>: Shall be = <i>IHR</i>.</p>
Data	<p>Actual command object values, coded according to Appendix A of this document. May be empty, if Com.type = <i>CBXC</i>.</p>	<p>Copy of data value from <i>ind.</i> The quality code shall be set by the RESPONDER.</p>
Length	<p>Length of Data in octets. If the data-field is empty: Length = 0</p>	<p>Copy of value from <i>ind.</i></p>

Parameter	<i>req.</i> (INITIATOR)	<i>res.</i> (RESPONDER)
Result	(Not applicable)	Functional OK/error code: Action performed as specified: <i>result-ok.</i> Action not performed, due to error condition: Error code; see chapter 7.6.3.2 "Error handling", below.

7.6.3.2. Error handling

7.6.3.2.1. FU disruption

Disruption by the *Permanent Association FU* or *Dynamic Association FU*:

Disruption of both the INITIATOR part and the RESPONDER part of the current invocation of the *Supervisory Control Data Transfer FU* shall be triggered locally, as a part of the handling of incoming *A-P-Abort ind.* primitives in both the INITIATOR part and the RESPONDER part of the *Permanent Association FU* or *Dynamic Association FU* invocation handling the association over which the *Supervisory Control Data Transfer FU* is active.

Both parts of the current invocation of the *Supervisory Control Data Transfer FU* shall be terminated gracefully, neither part attempting to issue any primitive associated with the termination itself.⁵¹

7.6.3.2.2. Illegal invocation attempt

FU not present:

If the *Supervisory Control Data Transfer FU* is not present in an INITIATOR UE:

The handling of this type of error resulting from an invocation attempt by the local Coordinating Function is a local issue, outside the scope of this document.

If the *Supervisory Control Data Transfer FU* is not present in a RESPONDER UE:

The RESPONDER UE shall respond to activation attempts in one of two ways:

Either:

- Ignoring the incoming *A-Command-Transfer ind.* primitive altogether

or:

- Issuing an *A-Command-Transfer res.* primitive with Result = *remote-service-user-unavailable*.

FU present, but attempt illegal:

Invocation attempts violating the rules stated in chapter 7.6.2.3.2 "Restrictions" above in an INITIATOR UE is a local issue, outside the scope of this document.

In a RESPONDER UE, invocation attempts violating the same rules shall be handled by the RESPONDER UE in one of two ways:

Either:

- Ignoring the incoming *A-Command-Transfer ind.* primitive altogether

or:

- Issuing an *A-Command-Transfer res.* primitive with Result = *remote-service-user-unavailable*, without actually (re-)invoking the *Supervisory Control Data Transfer FU* for the group concerned, in the RESPONDER UE.

⁵¹Local clean-up procedures are not specified by this document.

7.6.3.2.3. Incoming EASE service primitive out of context

INITIATOR part:

State	<i>A-Command-Transfer cnf.</i>
FU not invoked	Ignore, or local error indication

RESPONDER part: Not applicable: An *A-Command-Transfer ind.* service primitive is never out of context in the RESPONDER.

7.6.3.2.4. Timing errors

Error in INITIATOR part: Not applicable: There are no timing restraints placed upon the INITIATOR part of the *Supervisory Control Data Transfer FU*.

Error in RESPONDER part:

Error	Reaction from EASE	Specified action in FU
UE too late responding to <i>A-Command-Transfer ind.</i>	In RESPONDER: Local error from eventual attempt at issuing <i>A-Command-Transfer res.</i> In INITIATOR: <i>A-Command-Transfer cnf.</i> , with Result = <i>remote-service-user-unavailable</i> .	RESPONDER part: Terminate FU invocation locally. INITIATOR part: Terminate FU invocation locally.

The RESPONDER UE may impose a time limit between the two transactions, by locally terminating the FU invocation if no EXC/IHC is received within a time-out.

7.6.3.2.5. Congestion error

INITIATOR part of the FU:

When occurring with an *A-Command-Transfer req.* attempt:
Terminate FU invocation locally.

RESPONDER part of the FU:

When occurring with an *A-Command-Transfer res.* attempt:
Terminate FU invocation locally. (INITIATOR part of FU will eventually be terminated after time-out in the Elcom provider.)

7.6.3.2.6. EASE service primitive parameter errors

Errors in the **A-Command-Transfer ind.** primitive (detected by the RESPONDER part):

A. One-phase action, or first transaction of two-phase action:

The RESPONDER part shall in all these cases issue an *A-Command-Transfer res.* primitive, with Result = <Value from table below>, and the values of Gtype, Gnr, Index1, Index2, T, Time mode, Com.type and Data as in the corresponding *ind.*
The RESPONDER part of the FU must not be invoked.

Error	Value of parameter Result
Gtype value not equal to <i>Binary-command-group</i> , <i>Analog-setpoint-group</i> or <i>Digital-setpoint-group</i> , or not equal to value of attribute Group type in CS(R) for group no. Gnr. (The case of non-existing group no. Gnr is considered below.)	<i>gtype-out-of-range</i>
Value of Gnr is illegal, or group no. Gnr does not exist in the CS(R)	<i>gnr-out-of-range</i>
Index1 <= 0 or Index2 <= 0	<i>index-out-of-range</i>
If Gtype = <i>Binary-command-group</i> : Index1 >< Index2	<i>index-out-of-range</i>
If Gtype = <i>Analog-setpoint-group</i> or = <i>Digital-setpoint-group</i> : Index1 outside legal range of values of component attribute Object number for defined objects in group no. Gnr in CS(R). Legal range: 1 - Last defined object in group	<i>index-out-of-range</i>
If Gtype = <i>Analog-setpoint-group</i> or = <i>Digital-setpoint-group</i> : Index2 < Index1 or Index2 > upper bound of values of component attribute Object number for defined objects in group no. Gnr in CS(R). Upper bound: Last defined object in group.	<i>index-out-of-range</i>
If Com.type >< <i>CBXC</i> , or (Com.type = <i>CBXC</i> and non-empty Data): Number of values in Data not equal to Index2 - Index1 + 1.	<i>index-out-of-range</i>
Time mode not equal to <i>T-argument-not-used</i> , <i>latest-point-of-time-when-command-can-be-issued-at-the-RTU-side</i> or <i>point-of-time-when-command-shall-be-issued-at-the-RTU-side</i> .	<i>time-mode-out-of-range</i>
cont.	

Error	Value of parameter Result
<p>Time mode equal to <i>latest-point-of-time-when-command-can-be-issued-at-the-RTU-side</i> or <i>point-of-time-when-command-shall-be-issued-at-the-RTU-side</i>, the RESPONDER UE not supporting this time mode.</p>	<p><i>time-mode-not-supported-by-A-service-user</i></p>
<p>Time mode = <i>latest-point-of-time-when-command-can-be-issued-at-the-RTU-side</i> or <i>point-of-time-when-command-shall-be-issued-at-the-RTU-side</i>, and T value illegal or not a future time.</p>	<p><i>T-out-of-range</i></p>
<p>Com.type different from values <i>IXC</i> and <i>CBXC</i>.</p>	<p><i>command-type-out-of-range</i></p>
<p>Com.type = <i>IHC</i> or <i>EXC</i></p>	<p><i>CBXC-not-received-before-EXC</i></p>
<p>Invalid Data (See Appendix A in this document for validity conditions)</p>	<p><i>command-type-not-supported-by-A-service-user</i></p>
<p>Specified command cannot be executed, because of local implementation restrictions</p>	<p><i>command-type-not-supported-by-A-service-user</i></p>
<p>RESPONDER UE not able to perform the requested function, due to any reason other than those listed above.</p>	<p><i>remote-service-user-unavailable</i></p>
<p>For security class 2: The received authentication code >< the generated authentication code based in the received data.</p>	<p><i>invalid-authentication-code-received</i></p>
<p>For security class 3: The received checksum >< the generated checksum during the decipherment.</p>	<p><i>decipherment-error</i></p>

B. Second transaction of two-phase action:

The RESPONDER part shall in all these cases issue an *A-Command-Transfer res.* primitive, with Result = <Value from table below>, and the values of Gtype, Gnr, Index1, Index2, T, Time mode, Com.type and Data as in the corresponding *ind.*

The RESPONDER part of the FU shall be terminated locally, without attempting to execute the command registered during the first (previous) transaction.

Error	Value of parameter Result
Result <> <i>result-ok</i> returned in first transaction	<i>remote-service-user-unavailable</i>
Gtype not equal to corresponding parameter in first transaction	<i>gtype-out-of-range</i>
Gnr not equal to corresponding parameter in first transaction	<i>gnr-out-of-range</i>
Index1 not equal to corresponding parameter in first transaction	<i>index-out-of-range</i>
Index2 not equal to corresponding parameter in first transaction	<i>index-out-of-range</i>
T not equal to corresponding parameter in first transaction	<i>T-out-of-range</i>
Time mode not equal to corresponding parameter in first transaction	<i>time-mode-out-of-range</i>
Time mode = <i>latest-point-of-time-when-command-can-be-issued-at-the-RTU-side</i> or <i>point-of-time-when-command-shall-be-issued-at-the-RTU-side</i> , and Com.type = <i>EXC</i> and T value not a future time.	<i>T-out-of-range</i>
Com.type not equal to <i>EXC</i> or <i>IHC</i> .	<i>command-type-out-of-range</i>
Data not equal to corresponding parameter in first transaction if data <> empty in CBXC.	<i>EXC-command-different-from-CBXC</i>
RESPONDER UE not able to perform the requested function, due to any reason other than those listed above.	<i>remote-service-user-unavailable</i>
For security class 2: The received authentication code >< the generated authentication code based in the received data.	<i>invalid-authentication-code-received</i>
For security class 3: The received checksum >< the generated checksum during the decipherment.	<i>decipherment-error</i>

Errors in the **A-Command-Transfer** *cnf.* primitive (detected by the INITIATOR part):

Error	Action in INITIATOR part of FU
Mismatch between Gtype/Gnr/Index1/Index2/Data in primitive and Gtype/Gnr/Index1/Index2/Data in corresponding <i>A-Command-Transfer req.</i> , and Result = <i>result-ok</i> .	Terminate the FU invocation locally, registering the error.
Mismatch in Time mode according to rules described in chapter 7.6.3.1.2.	Terminate the FU invocation locally, registering the error.
Com.type in corresponding <i>req.</i> = <i>IXC</i> , and Com.type in primitive not equal to <i>EXR</i> , and Result = <i>result-ok</i> .	Terminate the FU invocation locally, registering the error.
Com.type in corresponding <i>req.</i> = <i>CBXC</i> , and Com.type in primitive not equal to <i>CBR</i> , and Result = <i>result-ok</i> .	Terminate the FU invocation locally, registering the error.
Com.type in corresponding <i>req.</i> = <i>EXC</i> , and Com.type in primitive not equal to <i>EXR</i> , and Result = <i>result-ok</i> .	Terminate the FU invocation locally, registering the error.
Com.type in corresponding <i>req.</i> = <i>IHC</i> , and Com.type in primitive not equal to <i>IHR</i> , and Result = <i>result-ok</i> .	Terminate the FU invocation locally, registering the error.
Length of Data parameter in ACTRQ > 0 and length of Data parameter in ACTRQ >< length of Data parameter in ACTC, and Result = <i>result-ok</i> .	Terminate the FU invocation locally, registering the error.
Result = <i>result-ok</i> , but one or more Quality codes >< OK	Terminate the FU invocation locally, registering the error.
Result >< <i>result-ok</i> .	Terminate the FU invocation locally, registering the error. Do not assume that the command has been accepted or executed.
For security class 2: The received authentication code >< the generated authentication code based in the received data.	Terminate the FU invocation locally, registering the error.
For security class 3: The received checksum >< the generated checksum during the decipherment.	Terminate the FU invocation locally, registering the error.

Section 8 : THE RESTART FUNCTION GROUP

TABLE OF CONTENTS

8.	THE RESTART FUNCTION GROUP	8-1
8.1.	Restart Reconfigure FU	8-1
8.1.1.	Function	8-1
8.1.2.	Coordination rules	8-1
8.1.2.1.	Association usage	8-1
8.1.2.2.	Relation to other FUs	8-2
8.1.2.2.1.	Invoking FUs	8-2
8.1.2.2.2.	Invoked FUs	8-2
8.1.2.2.3.	Disrupting FUs	8-2
8.1.2.2.4.	Disrupted FUs	8-2
8.1.2.3.	Invocation	8-3
8.1.2.3.1.	Prerequisites	8-3
8.1.2.3.2.	Restrictions	8-3
8.1.2.3.3.	Invoking events	8-3
8.1.2.4.	Termination	8-4
8.1.2.4.1.	Orderly termination	8-4
8.1.2.4.2.	Disruption	8-4
8.1.3.	Procedures	8-4
8.1.3.1.	EASE service primitives	8-4
8.1.3.1.1.	Action sequence	8-5
8.1.3.1.2.	Parameter values	8-8
8.1.3.2.	Error handling	8-8
8.1.3.2.1.	FU disruption	8-8
8.1.3.2.2.	Illegal invocation attempt	8-8
8.2.	Restart Reactivate FU	8-9
8.2.1.	Function	8-9
8.2.2.	Coordination rules	8-9
8.2.2.1.	Association usage	8-9
8.2.2.2.	Relation to other FUs	8-10
8.2.2.2.1.	Invoking FUs	8-10
8.2.2.2.2.	Invoked FUs	8-10
8.2.2.2.3.	Disrupting FUs	8-10
8.2.2.2.4.	Disrupted FUs	8-10
8.2.2.3.	Invocation	8-11
8.2.2.3.1.	Prerequisites	8-11
8.2.2.3.2.	Restrictions	8-11
8.2.2.3.3.	Invoking events	8-11
8.2.2.4.	Termination	8-12
8.2.2.4.1.	Orderly termination	8-12
8.2.2.4.2.	Disruption	8-12
8.2.3.	Procedures	8-12
8.2.3.1.	EASE service primitives	8-12
8.2.3.1.1.	Action sequence	8-13
8.2.3.1.2.	Parameter values	8-14
8.2.3.2.	Error handling	8-14
8.2.3.2.1.	FU disruption	8-14
8.2.3.2.2.	Illegal invocation attempt	8-15

8. THE RESTART FUNCTION GROUP

8.1. Restart Reconfigure FU

Type: Composite.

8.1.1. Function

A *Restart Reconfigure FU* invocation adheres to the following procedure:

1. The INITIATOR UE disrupt running data transfer FU's for the same RESPONDER system.
2. The INITIATOR UE invokes the *Dynamic Association FU* in order to create an association for later use by the *Group Configuration FU*, if such association does not already exist.
3. The INITIATOR UE invokes the *Group Management FU* for function *Delete-all-groups*, in order to clear the reconfigurable part of the CS(R).
4. For each group in the CS(I), the INITIATOR UE invokes the *Group Configuration FU* for that group with data from the CS(I), in order to bring the CS(R) into accordance with the CS(I).
5. The INITIATOR UE invokes the *Restart Reactivate FU*, in order to grant permissions to send those data that the INITIATOR UE expects to receive periodically or unsolicited.
6. The INITIATOR UE eventually terminates the *Dynamic Association FU* invocation in an orderly manner.
7. The INITIATOR UE processes errors encountered during phases 2 through 5.

8.1.2. Coordination rules

8.1.2.1. Association usage

The Elcom interactions that are part of the *Group Management FU* and *Group Configuration FU* invocations that are invoked by the *Restart Reconfigure FU* are conveyed by an association with the characteristics as specified for the *Group Management FU* and the *Group Configuration FU*.

8.1.2.2. Relation to other FUs

8.1.2.2.1. Invoking FUs

The *Restart Reconfigure FU* may be invoked by the following FUs:

- *Permanent Association FU*
- *Dynamic Association FU*

8.1.2.2.2. Invoked FUs

The *Restart Reconfigure FU* may invoke the following FUs:

- *Dynamic Association FU*
- *Group Management FU*
- *Group Configuration FU*
- *Restart Reactivate FU*

8.1.2.2.3. Disrupting FUs

The *Restart Reconfigure FU* shall not be disrupted. Disruption of supporting FU invocations are treated as errors within a *Restart Reconfigure FU* invocation, but do not disrupt it.

8.1.2.2.4. Disrupted FUs

The *Restart Reconfigure FU* may disrupt

- *Periodic Data Transfer FU*
- *Requested Data Transfer FU*
- *Unsolicited Data Transfer FU*
- *Unsolicited Mixed Data Transfer FU*
- *Supervisory Control Data Transfer FU*

8.1.2.3. Invocation

8.1.2.3.1. Prerequisites

No previous or concurrent FU invocations are required.

8.1.2.3.2. Restrictions

For any given INITIATOR/RESPONDER UE combination, a *Restart Reconfigure FU* invocation is not allowed, if:

- *Restart Reconfigure FU* is already running for the INITIATOR/RESPONDER UE combination, or:
- *Restart Reactivate FU* is already running for the INITIATOR/RESPONDER UE combination.¹

8.1.2.3.3. Invoking events

The INITIATOR part of the *Restart Reconfigure FU* may in principle be invoked whenever the INITIATOR UE determines that it is appropriate. At least, such invocation shall occur upon the following events:

1. If communicating in the Elcom-83 compatible mode:
The restart code *Restart, group management lost* being received by the INITIATOR part of a *Permanent Association FU* or *Dynamic Association FU* invocation.
2. If communicating in normal (Elcom-90) mode:
Configuration control field error² being detected by the INITIATOR part of a *Permanent Association FU* or *Dynamic Association FU* invocation.
3. Whenever re-invocation of the *Restart Reconfigure FU* is to be made, because of errors in FUs invoked by the *Restart Reconfigure FU*, subject to local decision in the INITIATOR UE.

The *Restart Reconfigure FU* is of the composite type. Consequently, there exists no specific RESPONDER part of the *Restart Reconfigure FU*, with an associated specific invoking event.

¹Consequently, any event triggering *Restart Reconfigure FU* or *Restart Reactivate FU* invocation occurring within the *Restart Reconfigure FU* or *Restart Reactivate FU* itself (triggered by an *Dynamic Association FU* or *Permanent Association FU* invocation) will have to be either ignored or deferred until the primary *Restart Reconfigure FU* or *Restart Reactivate FU* invocation is terminated.

²See chapter 5.5 "Group configuration integrity control".

8.1.2.4. Termination

8.1.2.4.1. Orderly termination

Orderly termination of the *Restart Reconfigure FU* is not defined³. Unexpected orderly termination of any FU that the *Restart Reconfigure FU* invokes is considered as an error within the *Restart Reconfigure FU* invocation.

8.1.2.4.2. Disruption

A *Restart Reconfigure FU* invocation shall not be disrupted. See chapter 8.1.2.2.3 "Disrupting FUs", above.

8.1.3. Procedures

8.1.3.1. EASE service primitives

The *Restart Reconfigure FU* is of the composite type, having no specific EASE service primitive sequence associated with it.

³This means that any *Restart Reconfigure FU* invocation will have to run its course, until it terminates itself.

8.1.3.1.1. Action sequence

The normal action sequence is partitioned into 7 phases:

Phase 1: Disrupting of running FU invocations.

If any *Data Transfer FU*⁴ invocation is running for the same RESPONDER system, it has to be disrupted.

Phase 2: Establishing the association for *Group Configuration FU* invocations.

If an association with the desired characteristics does not already exist between the INITIATOR UE and the RESPONDER UE:

If the prerequisites and restrictions that apply to the *Dynamic Association FU* can be met:

The *Dynamic Association FU* is invoked by the INITIATOR UE, in order to establish an association with the desired characteristics between the INITIATOR UE and the RESPONDER UE.

If the *Dynamic Association FU* is running OK:

The *Restart Reconfigure FU* invocation enters phase 3.

If not:

The *Restart Reconfigure FU* invocation processes⁵ the error, then enters phase 7.

If not:

The *Restart Reconfigure FU* invocation processes the error, then enters phase 7.

Phase 3: Clearing the reconfigurable part of the CS(R).

If the prerequisites and restrictions that apply to a *Group Management FU* invocation for the group can be met:

The *Group Management FU* is invoked by the INITIATOR UE for function *delete-all-groups*, in order to clear the part of the CS(R) that may be modified via the Elcom interface.

If the *Group Management FU* is running ok:

The *Restart Reconfigure FU* enters phase 4.

If the *Group Management FU* is NOT running OK or NOT terminated normally:

The *Restart Reconfigure FU* invocation processes the error, then enter phase 7.

If not:

The *Restart Reconfigure FU* invocation processes the error, then enter phase 7.

⁴See "Disrupted FUs" for a detailed list of relevant FUs.

⁵Within the Restart Function Group, processing of errors may generally comprise a locally determined number of retries, on the part of the INITIATOR UE.

Phase 4: Re-establishing correct CS(R).

For each group indicated by the CS(I)⁶:

If the prerequisites and restrictions that apply to a **Group Configuration FU** invocation for the group can be met:

The **Group Configuration FU** is invoked by the INITIATOR UE for the group with data from CS(I), in order to configure the group in the CS(R).

If the **Group Configuration FU** is running ok:

The **Restart Reconfigure FU** enters phase 5.

If the **Group Configuration FU** is NOT running OK or NOT terminated normally:

The **Restart Reconfigure FU** invocation processes the error, then enter phase 7.

If not:

The **Restart Reconfigure FU** invocation processes the error, then enter phase 7.

Phase 5: Re-activating periodic and unsolicited data transfers.

If the prerequisites and restrictions that apply to the **Restart Reactivate FU** can be met:

The **Restart Reactivate FU** is invoked by the INITIATOR UE, in order to grant to the RESPONDER UE permission to send those periodic and unsolicited data that the INITIATOR UE expects⁷.

If the **Restart Reactivate FU** is running ok:

The **Restart Reconfigure FU** enters phase 6.

If the **Restart Reactivate FU** is NOT running OK or NOT terminated normally:

The **Restart Reconfigure FU** invocation processes the error, then enter phase 7.

If not:

The **Restart Reconfigure FU** invocation processes the error, then enter phase 7.

Phase 6: Releasing the association for the **Group Configuration FU**.

At a point in time subject to local decision in the INITIATOR UE⁸, the **Dynamic Association FU** invoked in phase 2 is orderly terminated. Errors occurring with this termination shall have no effect on any **Restart Reconfigure FU** invocation.

⁶This is to be interpreted as only encompassing those parts of the CS that currently are supposed to be configurable in the CS(R) via the Elcom interface. Groups with attribute **Persistent** = *true* in the CS(I), if any, shall be ignored.

⁷For example, as determined from a group state database in the INITIATOR UE.

⁸*Dynamic Association FU* termination shall not be triggered before phase 3 is completed. The INITIATOR UE may for example terminate the *Dynamic Association FU* after completion of that phase, immediately or after a time delay. In the latter case, a "release-on-demand" mechanism for the Elcom resources occupied by the *Dynamic Association FU* invocation should be implemented, allowing termination the *Dynamic Association FU* invocation ahead of delay time expiry.

Phase 7: Error handling.

If at least one serious⁹ error has been encountered during the current **Restart Reconfigure FU** invocation, the INITIATOR UE shall follow one of two courses of action:

1. Terminate the **Restart Reconfigure FU** invocation, flagging failure to re-initialize the RESPONDER UE.¹⁰
2. Terminate the **Restart Reconfigure FU** invocation. Wait for a locally determined amount of time, then re-invoke the **Restart Reconfigure FU**, from the beginning.
Perform the wait/re-invoke cycle until success or a locally determined max. number of invocations have been performed.
If still not successful:
Terminate, and flag the error, as described in 1., above.

⁹Subject to local decision in the INITIATOR UE.

¹⁰Further processing of re-initialization failure is a local issue within the INITIATOR UE, outside the scope of this document. However, the failure means that there is a persistent mismatch between the CS(I) and the CS(R), so some sort of corrective action should eventually be triggered. Such corrective action will typically be manual.

8.1.3.1.2. Parameter values

The FU is of the composite type. See description of component FUs.

8.1.3.2. Error handling

Errors occurring with EASE service primitives are not considered here, since the *Restart Reconfigure FU* is of the composite type.

Handling of errors occurring in FUs invoked by the *Restart Reconfigure FU* is described in chapter 8.1.3.1.1 "Action sequence", above.

8.1.3.2.1. FU disruption

The *Restart Reconfigure FU* can not be disrupted.

8.1.3.2.2. Illegal invocation attempt

FU not present:

If the *Restart Reconfigure FU* is not present in an INITIATOR UE:

Invocation requests are always generated locally; see chapter 8.2.2.3 "Invocation". Consequently, the handling of this type of error is a local issue, outside the scope of this document.

No specific RESPONDER part is defined for the *Restart Reconfigure FU*, since it is of the composite type.

FU present, but attempt illegal:

Attempts of multiple simultaneous invocations of the *Restart Reconfigure FU* for a given INITIATOR/RESPONDER UE combination is a local issue, outside the scope of this document. However, the INITIATOR UE must ensure that *Restart Reconfigure FU* failures are properly registered, since such failure signifies inconsistency between the CS(I) and the CS(R).

The same applies for attempts of invoking the *Restart Reconfigure FU* for an INITIATOR/RESPONDER UE combination for which the *Restart Reactivate FU* is already being running.

8.2. Restart Reactivate FU

Type: Composite.

8.2.1. Function

A *Restart Reactivate FU* invocation adheres to the following procedure:

1. For each group in the CS(I) that the INITIATOR UE determines appropriate, the INITIATOR UE invokes the *Unsolicited Data Transfer FU* or *Periodic Data Transfer FU*, in order to grant the RESPONDER UE permission to send data belonging to the group, unsolicited or periodically.
2. The INITIATOR UE processes errors encountered during phase 1.

8.2.2. Coordination rules

8.2.2.1. Association usage

The Elcom interactions that are part of the *Unsolicited Data Transfer FU* or *Periodic Data Transfer FU* invocations invoked by the *Restart Reactivate FU* are conveyed by associations with the characteristics as specified for the *Unsolicited Data Transfer FU* or *Periodic Data Transfer FU*.

8.2.2.2. Relation to other FUs

8.2.2.2.1. Invoking FUs

The *Restart Reactivate FU* may be invoked by the following FUs:

- *Permanent Association FU*
- *Dynamic Association FU*
- *Restart Reconfigure FU*

8.2.2.2.2. Invoked FUs

The *Restart Reactivate FU* may invoke the following FUs:

- *Unsolicited Data Transfer FU*
- *Periodic Data Transfer FU*

8.2.2.2.3. Disrupting FUs

The *Restart Reactivate FU* shall not be disrupted. Disruption of supporting FU invocations are treated as errors within a *Restart Reactivate FU* invocation, but do not disrupt it.

8.2.2.2.4. Disrupted FUs

The *Restart Reactivate FU* shall not disrupt any other FU.

8.2.2.3. Invocation

8.2.2.3.1. Prerequisites

No previous or concurrent FU invocations are required¹¹.

8.2.2.3.2. Restrictions

For any given INITIATOR/RESPONDER UE combination, a *Restart Reactivate FU* invocation is not allowed, if:

- *Restart Reactivate FU* is already running for the INITIATOR/RESPONDER UE combination, or:
- *Restart Reconfigure FU* is already running for the INITIATOR/RESPONDER UE combination.¹²

8.2.2.3.3. Invoking events

The INITIATOR part of the *Restart Reactivate FU* may in principle be invoked whenever the INITIATOR UE determines that it is appropriate. At least, such invocation shall occur upon the following events:

1. The restart code *Restart, spontaneous management lost* being received by the INITIATOR part of a *Permanent Association FU* or *Dynamic Association FU* invocation.
2. Whenever re-invocation of the *Restart Reactivate FU* is to be made, because of errors in FUs invoked by the *Restart Reactivate FU*, subject to local decision in the INITIATOR UE.

The *Restart Reactivate FU* is of the composite type. Consequently, there exists no specific RESPONDER part of the *Restart Reactivate FU*, with an associated specific invoking event.

¹¹ *Permanent Association FU* invocations are a requirement of the *Unsolicited Data Transfer FU* or *Periodic Data Transfer FU* invoked by the *Restart Reactivate FU*, not of the *Restart Reactivate FU* itself.

¹² Consequently, any event triggering *Restart Reactivate FU* or *Restart Reconfigure FU* invocation occurring within the *Restart Reactivate FU* or *Restart Reconfigure FU* itself (triggered by a *Dynamic Association FU* or *Permanent Association FU* invocation) will have to be either ignored or deferred until the primary *Restart Reactivate FU* or *Restart Reconfigure FU* invocation is terminated.

8.2.2.4. Termination

8.2.2.4.1. Orderly termination

Orderly termination of the *Restart Reactivate FU* is not defined¹³. Unexpected orderly termination of any FU that the *Restart Reactivate FU* invokes is considered an error within the *Restart Reactivate FU* invocation.

8.2.2.4.2. Disruption

A *Restart Reactivate FU* invocation shall not be disrupted. See chapter 8.1.2.2.3 "Disrupting FUs", above.

8.2.3. Procedures

8.2.3.1. EASE service primitives

The *Restart Reactivate FU* is of the composite type, having no specific EASE service primitive sequence associated with it.

¹³This means that any *Restart Reactivate FU* invocation will have to run its course, until it terminates itself.

8.2.3.1.1. Action sequence

The normal action sequence has 2 phases:

Phase 1: Granting the RESPONDER UE permissions to send.

For each group that is defined in the CS(I)¹⁴:

The INITIATOR UE examines its local state information regarding the group, and decides whether the RESPONDER UE should be granted permission to send data belonging to the group, either unsolicited or periodically.

If permission to send unsolicited should be granted:

If the prerequisites and restrictions that apply to a **Unsolicited Data Transfer FU** invocation for the group can be met:

The INITIATOR UE invokes the **Unsolicited Data Transfer FU** for the group.

If the **Unsolicited Data Transfer FU** is NOT running OK or phase 1¹⁵ of the **Unsolicited Data Transfer FU** invocation does NOT complete OK:

The **Restart Reactivate FU** invocation processes the error.

If not:

The **Restart Reactivate FU** invocation processes the error¹⁶.

If permission to send periodically should be granted:

If the prerequisites and restrictions that apply to a **Periodic Data Transfer FU** invocation for the group can be met:

The INITIATOR UE invokes the **Periodic Data Transfer FU** for the group.

If the **Periodic Data Transfer FU** is NOT running OK or phase 1¹⁷ of the **Periodic Data Transfer FU** invocation does NOT complete OK:

The **Restart Reactivate FU** invocation processes the error.

If not:

The **Restart Reactivate FU** invocation processes the error¹⁸.

¹⁴Groups that are fixedly predefined in the CS, if any, are included here, in contrast to what is the case for phase 1 of the *Restart Reconfigure FU*.

¹⁵OK completion of phase 1 of the *Unsolicited Data Transfer FU* is sufficient for success in this context.

¹⁶If the *Unsolicited Data Transfer FU* is already being invoked for the group in question, this is NOT considered an error.

¹⁷OK completion of phase 1 of the *Periodic Data Transfer FU* is sufficient for success in this context.

¹⁸If the *Periodic Data Transfer FU* is already being invoked for the group in question, this is NOT considered an error.

Phase 2: Error handling.

If at least one serious¹⁹ error has been encountered during the current **Restart Reactivate FU** invocation, the INITIATOR UE shall follow one of two courses of action:

1. Terminate the **Restart Reactivate FU** invocation, flagging failure to re-activate groups in the RESPONDER UE.²⁰
2. Terminate the **Restart Reactivate FU** invocation. Wait for a locally determined amount of time, then re-invoke the **Restart Reactivate FU**, from the beginning.
Perform the wait/re-invoke cycle until success or a locally determined max. number of invocations have been performed.
If still not successful:
Terminate, and report the error, as described in 1., above.

8.2.3.1.2. Parameter values

The FU is of the composite type. See description of component FUs.

8.2.3.2. Error handling

Errors occurring with EASE service primitives are not considered here, since the **Restart Reactivate FU** is of the composite type.

Handling of errors occurring in FUs invoked by the **Restart Reactivate FU** is described in chapter 8.1.3.1.1 "Action sequence", above.

8.2.3.2.1. FU disruption

The **Restart Reactivate FU** can not be disrupted.

¹⁹Subject to local decision in the INITIATOR UE.

²⁰Further processing of re-activation failure is a local issue within the INITIATOR UE, outside the scope of this document. However, the failure means that data for failing groups will not be transmitted by the RESPONDER UE as expected, so some sort of corrective action should eventually be triggered. Such corrective action will typically be manual.

8.2.3.2.2. Illegal invocation attempt

FU not present:

If the *Restart Reactivate FU* is not present in an INITIATOR UE:

Invocation requests are always generated locally; see chapter 8.2.2.3 "Invocation", above. Consequently, the handling of this type of error is a local issue, outside the scope of this document.

No specific RESPONDER part is defined for the *Restart Reactivate FU*; since it is of the composite type.

FU present, but attempt illegal:

Attempts of multiple simultaneous invocations of the *Restart Reactivate FU* for a given INITIATOR/RESPONDER UE combination is a local issue, outside the scope of this document. However, the INITIATOR UE must ensure that *Restart Reactivate FU* failures are properly registered, since such failure signifies that expected data will not be transmitted by the RESPONDER UE.

The same applies for attempts of invoking the *Restart Reactivate FU* for an INITIATOR/RESPONDER UE combination for which the *Restart Reconfigure FU* is already being running.

Section 9 : SECURITY

TABLE OF CONTENTS

9.	SECURITY	9-1
9.1	Introduction	9-1
9.2	Security Services	9-2
9.3	Security Mechanisms	9-3
9.4	Security classes and options	9-4
9.5	Suggested use of the security mechanisms	9-5
9.6	Definition of the security information field	9-7

9. SECURITY

NOTE!

The conventions described in this chapter are not widely adopted, and should be considered deprecated. For an alternative, please consider TR A6196: Securing Elcom-90 with TLS.

9.1 Introduction

An organisation must work out a *security policy* based on its needs for a secure transfer of information using ELCOM-90. The *security services* needed shall reflect the security policy. These services are realised by a set of *security mechanisms*.

ELCOM-90 does not offer any security mechanisms as part of the service element (EASE). The security mechanisms must reside in the User Element. However, ELCOM-90 enables two User Elements to *authenticate* each other and to employ security mechanisms during the information exchange.

ELCOM-90 offers facilities to use a set of security mechanisms. This may put some constraints on the security policy for an organisation using ELCOM-90. However, at the current stage the security mechanisms offered is envisaged to be more than adequate for most organisations.

The concepts, security services and mechanisms are based on ISO 7498 - 2 [17] which defines the ISO security architecture and ITU-T.509 (1988) [18] which defines the authentication framework for the directory services.

The implementation of a full set of security mechanisms as described above is optional. The options regarding security are defined in chapter 9.4.

9.2 Security Services

Three security services are regarded as important for ELCOM-90 User Elements. They are:

- peer entity authentication
- selective field connection integrity
- selective field confidentiality

The services will provide:

- | | |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| • Peer entity authentication | This service provides evidence that a user in a certain instance of communication is the one claimed. |
| • Selective field connection | This service provides proof of the integrity of one field of an integrity Protocol Data Unit exchanged with its peer entity. The service assures that data is not changed during the transfer. |
| • Selective field confidentiality | This service is used to protect field(s) of a Protocol Data Unit from unauthorised disclosure. |

Peer entity authentication is considered the most important security service. If peer entity authentication is selected, this will take place when an association is established.

Selective field connection integrity is likely to be important by some organisations when certain types of data are transferred, e.g. the transfer of Supervisory Control Data. ELCOM-90 provides functions to negotiate the use of this for the transfer of user data as well as for the exchange of group definition and group readout.

Selective field confidentiality can be used to protect sensitive information, e.g. information of economical value, from unauthorised inspection. ELCOM-90 provides functions to negotiate the use of this during the transfer of user data as well as for the exchange of group definition and group readout.

9.3 Security Mechanisms

The security mechanisms used to offer the security services defined in the previous chapter are:

- Authentication exchange Used to provide peer entity authentication.
- Field authentication Used to provide selective field connection integrity. An authentication code appended after the data of the field is used to provide this service. The code is generated on the basis of the selected field and not the complete message.
- Encipherment Used to provide the selective field confidentiality.

The authentication exchange will take place during the association establishment. An *authentication information field* is used.

Two modes are possible: one-way exchange and two-way exchange. One-way exchange will authenticate the initiator. When two-way exchange is used, the initiator is authenticated for the responder, and the responder is authenticated for the initiator.

When one-way exchange is selected, an authentication information field is sent by the initiator to the responder. No authentication information field is returned from the responder to the initiator.

When two-way exchange is selected, the initiator will send an authentication information field to the responder and the responder must return an authentication information field to the initiator.

The UEs must negotiate the use of field authentication and encipherment. This is done during association establishment. A *security options field* is sent from the initiator to the responder during association establishment.

The security options field and the authentication information field are sent transparently through the EASE. The use of field authentication and encipherment is transparent for the EASE, as well.

The authentication code is appended to the field subject to authentication. For ELCOM-90 the authentication code is an integral part of the field. This will reduce the available space for data. The authentication code will in most cases be 32 bits or 64 bits in length. The generation of the authentication code is the responsibility of the UE sending the data.

Certain fields may be subject to encipherment. The encipherment is the responsibility of the sending UE. The receiving UE must be able to detect an encipherment error. This is usually done with a checksum which may be included in the data subject to encipherment or appended after the enciphered data part. The resulting data field, being larger than the original data, must not exceed the maximum field length. Decipherment must be done by the receiving UE.

9.4 Security classes and options

ELCOM-90 defines 4 security classes. They are:

- Class 0: No security mechanisms are used. This is the default value.
- Class 1: Authentication exchange is employed. One-way or two-way exchange may be used.
- Class 2: Class 1 + use of field authentication.
- Class 3: Class 1 + use of encipherment.

The selected class is in effect for the association being established. It is not possible to use field authentication as well as encipherment for the same association. One UE may handle several associations and each association has its individual security class set.

The INITIATOR UE will as part of the association establishment:

- select the security class (the selection of a security class is not subject to negotiations)
- select one-way or two-way authentication exchange
- determine if field authentication *or* encipherment shall be used
- determine the data types for which field authentication or encipherment shall be used.

The fields (parameters) where field authentication or encipherment can be used are:

- the **Objid** parameter used in the *A-Def-Group* and *A-Get-Group* primitives.
- the **Data** parameter in the *A-Command-Transfer* primitive.
- the **Data** parameter in the *A-Data* and *A-Mixed-Data* primitives.

The content of the parameter in a single primitive invocation is subject to field authentication or encipherment. The resulting length of the parameter *after* authentication code generation or encipherment must *not* exceed the maximum size of the parameter.

9.5 Suggested use of the security mechanisms

This document defines *exchange of security information during association establishment*. It is outside the scope of this document to determine the content of the authentication information field and select the field authentication or encipherment algorithms. These aspects are subject to local conventions. However, in this chapter it is outlined how it is *possible* to use the security mechanisms.

The content of the authentication information field depends on the other options selected. It is possible to employ a Public Key CryptoSystem (PKCS) like RSA or a symmetric cryptosystem like DES for the authentication information field. The maximum size of the authentication information field is 64 octets.¹

When a symmetric cryptosystem is used, the whole field may be enciphered or just a part of the field.

The content of the field will depend on the security class selected.

In the simplest case for class 1 the authentication information field may contain (when a symmetric cryptosystem is used):

- INITIATOR UE id
- time stamp
- a random number
- an enciphered sub field based on the time stamp, the random number and a password of the INITIATOR UE.

The CCITT recommendation X.509 suggests several ways to do this.

The time stamp and the random number are important to avoid the same authentication field in two subsequent association calls; this to reduce the risk of masquerade (the pretence by an entity to be a different entity). It is non-trivial for one entity to tap the transmission and replay the message. A three-way hand-shake is the only mechanism to completely eliminate the risk of masquerade.

When class 2 or 3 is selected, *session keys* must be selected for the association. The session keys are used for field authentication or encipherment. These session keys are conveyed in the authentication information field. Two-way exchange is employed to provide for one session key in each direction.²

When field authentication is used, an authentication code is appended to the original data. This code will typically be 32 or 64 bits. It is generated with a DES-like algorithm or a hashing function.

Encipherment can be used in several ways. One important requirement is to detect an encipherment error. A checksum (in many ways like an authentication code) is appended to the data. Such a

¹If RSA shall be used, the recommended key length is 512 bits, i.e. 64 octets. This implies that the absolute minimum length of an RSA encrypted field is the maximum length of the authentication information field, i.e. the use of RSA is subject to limitations. When RSA is used, there is not enough room for a digital signature (i.e. the enciphered 64 octet field) and the information itself (e.g. time stamp, random number, used id). The authentication field has space for the enciphered field, only. This field can be used to convey information, but a checksum is necessary to maintain the integrity of the information. In addition at least one (sub-)field of the information must be conveyed in clear, as well. It is suggested to use the INITIATOR address (another parameter in the A-Connect primitive) for this purpose. The RSA-enciphered authentication information field may contain the following fields: INITIATOR address, time stamp, random number, password of the INITIATOR UE and a checksum.

²If RSA is employed, the sender should encipher the session key with the *public key of the receiver* (see [17]). However, the sender should encipher the whole field with the *secret key of the sender*. This would require an authentication field larger than 128 octets which is impossible with ELCOM-90. One possible way to solve this is first to encipher the whole field with the public key of the receiver and then the result with the secret key of the sender. In this way the receiver is assured that the information comes from the right entity and the sender is assured that only the right receiver can decipher the session keys conveyed in the authentication information field.

Appendix A : Encoding of user defined data parameters

TABLE OF CONTENTS

Encoding of user defined data parameters	11-1
A.1 Summary	11-1
A.2 Structure	11-2
A.3 User Data Types	11-3
A.3.1 Real Values	11-4
A.3.2 Discrete Values	11-4
A.3.3 Status Values	11-4
A.3.4 Logical breaker status values	11-5
A.3.5 Binary command values	11-5
A.3.6 Analogue setpoint values	11-5
A.3.7 Digital setpoint values	11-5
A.3.8 Text message strings	11-5
A.4 Quality codes	11-6
A.4.1 Real value quality codes	11-7
A.4.2 Discrete value quality codes	11-7
A.4.3 Status value quality codes	11-8
A.4.4 Logical breaker status value quality codes	11-8
A.4.5 Binary command value quality codes	11-8
A.4.6 Analogue setpoint value quality codes	11-8
A.4.7 Digital setpoint value quality codes	11-9
A.5 A-Data	11-10
A.6 A-Command-Transfer request	11-11
A.7 A-Command-Transfer response	11-12
A.8 A-Mixed-Data request	11-13
A.9 A-Connect request	11-14
A.10 A-Connect response	11-15

Encoding of user defined data parameters

A.1 Summary

This appendix describes the encoding of the "Data" parameters in the A-Data, A-Command and A-Mixed-Data primitives, and the "User Data" parameter in the A-Connect primitive

In the data transfer primitives, a "Data" parameter is available to the user. These primitives are:

- A-Data
- A-Command-Transfer
- A-Mixed-Data

In the connection establishment primitive, a "User-Data" parameter is available to the user. The primitive is:

- A-Connect

This appendix describes the use of the "Data" and the "User-Data" parameters of these primitives.

A.2 Structure

The octets in this appendix are numbered starting from 0 and increasing in order of transmission. The bits in an octet are numbered from 0 to 7, where bit 0 is the low-ordered bit.

All octets are numbered in decimal. All values are given in decimal when nothing else is stated. Codes are given in binary.

One-octet values are unsigned and two-octet integer values are signed when nothing else is stated. For signed values twos complement representation¹ is used.

Integer values represented in two octets have their least significant part stored in the octet with the highest octet no.

All arrays are octet arrays.

¹ This is a representation of signed integers on digital computers. If we use n bits to represent an integer, the numbers in the range $0 \dots (2^{n-1}-1)$ are represented in the obvious way. To represent the negative of any representable positive number we take what is called the twos complement of the representation of the positive number. Taking the twos complement of the representation of a negative number yields the representation of the corresponding positive number. See any basic book on computer architecture for the details. e.g. Chapter 4 in [22].

A.3 User Data Types

The current version of ELCOM has eight predefined types of data:

- Real (Measure) values²
- Discrete values
- Status values
- Logical breakers status values
- Binary command values
- Analogue setpoint values
- Digital setpoint values
- Text message strings

The type numbers from 100 and above are reserved for local conventions.

The numbers below 100 are reserved for predefined types.

Real (Measure) values are expressed in floating point format ,e.g. they could express a process measured value, a calculated value, a summation value etc..

Discrete values are expressed in two octet integer format ,e.g. they could express a transformer tap position or a number of active devices.

Status values are typical on/off-information, e.g. they could express the position of a switch. When a switch is bipolar, it could have one of the values on/off/between.

Logical breaker status values are status values describing the state of a feeder. Normally the values are calculated locally from the breaker's status values.

Binary command values are typical on/off information, e.g. they can be used to control a breaker to the wanted state.

Analogue setpoint values are setpoints expressed in floating point format. They are typically used as an input parameter for a regulator.

Digital setpoint values are setpoints expressed in two octet integer format. They are typically used as an input parameter for a regulator.

Text message strings are strings of 8 bit ASCII coded characters.

All values except text message strings are delivered together with a quality code denoting the validity and the origin of the data.

² In the ELCOM protocol the data type for real values is named "Measured Values" even as that data type may be used for others than process measured values (e.g. calculated values, estimated values etc.).

A.3.1 Real Values

Real (Measured) values are represented in a 32 bit floating point format:

Mantissa magnitude: 22 + 1 bits (See note).
 Sign of the number: 1 bit.
 Signed exponent: 9 bits.

The mantissa is always normalised, $0,5 \leq \text{mantissa} < 1$. The exponent base is 2. The exponent is biased with $2^{**}8$, i.e., 400 octal is added to the actual exponent, so that a standardised floating zero contains zero in all 32 bits.

One real value occupies 4 octets in a value field:

Octet 0		Octet 1		Octet 2		Octet 3	
31	30	22	21			0	
Sign	Exponent		Normalised Mantissa				

The range of a floating point number is approximately:
 $-10^{**}76$ through $+10^{**}76$

NOTE:

The one extra bit in the mantissa is the most significant, and is set to one if not all bits in the exponent are zero. It is removed in the floating format, and is therefore to be regarded as a "hidden" bit.

A.3.2 Discrete Values

Discrete values are represented in 16 bit twos complement integer format. The range is:

$$-32768 \leq I \leq 32767$$

The least significant octet is stored in the octet with the highest octet no.

A.3.3 Status Values

Status values are represented in binary format. As stated in A.3 they can have 3 values: ON, OFF and BETWEEN. They therefore occupy only the 2 low-ordered bits in an octet. The values are:

xx xxx x10 : ON
 xx xxx x01 : OFF
 xx xxx x11 : BETWEEN
 xx xxx x00 : Undefined

The remaining bits are used to express the quality code.

A.3.4 Logical breaker status values

Logical breaker status values are represented in binary format in 8 bits. As described for status values the logical breaker status values can represent a bipolar switch which makes it necessary to define the between state. The values are:

xx xxx x10	Connected to bus bar A
xx xxx x01	Not connected to bus bar A
xx xxx x11	Between
xx xxx x00	Not allowed
xx xx1 0xx	Connected to bus bar B
xx xx0 1xx	Not connected to bus bar B
xx xx1 1xx	Between
xx xx0 0xx	Not allowed
xx 10x xxx	Bus bar A and B connected together
xx 01x xxx	Bus bar A and B not connected together
xx 11x xxx	Between
xx 00x xxx	Not allowed

The quality code for logical breakers comes in a separate octet.

A.3.5 Binary command values

Binary command values are represented in binary format. They can have two values ON or OFF. The values are:

00 000 010	:	ON
00 000 001	:	OFF

The quality code for binary commands comes in a separate octet.

A.3.6 Analogue setpoint values

Same format as Real Values described in A.3.1.

A.3.7 Digital setpoint values

Same format as Discrete Values described in A.3.2.

A.3.8 Text message strings

The format of the text message strings is 8 bit ASCII where every character is stored in one octet. All values from 0 to 255 are allowed.

No quality code is transmitted with the text message strings.

A.4 Quality codes

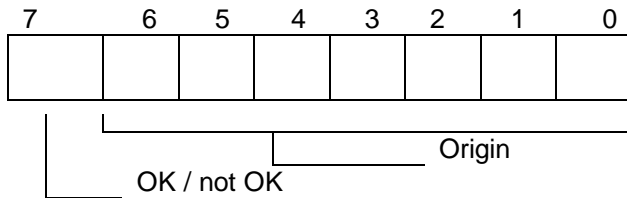
Each value transmitted, except the text message strings, is delivered together with a quality code denoting the quality and origin of the value.

For all values, except the status values, the quality code is delivered in a separate octet. For status values the quality code is coded in the most significant bits of the octet.

The most significant bit of the octet is used to express the validity of the corresponding value. If it is 0 the value is regarded OK, else it is regarded not OK.

For status values bit 2 - 6 are used to express the origin of the data.

For other values bit 0 - 6 are used to express the origin of the data.



OK code: 0 - OK
1 - Not OK

A.4.1 Real value quality codes

The present defined origin codes are:

x0 000 000	Measured
x0 000 100	Manually entered
x0 001 000	Estimated
x0 001 100	Computed
x0 010 000	Held

The meanings of these terms are indicated as follows:

Measured	A point is "measured" when its value is acquired by one of several possible measuring methods (e.g. scanning).
Manually entered	The value of a point is "manually entered" when its current value was provided by input from an operator or dispatcher.
Estimated	A point is "estimated" when its value is calculated by a state estimator program.
Computed	A point is "computed" when its value is the result of a calculation using other data (scanned, computed, and/or estimated) as input variables.
Held	A numerical point whose value is measured is "held" when the most recent update was unsuccessful and an old value is held in the data base.
OK/Not OK	<p>A point whose value is measured is "OK" when its value was acquired by the previous update; i.e., the point is not off-scan, and communications with the substation are successful.</p> <p>A manually entered value is always "OK".</p> <p>A point whose value is estimated by a state estimator is "OK" when the state estimator is running at its normally assigned frequency.</p> <p>A point whose value is computed is "OK" when all the independent data points from which it is computed (measured, manually entered and/or estimated values) are "OK".</p> <p>A point whose value is held is always "Not OK".</p>

A.4.2 Discrete value quality codes

The present defined origin codes are:

x0 000 000	Measured
x0 000 100	Manually entered
x0 001 000	Estimated
x0 001 100	Computed
x0 010 000	Held

The meanings of these terms are indicated in A.4.1.

A.4.3 Status value quality codes

The present defined origin codes are:

x0 000 0xx	Measured
x0 000 1xx	Manually entered
x0 001 0xx	Estimated
x0 001 1xx	Computed
x0 010 0xx	Held

Bit 0 - 1 denotes the status value.

The meanings of these terms are indicated in A.4.1.

A.4.4 Logical breaker status value quality codes

The present defined origin codes are:

x0 000 000	Measured
x0 000 100	Manually entered
x0 001 000	Estimated
x0 001 100	Computed
x0 010 000	Held

The meanings of these terms are indicated in A.4.1.

A.4.5 Binary command value quality codes

The present defined origin codes are:

x0 000 000	(OK)
x0 000 001	Object blocked at RTU side
x0 000 010	No connection to local device
x0 000 100	Command has illegal value
x0 000 101	Not authorised for supervisory control.

The origin codes gives an explanation for why a supervisory command could not be executed by the RESPONDER system. When the command has been handed over to the RESPONDER system's SCADA/EMS controlling function without failure, the value will be marked "OK".

When the command could not be handed over to the RESPONDER system's SCADA/EMS controlling function or was rejected, the value is "Not OK" and is marked with one of the origin codes.

A.4.6 Analogue setpoint value quality codes

The present defined origin codes are:

x0 000 000	(OK)
x0 000 001	Object blocked at RTU side
x0 000 010	No connection to local device
x0 000 101	Not authorised for supervisory control.

The meanings of these terms are indicated in A.4.5.

A.4.7 Digital setpoint value quality codes

The present defined origin codes are:

x0 000 000	(OK)
x0 000 001	Object blocked at RTU side
x0 000 010	No connection to local device
x0 000 101	Not authorised for supervisory control.

The meanings of these terms are indicated in A.4.5.

A.5 A-Data

Data : User data are of five types (See A.3):

Real (Measure)-Group	:	Floating Point Values
Discrete-Group	:	Integer Values
Status-Group	:	Binary Values
Logical Breaker		
Status Group	:	Binary Values
Text Message Group	:	ASCII Values

For the coding of each group, see A.3.1 - A.3.4 and A.3.8.

Each value must be considered together with a quality code denoting its validity (See A.4).

Structure for Real (Measure)-Group, Logical Breaker Status-Group and Discrete-Group:

Quality Code 1
Value 1
.
.
.
Quality Code n
Value n

Structure for Status-Group:

Quality Code and Status Value 1
.
.
.
Quality Code and Status Value n

Structure for Text Message-Group, text corresponding to one object:

Octet 1
.
.
.
Octet n

For security class 2: Append an authentication code to the Data field. The length of the authentication code is subject to local conventions.³ The length of the Data field must be adjusted accordingly.

For security class 3: A checksum must be appended and the Data field enciphered. The length of the checksum is subject to local conventions.⁴ The length of the Data field must be adjusted accordingly.

³The authentication code will typically be 32 or 64 bits long.

⁴The checksum will typically be 32 or 64 bits long.

A.6 A-Command-Transfer request

Data : User data are of three types (See A.3):

Binary command group
Analogue setpoint group
Discrete setpoint group

For the coding of each group see A.3.5 - A.3.7,
Quality code equals 0.

Structure for Analogue Setpoint-Group and Discrete Setpoint Group:

Quality Code 1
Value 1
.
.
Quality Code n
Value n

Structure for Binary Command-Group:

Quality Code 1
Value 1

For security class 2: Append an authentication code to the Data field. The length of the authentication code is subject to local conventions.⁵ The length of the Data field must be adjusted accordingly.

For security class 3: A checksum must be appended and the Data field enciphered. The length of the checksum is subject to local conventions.⁶ The length of the Data field must be adjusted accordingly.

⁵The authentication code will typically be 32 or 64 bits long.

⁶The checksum will typically be 32 or 64 bits long.

A.7 A-Command-Transfer response

Data : User data are of three types (See A.3):

Binary command group
Floating setpoint group
Discrete setpoint group

For the coding of each group see A.3.5 - A.3.7 and A.4.5 - A.4.7.

Structure for Analogue Setpoint-Group and Discrete Setpoint Group:

Quality Code 1
Value 1
.
.
Quality Code n
Value n

Structure for Binary Command-Group:

Quality Code 1
Value 1

For security class 2: Append an authentication code to the Data field. The length of the authentication code is subject to local conventions.⁷ The length of the Data field must be adjusted accordingly.

For security class 3: A checksum must be appended and the Data field enciphered. The length of the checksum is subject to local conventions.⁸ The length of the Data field must be adjusted accordingly.

⁷The authentication code will typically be 32 or 64 bits long.

⁸The checksum will typically be 32 or 64 bits long.

A.8 A-Mixed-Data request

Data : User data field. Consists of repetitive fields of the following parameters.

Gnr 1
Index 1
Delta T 1
Quality code and value 1
.
.
.
Gnr n
Index n
Delta T n
Quality code and value n

Gnr: Group number (2 octets) 0 > = value < = 32767
Gnr = 0 terminates the data field

Index: Object index (2 octets) 1 < = value < = 32767

Delta T: Delta time relative to T0, represented as milliseconds (2 octets) 0 < = value < = 32767

Quality code and Value: The coding follows the description given in A.3 for the following types:

- Real (Measure) values
- Status values
- Discrete values
- Logical breaker status values

For security class 2: Append an authentication code to the Data field. The length of the authentication code is subject to local conventions.⁹ The length of the Data field must be adjusted accordingly.

For security class 3: A checksum must be appended and the Data field enciphered. The length of the checksum is subject to local conventions.¹⁰ The length of the Data field must be adjusted accordingly.

⁹The authentication code will typically be 32 or 64 bits long.

¹⁰The checksum will typically be 32 or 64 bits long.

A.9 A-Connect request

User-Data: Octet 0: not used

Octet 1: Marking of spontaneous mode (ref. chap.5, "Spontaneous mode codes").

The security information field starts from octet 2 if present. The maximum length is 66 octets.

The encoding is described below:

Definition of the security information field:

Octet no.:

- | | |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | Length of security information field
(0 for security class 0;
if security class > 0 then $1 \leq \text{length} \leq 65$, where
$\text{length} = \text{length of Authentication information field} + 1$) |
| 1 | Security options field |
| 2-65 | Authentication information field |

Definition of the security options field:

Bit no.

- | | |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7,6,5,4 | Security class number. ¹¹
0000: Class 0
0001: Class 1
0010: Class 2
0011: Class 3
The other values are reserved for future use. |
| 3 | (=0) One-way authentication exchange.
(=1) Two-way authentication exchange. |
| 2 | Use field authentication/encipherment for objid in <i>A-Def-Group</i> and <i>A-Get-Group</i> |
| 1 | Use field authentication/encipherment for data in <i>A-Data</i> and <i>A-Mixed-Data</i> |
| 0 | Use field authentication/encipherment for data in <i>A-Command-Transfer</i> |

The security information field may be absent.
In this case security class 0 is chosen.

The length of the authentication information field may only be zero for security class 0.

If the authentication information field is not present, security class number must be set to 0 in the *security options field*.

The security information field must be present if additional user data shall be appended to the User-Data field, i.e. the security information length field is set to 0 if no security mechanisms shall be used.

¹¹ The Ekcin-90 Reference Version uses the security class 4 *internally* as a part of its TLS implementation. See TR A4124 for details.

A.10 A-Connect response

User-Data : Octet 0: Restart marking.
Restart marking = '0' (decimal 48) No Restart.
Restart marking = '1' (decimal 49) Reserved.
Restart marking = '2' (decimal 50) Restart, Spontaneous
management lost.
Restart marking = '3' (decimal 51) Restart, group configuration and
spontaneous management lost.
Octet 1 - 12: Array CF (12).
Control Field for checking the group configuration consistency . CF
(0-7) is used to transfer the time when the last group configuration
was accepted and stored in the responding system.

CF(0) =	Year	0 <= value <= 254
CF(1) =	Month	1 <= value <= 12
CF(2) =	Day	1 <= value <= 31
CF(3) =	Hour	0 <= value <= 24
CF(4) =	Minute	0 <= value <= 59
CF(5) =	Second	0 <= value <= 59
CF(6, ms),		
CF(7, ls) ¹² =	Millisec.	0 <= value <= 999

CF (8-11) is used by responder to return the result of checksum
calculations on the internal group configuration data structures.
Format and methods are implementation dependent.

CF(8)=	Checksum: MS octet of the 16-bit integer corresponding to element no. 8 in the CF parameter of the <i>A-Group Mgnt</i> and <i>A-Def-Group</i> primitives
CF(9)=	Checksum: LS octet of the 16-bit integer corresponding to element no. 8 in the CF parameter of the <i>A-Group Mgnt</i> and <i>A-Def-Group</i> primitives
CF(10)=	Checksum: MS octet of the 16-bit integer corresponding to element no. 9 in the CF parameter of the <i>A-Group Mgnt</i> and <i>A-Def-Group</i> primitives
CF(11)=	Checksum: LS octet of the 16-bit integer corresponding to element no. 9 in the CF parameter of the <i>A-Group Mgnt</i> and <i>A-Def-Group</i> primitives

Octet 13 - : Security information field.
The security information field starts from octet 13 if present. The
maximum length is 66 octets. The security information field is
present only if two-way authentication is requested. An exception is
when user data are appended after the security information field. In
this case the length of the security information field is 0. The
encoding is described below:

¹²ms = most significant, ls = least significant

Definition of the security information field:

Octet no.:

- 0 Length of security information field
(0 for security class 0;
if security class > 0 then $1 \leq \text{length} \leq 65$), where
length = length of Authentication information field+1)
- 1 Security options field
- 2-65 Authentication information field

The security options field shall be set to 00 000 000.¹³

The security information field must be present if additional user data shall be appended to the User-Data field, i.e. the security information length field is set to 0 if no security mechanisms shall be used.

The security information field must be present for security classes 1 (with two-way authentication selected), 2 or 3.

¹³ The Elcom-90 Reference Version uses the security options field with a security class of 4 internally, as part of its TLS implementation. See TR A4124 for details.

Appendix B : Guide lines for local conventions

TABLE OF CONTENTS

Guide lines for local conventions	12-1
-----------------------------------	------

Guide lines for local conventions

The ELCOM 90 conventions described in this technical report are designed to meet the most common communication requirements for telecontrol. The communication requirements that are to be fulfilled in an actual case may not be met by the functional units and data types described here. The actual organisation responsible for establishing the communication system can decide to establish local conventions that make use of the flexibility built into the Elcom 90 protocol to design special functional units, modify existing or just define new data types and status codes.

Local conventions means that two sides agree with each other that communication between them may use additional features, i.e. additional suffices, additional group types or additional quality codes. By doing this the organisation must be aware that their communication system may not be able to interoperate with other communication systems not obeying to the same local conventions. In general, local conventions should be avoided where ever possible.

In order to keep record of local conventions in use, and thereby preventing design of many solutions to solve the same needs, all local conventions shall be reported to SINTEF Energy Research for inclusion in a register. This register is available as a report from SINTEF Energy Research [15] to be ordered by any organisation that want to benefit from using proven solutions.

Appendix C : FU invocation hierarchy

TABLE OF CONTENTS

FU invocation hierarchy	13-1
-------------------------	------

FU invocation hierarchy

Name of FU:	See chapter:
<u>Permanent Association FU may invoke:</u> Restart Reconfigure FU Restart Reactive FU	5.6.2.2.2
<u>Dynamic Association FU may invoke:</u> Restart Reconfigure FU	5.7.2.2.2
<u>Test Association FU may invoke:</u>	(None)
<u>Group Configuration FU may invoke:</u> Group Management FU Group Definition FU	6.5.2.2.2
<u>Group Management FU may invoke:</u>	(None)
<u>Group Definition FU may invoke:</u>	(None)
<u>Group Readout FU may invoke:</u>	(None)
<u>Requested Data Transfer FU may invoke:</u>	(None)
<u>Periodically Requested Data Transfer FU may invoke:</u> Requested Data Transfer FU	7.2.2.2.2
<u>Periodic Data Transfer FU may invoke:</u> Unsolicited Mixed Data Transfer FU	7.3.2.2.2
<u>Unsolicited Data Transfer FU may invoke:</u> Unsolicited Mixed Data Transfer	7.4.2.2.2
<u>Unsolicited Mixed Data Transfer FU may invoke:</u>	(None)
<u>Supervisory Control Data Transfer FU may invoke:</u>	(None)
<u>Restart Reconfigure FU may invoke:</u> Dynamic Association FU Group Management FU Group Configuration FU Restart Reactive FU	8.1.2.2.2
<u>Restart Reactivate FU may invoke:</u> Unsolicited Data Transfer FU Periodic Data Transfer FU	8.2.2.2.2

Appendix D : FU disruption hierarchy

TABLE OF CONTENTS

FU disruption hierarchy	14-1
-------------------------	------

FU disruption hierarchy

Name of FU:	See subclause:
<u>Permanent Association FU may disrupt:</u> Test Association FU Group Configuration FU Group Management FU Group Definition FU Group Readout FU Requested Data Transfer FU Periodically Requested Data Transfer FU Periodic Data Transfer FU Unsolicited Data Transfer FU Unsolicited Mixed Data Transfer FU Supervisory Control Data Transfer FU	5.6.2.2.4
<u>Dynamic Association FU may disrupt:</u> 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	5.7.2.2.4
<u>Test Association FU may disrupt:</u>	(None)
<u>Group Configuration FU may disrupt::</u>	(None)
<u>Group Management FU may disrupt:</u> Group Configuration FU Requested Data Transfer FU Periodically Requested Data Transfer FU Periodic Data Transfer FU Unsolicited Data Transfer FU Unsolicited Mixed Data Transfer FU Supervisory Control Data Transfer FU	6.2.2.2.4
<u>Group Definition FU may disrupt:</u> Group Configuration FU Requested Data Transfer FU Periodic Requested Data Transfer FU Periodic Data Transfer FU Unsolicited Data Transfer FU Unsolicited Mixed Data Transfer FU Supervisory Control Data Transfer FU	6.3.2.2.4
<u>Group Readout FU may disrupt:</u>	(None)
<u>Requested Data Transfer FU may disrupt:</u>	(None)
<u>Periodically Requested Data Transfer FU may disrupt:</u>	(None)
<u>Periodic Data Transfer FU may disrupt:</u>	(None)

<u>Unsolicited Data Transfer FU may disrupt:</u>	(None)
<u>Unsolicited Mixed Data Transfer FU may disrupt:</u>	(None)
<u>Supervisory Control Data Transfer FU may disrupt:</u>	(None)
<u>Restart Reconfigure FU may disrupt:</u>	8.1.2.2.4
Periodic Data Transfer FU	
Requested Data Transfer FU	
Unsolicited Data Transfer FU	
Unsolicited Mixed Data Transfer FU	
Supervisory Control Data Transfer FU	
<u>Restart Reactivate FU may disrupt:</u>	(None)

Appendix E : Use of result code values

TABLE OF CONTENTS

Use of result code values	15-1
---------------------------	------

Use of result code values

The table below contains, for each of the defined Elcom-90 error result codes that may be generated by a UE, references to the sections and chapters of this document which specify the use of the code.

Other result codes must not be used!

Result code value	Use specified in chapter
authentication-failure	5.6.3.2.6; 5.7.3.2.6
CBXC-not-received-before-EXC	7.6.3.2.6
command-type-not-supported-by-A-service-user	7.6.3.2.6
command-type-out-of-range	7.6.3.2.6
config-buffer-overflow	6.3.3.1.2; 6.3.3.2.6
decipherment-error	6.3.3.2.6; 7.1.3.2.6; 7.5.3.2.6; 7.6.3.2.6
Dt-out-of-range	7.1.3.2.6
EXC-command-different-from-CBXC	7.6.3.2.6
gnr-out-of-range	6.2.3.2.6; 6.3.3.1.2; 6.3.3.2.6; 6.4.3.2.6; 7.1.3.2.6; 7.3.3.2.6; 7.4.3.2.6; 7.5.3.2.6; 7.6.3.2.6
group-exists	6.2.3.2.6
gsize-out-of-range	6.2.3.2.6
gtype-out-of-range	6.2.3.2.6; 6.3.3.1.2; 6.3.3.2.6; 6.4.3.2.6; 7.1.3.2.6; 7.3.3.2.6; 7.4.3.2.6; 7.6.3.2.6;
incompatible-security-options	5.6.3.2.6; 5.7.3.2.6
incompatible-versions	5.6.3.2.6; 5.7.3.1.3; 5.7.3.2.6
index-out-of-range	6.3.3.1.2; 6.3.3.2.6; 6.4.3.2.6; 7.1.3.2.6; 7.3.3.2.6; 7.4.3.2.6; 7.5.3.2.6; 7.6.3.2.6
invalid-authentication-code-received	6.3.3.2.6; 7.1.3.2.6; 7.5.3.2.6; 7.6.3.2.6
misbehaviour-of-local-service-user	5.6.3.2.4; 5.7.3.2.4; 5.8.3.2.4; 7.1.3.2.4; 7.3.3.2.4; 7.4.3.2.4; 7.4.3.2.5
misbehaviour-of-remote-service-user	7.4.3.2.6
no-answer-from-remote-system	5.6.3.2.4; 5.7.3.2.4

no-memory	6.2.3.2.6
not-deleteable	6.2.3.2.6
not-reconfigurable	6.3.3.1.2; 6.3.3.2.6
objid-unknown	6.3.3.1.2; 6.3.3.2.6
objlength-out-of-range	6.2.3.2.6; 6.3.3.1.2; 6.3.3.2.6
priority-class-out-of-range	6.2.3.2.6
remote-service-user-unavailable	5.8.3.2.4; 6.2.3.2.2; 6.2.3.2.4; 6.2.3.2.6; 6.3.3.2.2; 6.3.3.2.4; 6.3.3.2.6; 6.4.3.2.2; 6.4.3.2.4; 6.4.3.2.6; 7.1.3.2.2; 7.1.3.2.4; 7.1.3.2.6; 7.3.3.2.2; 7.3.3.2.4; 7.3.3.2.6; 7.4.3.2.2; 7.4.3.2.4; 7.6.3.2.2; 7.6.3.2.4; 7.6.3.2.6
security-is-not-supported-by-A-service-user	5.6.3.2.6; 5.7.3.2.6
spontaneous-transfer-not-initiated	7.3.3.2.3; 7.4.3.2.3
T-out-of-range	7.1.3.2.6; 7.5.3.2.6; 7.6.3.2.6
T0-out-of-range	7.1.3.2.6
time-mode-not-supported-by-A-service-user	7.6.3.2.6
time-mode-out-of-range	7.6.3.2.6

Appendix F : Communicating with Elcom-83 systems

TABLE OF CONTENTS

Communicating with Elcom-83 systems	16-1
-------------------------------------	------

Communicating with Elcom-83 systems

When designing an Elcom-90 based system to communicate with an Elcom-83 based system [1] – [7], the following limitations must be kept in mind:

1. The *Unsolicited Mixed Data Transfer FU* cannot be implemented in an Elcom-83 based system, because neither the *A-Mixed-Data* EASE service nor the *A-Mixed-Data-Error* EASE service is defined in Elcom-83.
2. The *Supervisory Control Data Transfer FU* cannot be implemented in an Elcom-83 based system, because the *A-Command-Transfer* EASE service is not defined in Elcom-83.
3. The following group types are not defined in Elcom-83:
 - *Logical-breaker-status-group*
 - *Binary-command-group*
 - *Analog-setpoint-group*
 - *Digital-setpoint-group*
 - *Text-message-group*
4. The following quality codes are not defined in Elcom-83:
 - *Computed*
 - *Held*
5. Elcom-83 based implementations always use the **restart code** mechanism (chapter 5.4) for signalling requests for group configuration updates. The mechanism described in chapter 5.5 is never used with Elcom-83.
An Elcom-83 responder may send a restart code when the connection is rejected due to incompatible versions. In this case the INITIATOR shall react according to the restart code with the highest value.
6. When accepting an association with an Elcom-83 based INITIATOR, an Elcom-90 based RESPONDER UE must answer with Version value in Version parameter = 0, in order to achieve association establishment.
7. Groups in Elcom-83 based systems may never be larger than the maximum number of object identifiers that may fit into a single definition/readout PDU: Group definitions/readouts spanning multiple Elcom transactions are not legal in Elcom-83.
8. Elcom-90 restriction on communication with an Elcom-83 based system: Both indices in *A-Def-Group req.* and *A-Get-Group req.* primitives must be equal to zero.
9. ELCOM-83 based systems may not use UTC as a base for time stamps, but e.g. local time compensated for daylight saving time. The initiating UE shall compensate for that.

Appendix G : Address formats

TABLE OF CONTENTS

Address Formats	17-1
-----------------	------

Address Formats

TCP/IP

The format of the Initiator and Acceptor address when TCP/IP is used, is shown in the following table, where one element represents one octet:

OCTET 1	X=LENGTH OF LOWER PART (=17)
	TCP_ID (=82 hex)
	AF_INET part one (see below)
	AF_INET part two (- " -)
	Port number (1. Octet)
	Port number (2. Octet)
	IP address (1. Octet)
	IP address (2. Octet)
	IP address (3. Octet)
	IP address (4. Octet)
	0
	0
	0
	0
	0
	0
	0
	0
	0
	y=length of A-suffix (=2)
	A-suffix (1. Octet)
Octet x+y+2 (=21)	A-suffix (2. Octet)

Remarks

- The layout of the TCP/IP address is based on a memory copy of a *struct sockaddr_in*, as used with Berkeley sockets. Although this is not the most space-efficient format, all implementations should use this format for maximum compatibility.
- The port number and the IP address are in network byte order (ie. Most significant byte first).
- UEs using the reference version of the provider should initialise the AF_INET field to a 16-bit representation of the compile-time constant AF_INET in the native (host) byte order of the system. The presence of eight zero bytes at the end of the IP address is also required.
- Other implementations should initialise the AF_INET field and padding to zero on output.
- All implementations should ignore the contents of the Port number field, the AF_INET field and the padding bytes on input, at all levels (provider and UE). Address validation should only be done on the 4 octets representing the IP address.
- The AF_INET bytes are used for handshaking between providers that support connection-level link supervision with TCP/IP.

X.25

For X.25 there are two legal formats, the old Elcom-83 format, and the new Elcom-90 format. An Elcom-90 Responder shall be able to support both formats, and an Elcom-90 Initiator shall be careful not to use the Elcom-90 format when communicating with an Elcom-83 system. The old Elcom-83 format is most used, even between Elcom-90 systems.

Elcom-83 format:

OCTET 1	X=LENGTH OF LOWER LEVEL PART
	DTE number digit 1 (ASCII)
	DTE number digit 2 (ASCII)
	.
	.
	.
	DTE number digit x (ASCII)
	y=length of A-suffix (=2)
	A-suffix octet 1 (ASCII)
Octet x+y+2	A-suffix octet 2 (ASCII)

Elcom-90 format:

In the Elcom-90 X.25 format the address information is sent as BCD digits (i.e. in the way X.25 expects it). The length of the *Call Address Field* is the number of bytes, not the number of digits. If there is an odd number of BCD digits, the last nibble of the last byte has the value of 0xF (hex), which is the "padding value".

Octet 1	x=length of lower level part	
	X.25_ID (=80 hex)	
	DTE number digit 1 (BCD)	DTE number digit 2 (BCD)
	DTE number digit 3 (BCD)	DTE number digit 4 (BCD)
	.	.
	.	.
	.	.
	DTE number digit n-1 (BCD)	DTE number digit n (BCD) (or 0xF)
	y=length of A-suffix (=2)	
	A-suffix octet 1 (ASCII)	
Octet x+y+2	A-suffix octet 2 (ASCII)	

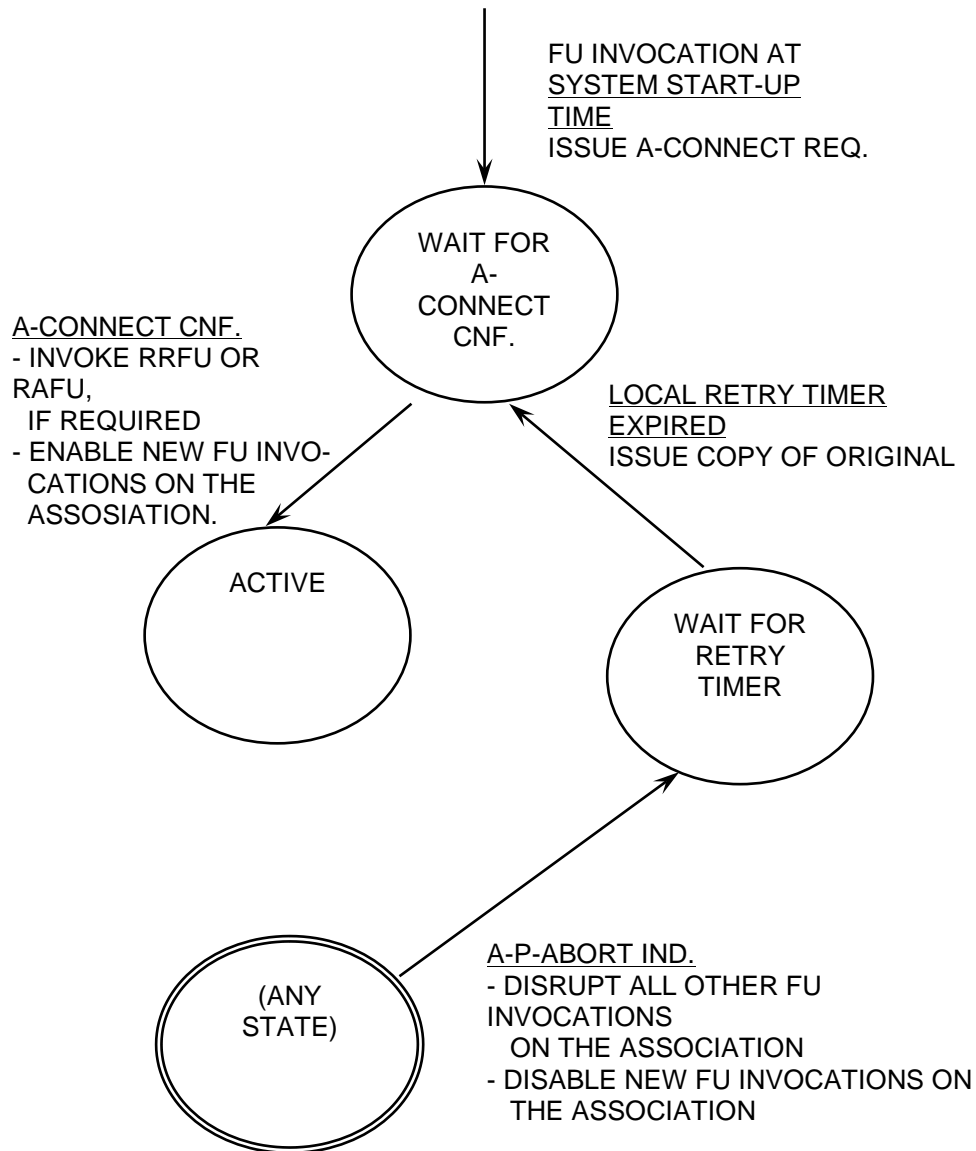
Appendix H : State diagram descriptions of functional units

TABLE OF CONTENTS

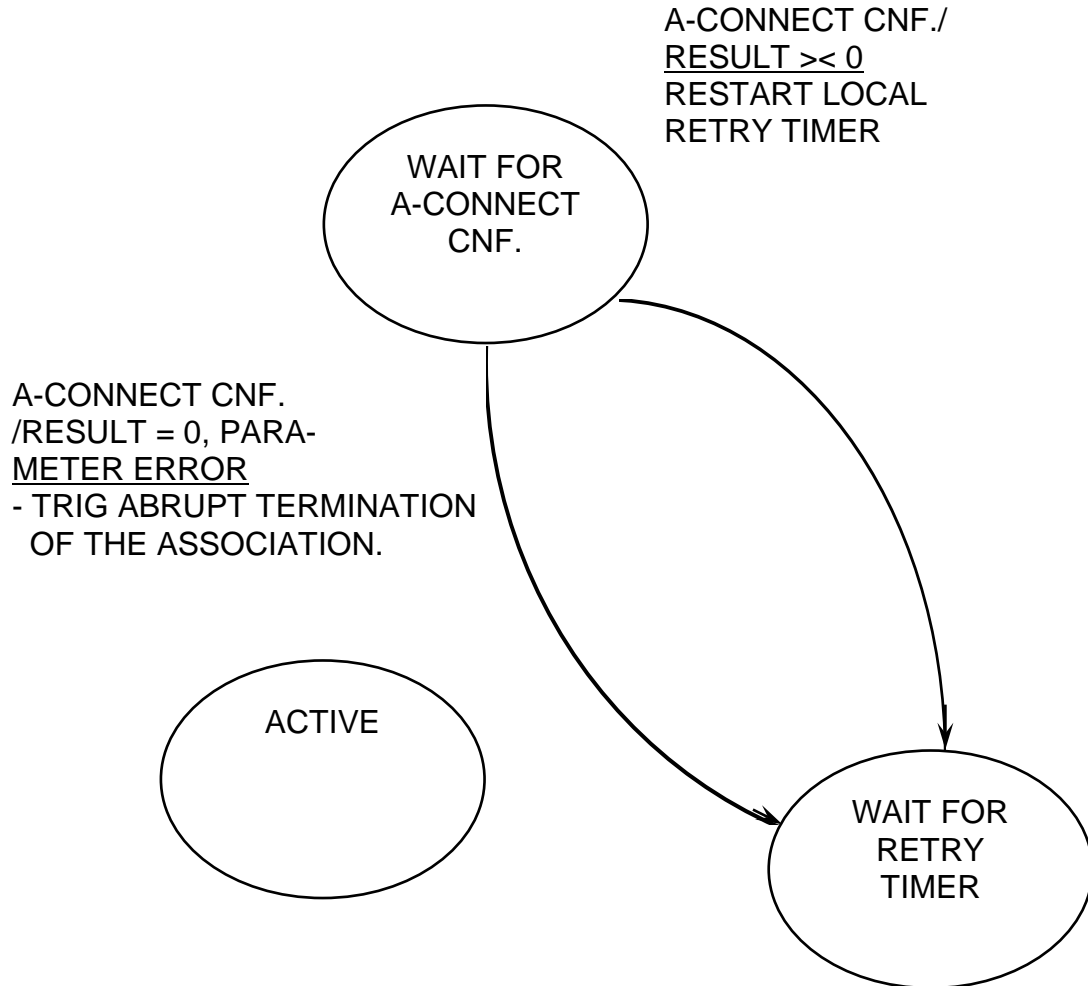
State diagram descriptions of functional units	18-1
1. Permanent Association FU – APFU	18-1
2. Dynamic Association FU – ADFU	18-5
3. Test Association FU – ATFU	18-7
4. Group Management FU – GMFU	18-9
5. Group Definition FU – GDFU	18-13
6. Group Readout FU – GRFU	18-17

State diagram descriptions of functional units

1. Permanent Association FU – APFU

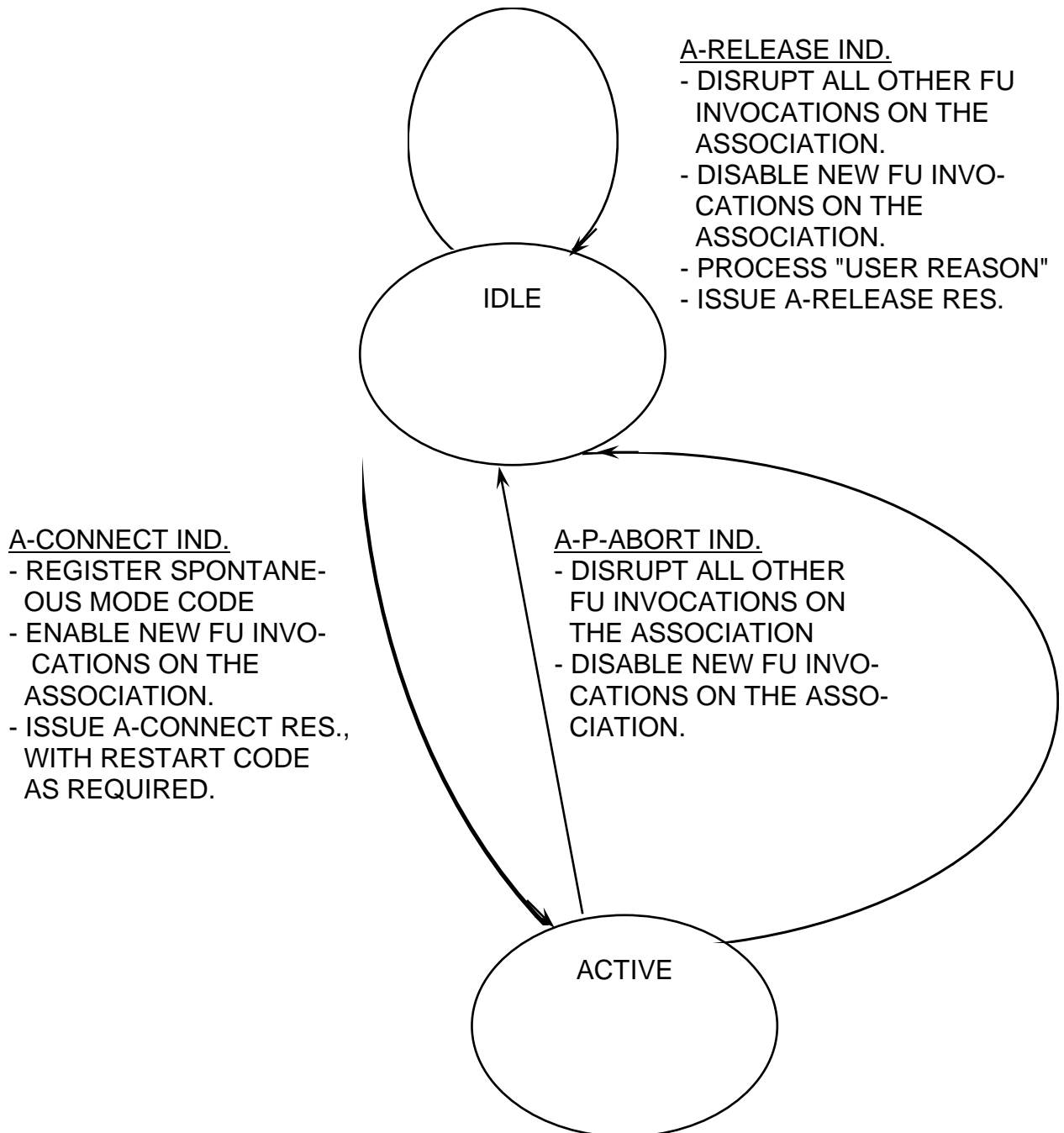


APFU-01: Initiator part, error-free execution

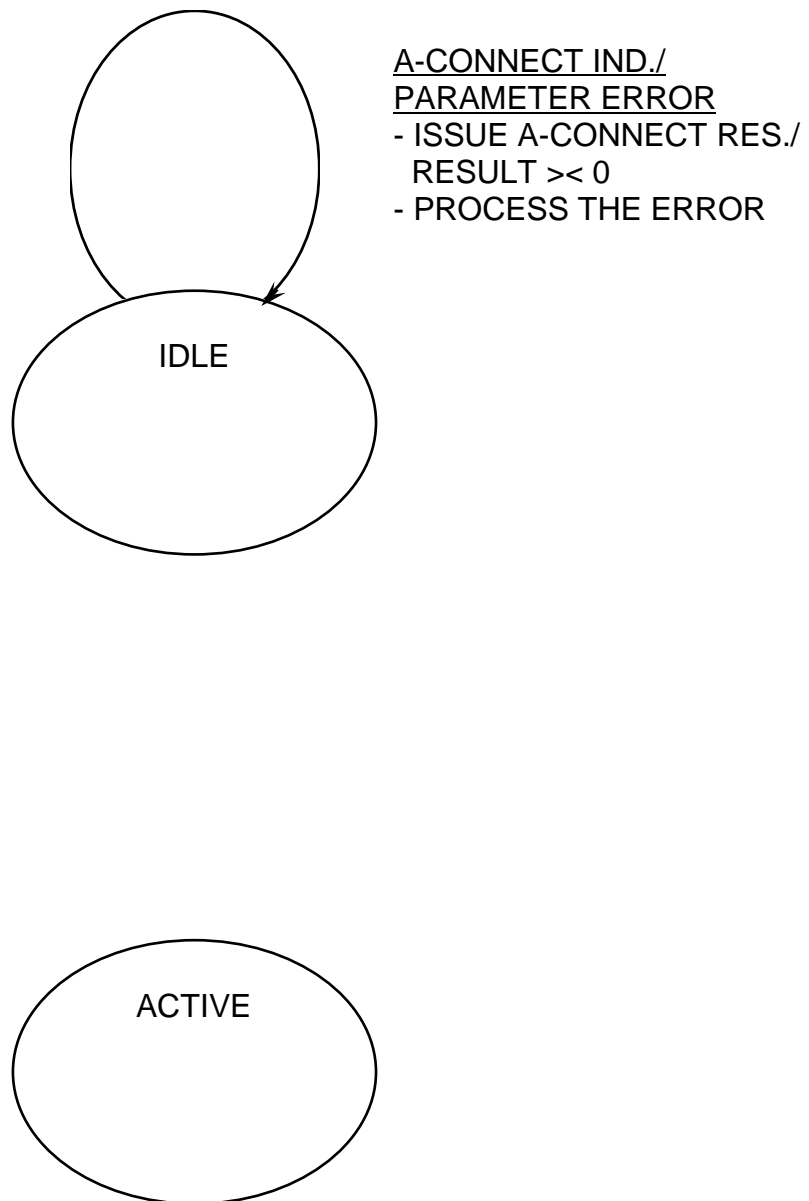


APFU-02: Initiator part, error handling

A-P-ABORT IND.

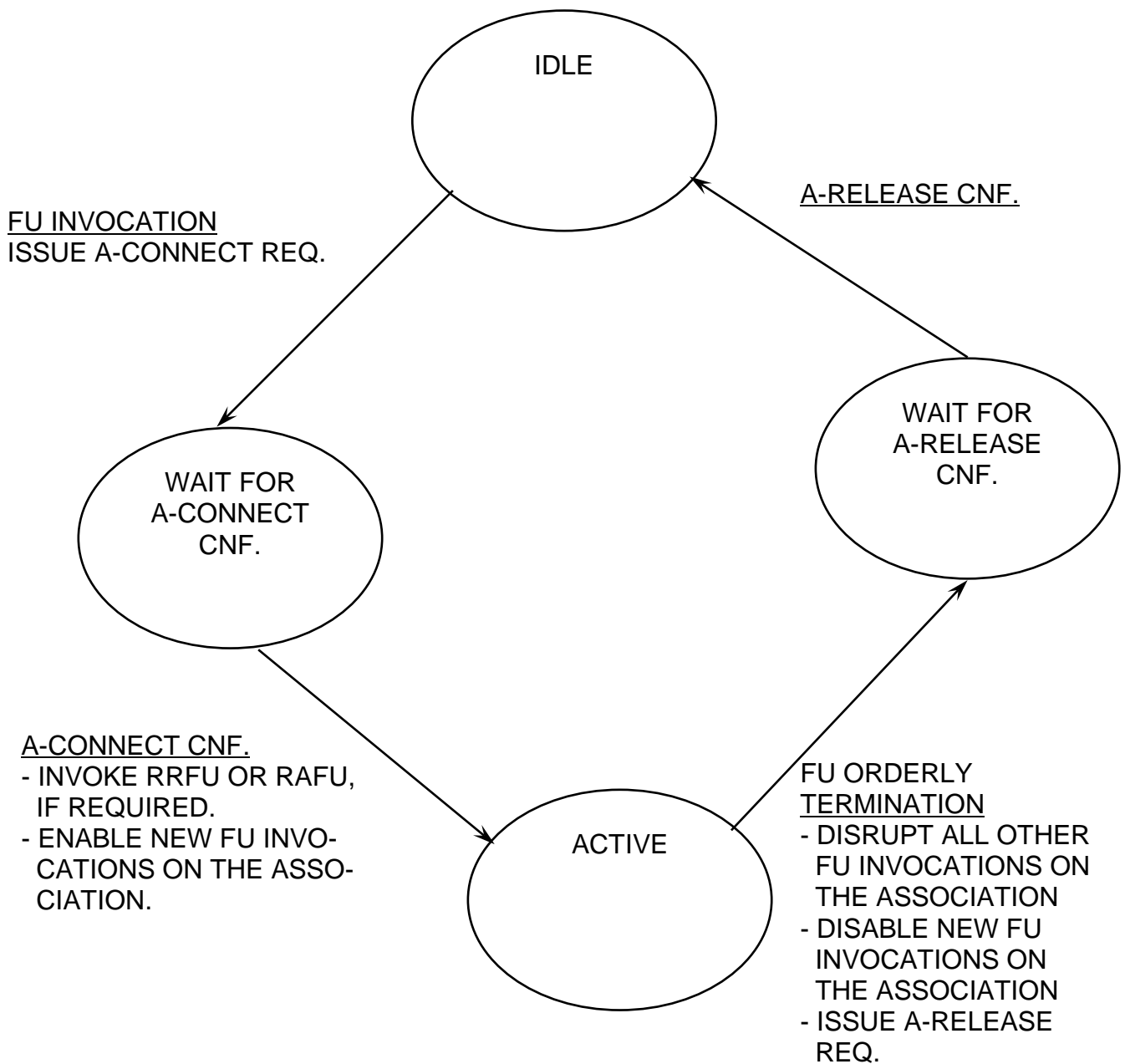


APFU-03: Responder part of the APFU/ADFU error free execution

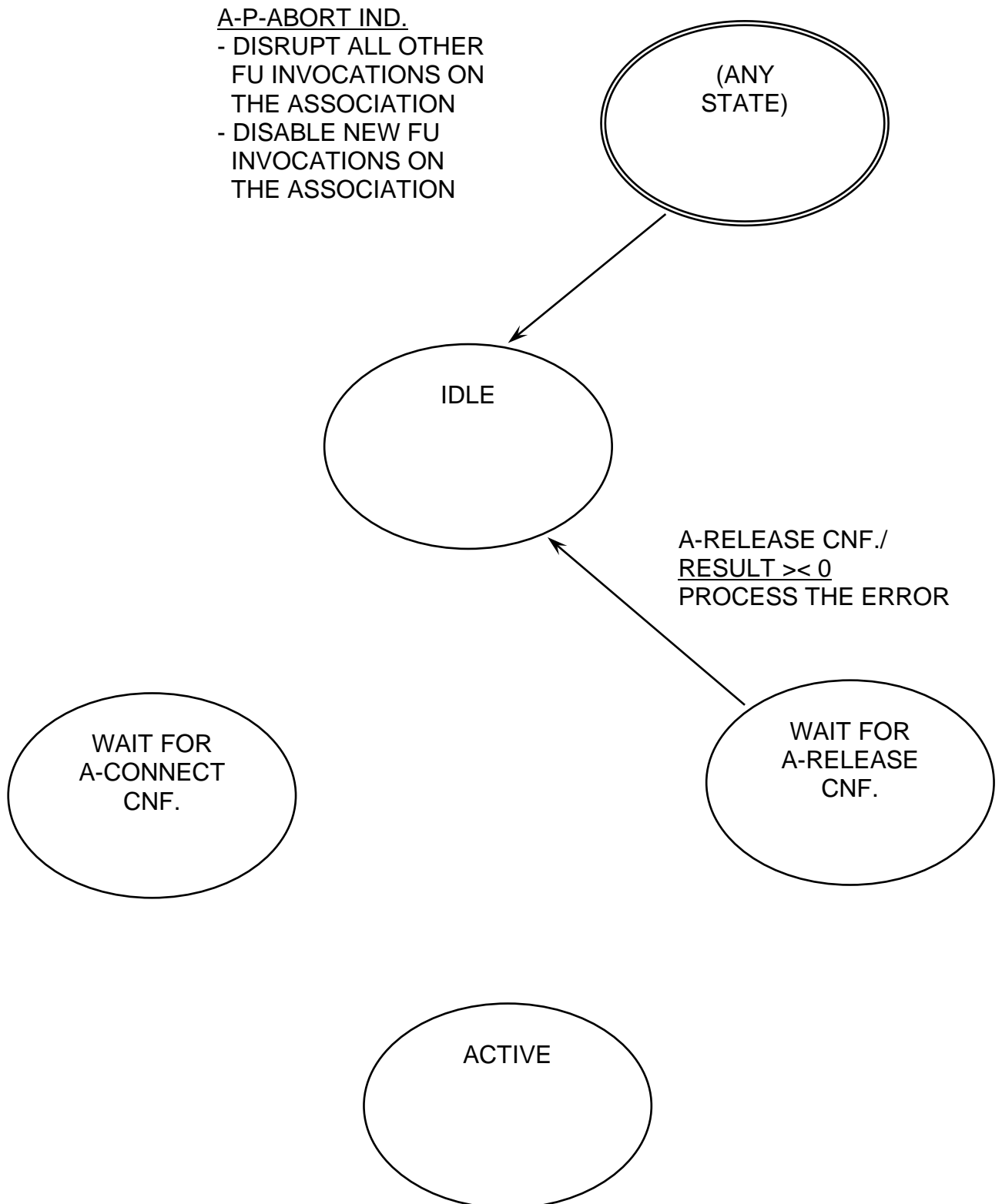


APFU-04: Responder part of the APFU/ADFU error handling

2. Dynamic Association FU – ADFU

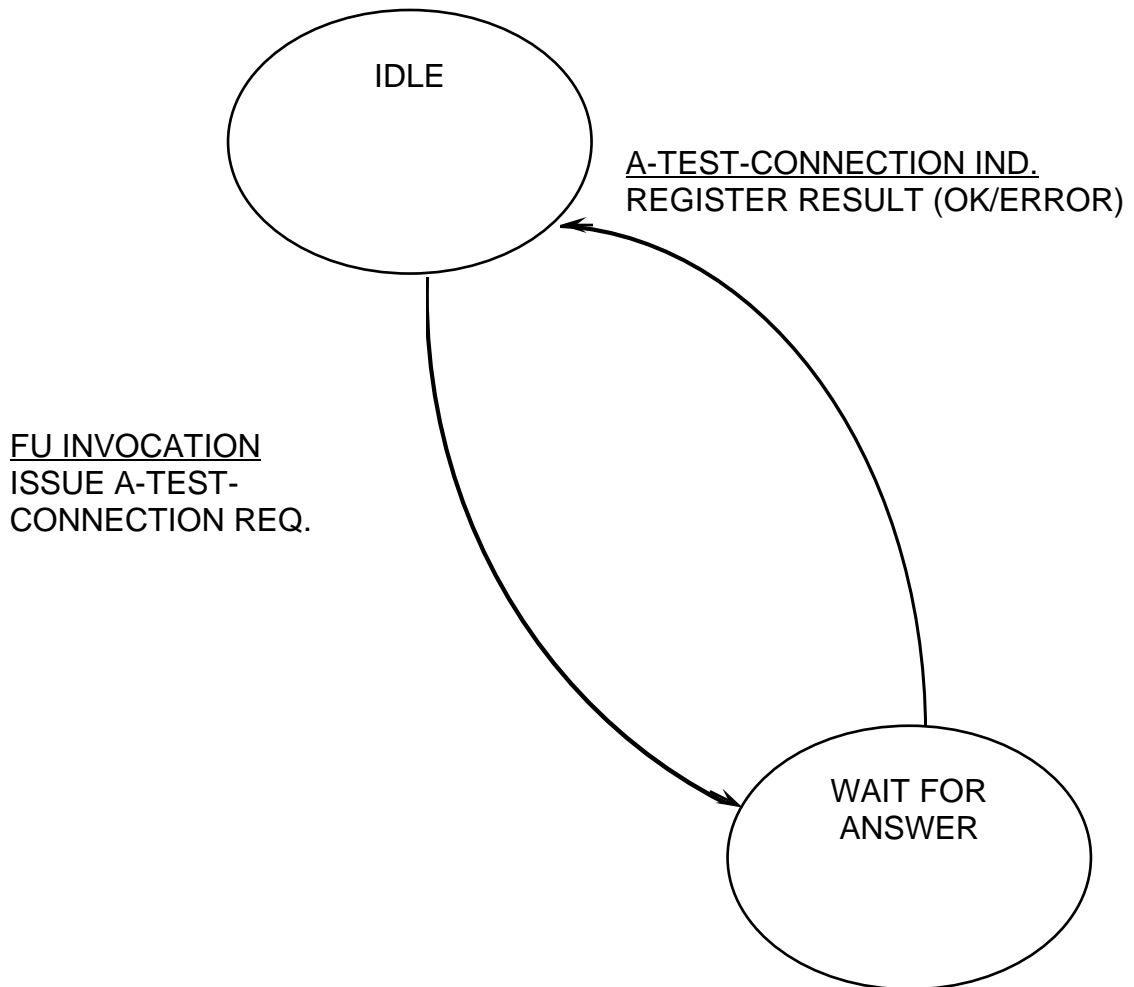


ADFU-01: Initiator part, error-free execution (responder part: Identical to APFU-03)

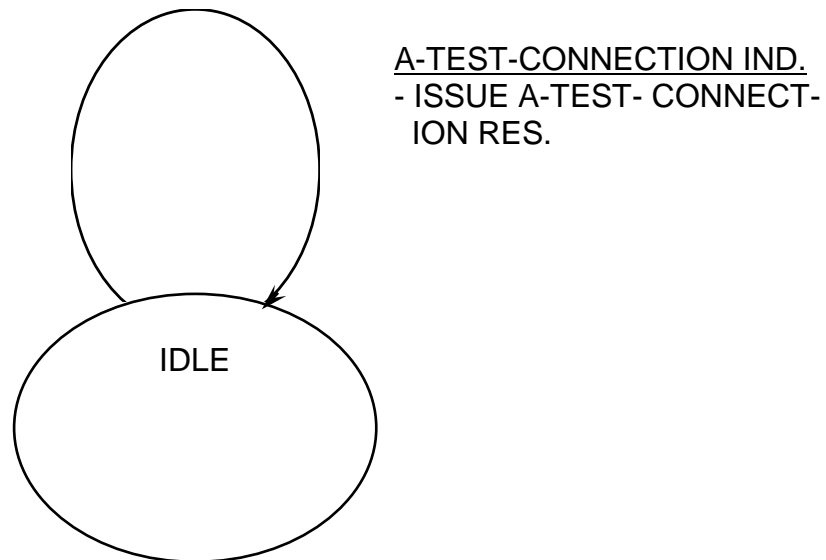


ADFU-02: Initiator part, error handling (responder part: Identical to APFU-04)

3. Test Association FU – ATFU

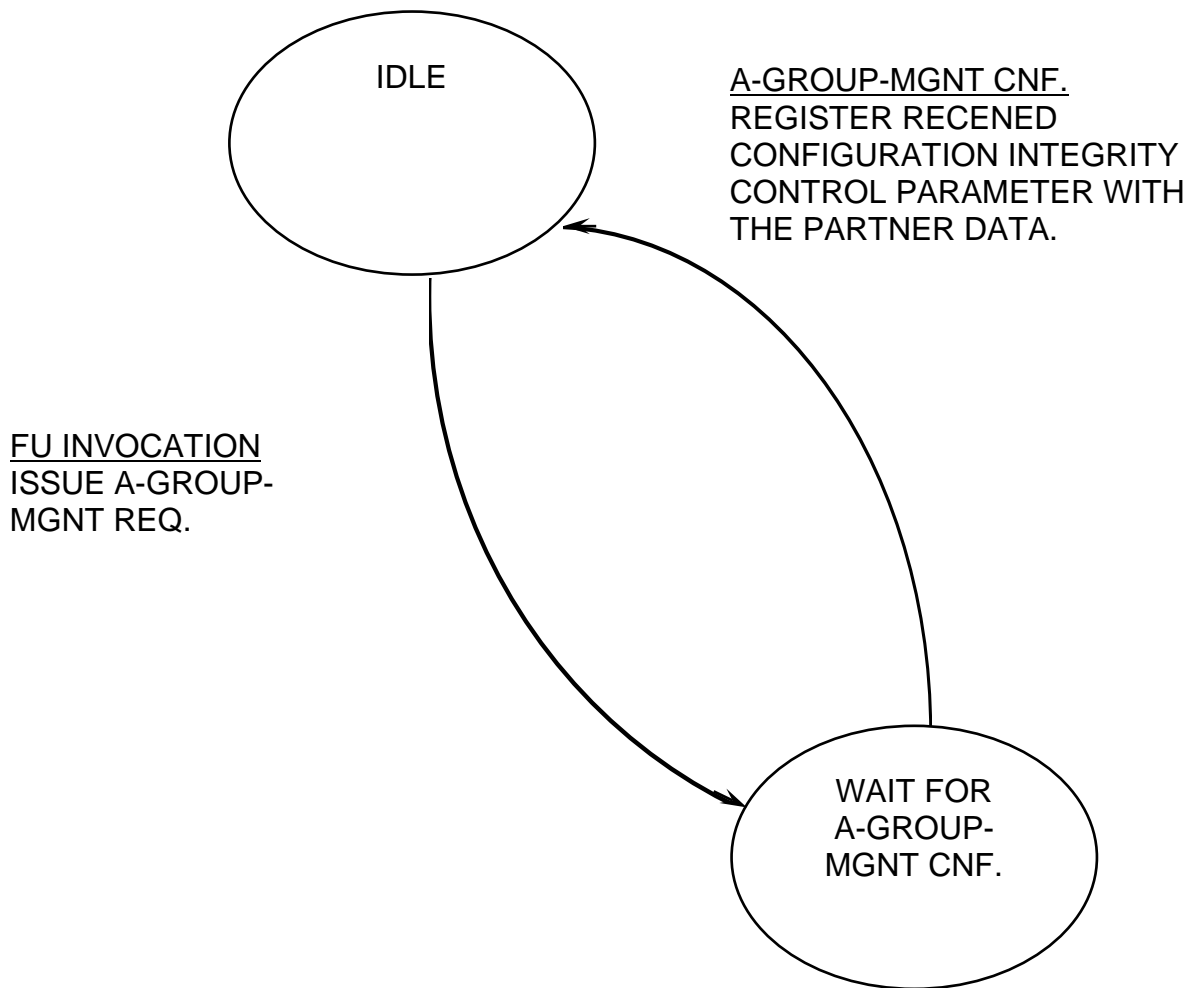


ATFU-01: Initiator part, error-free execution (no special error handling)

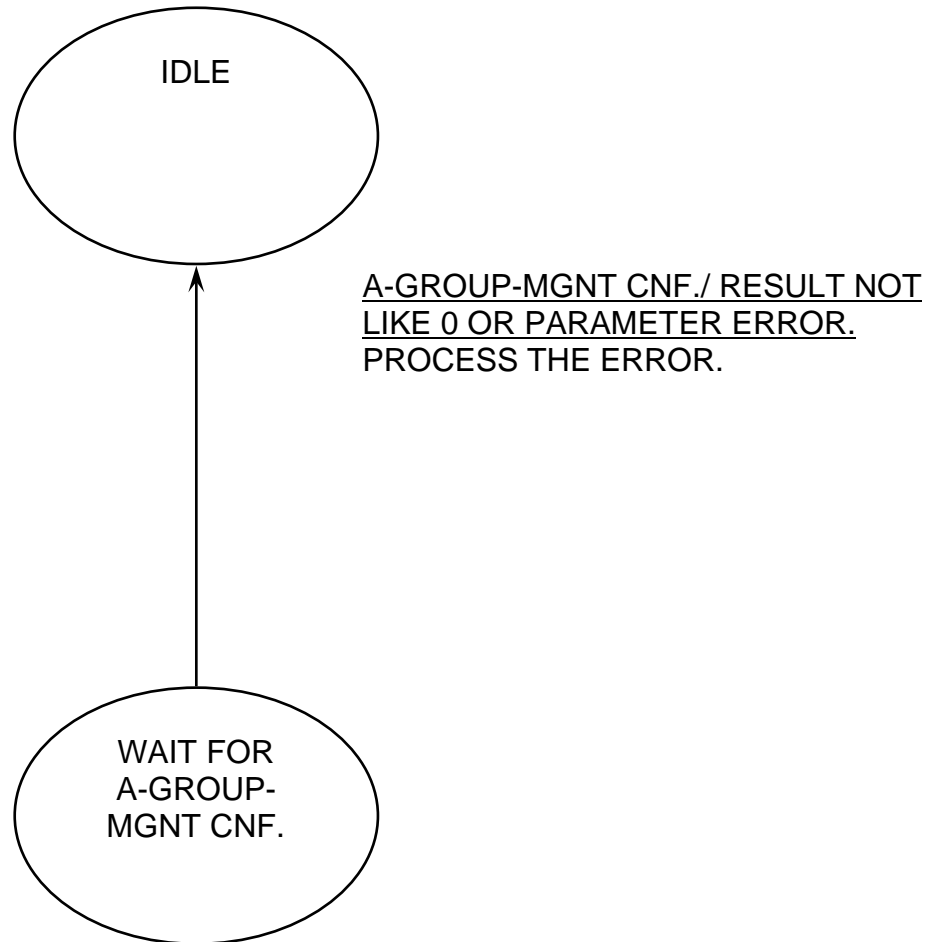


ATFU-02: Responder part, error-free execution (no special error handling)

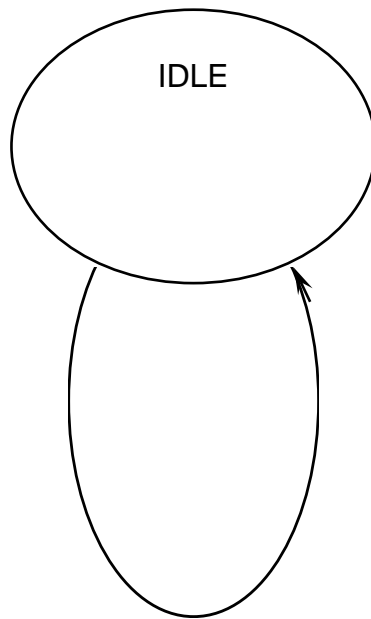
4. Group Management FU – GMFU



GMFU-01: Initiator part, error-free operation



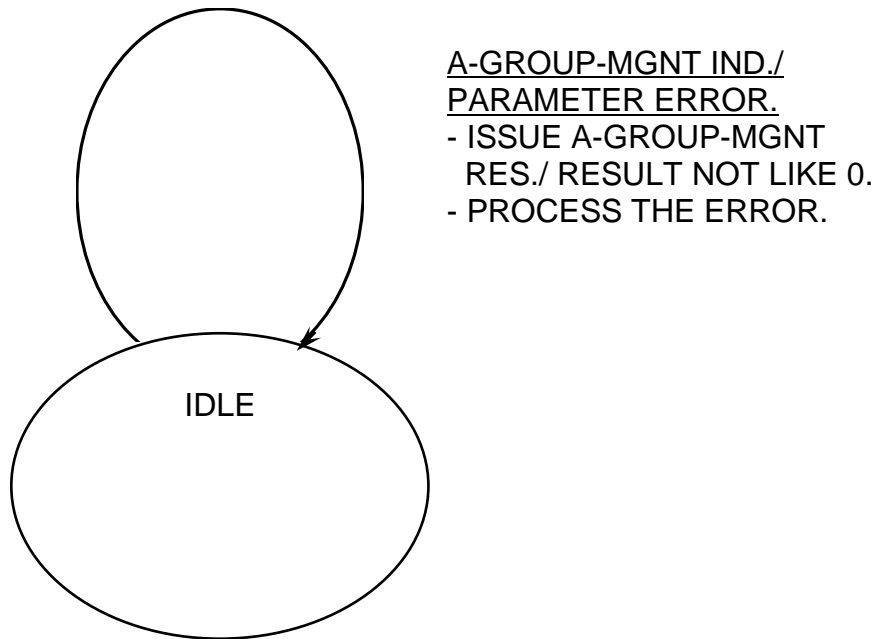
GMFU-02: Initiator part, error handling



A-GROUP-MGNT IND.

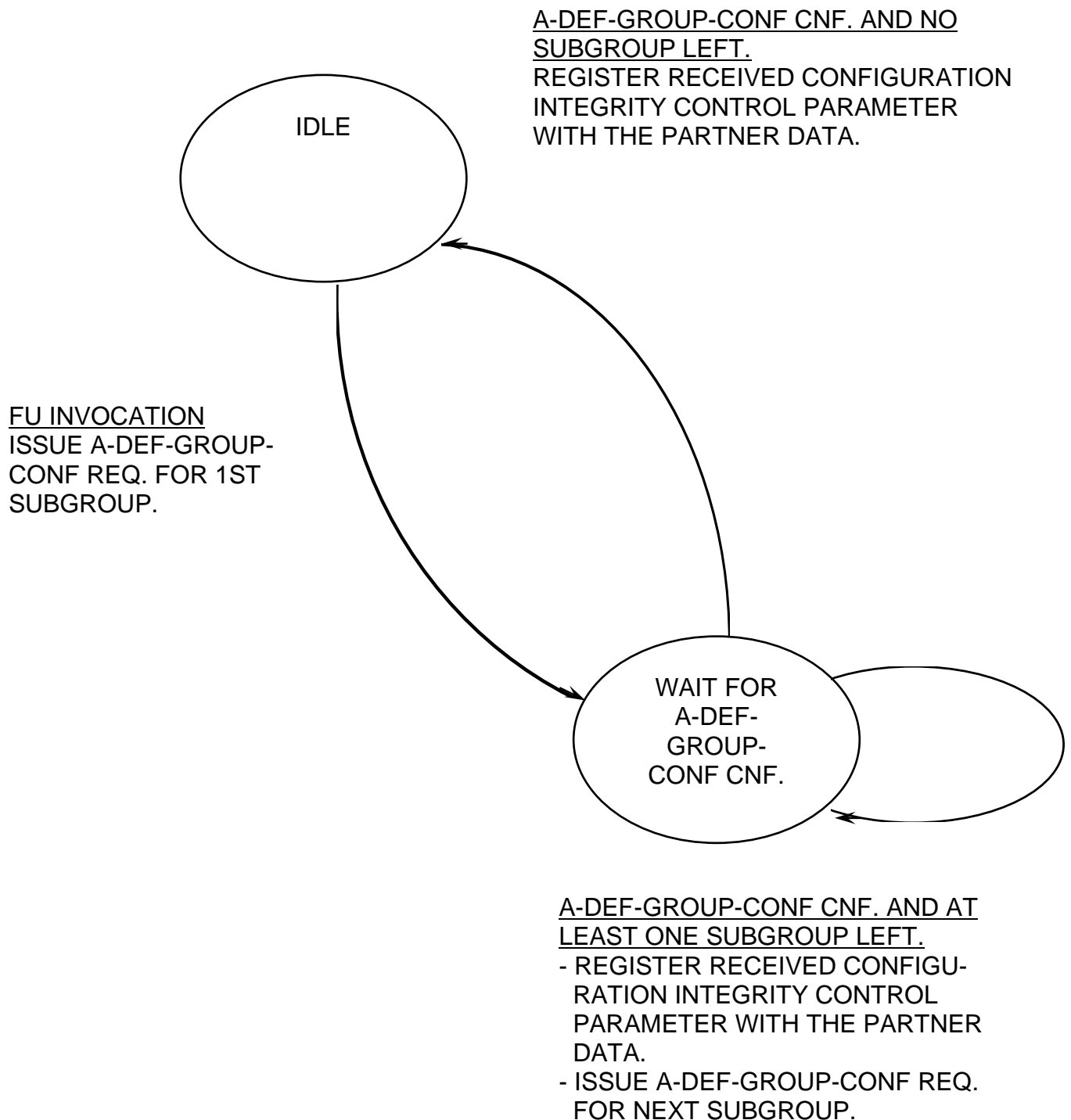
- UPDATE GROUP DESCRIPTOR ATTRIBUTES
- UPDATE CONFIGURATION INTEGRITY CONTROL
PARAMETER FOR PARTNER.
- ISSUE A-GROUP-MGNT RES.

GMFU-03: Responder part, error-free operation

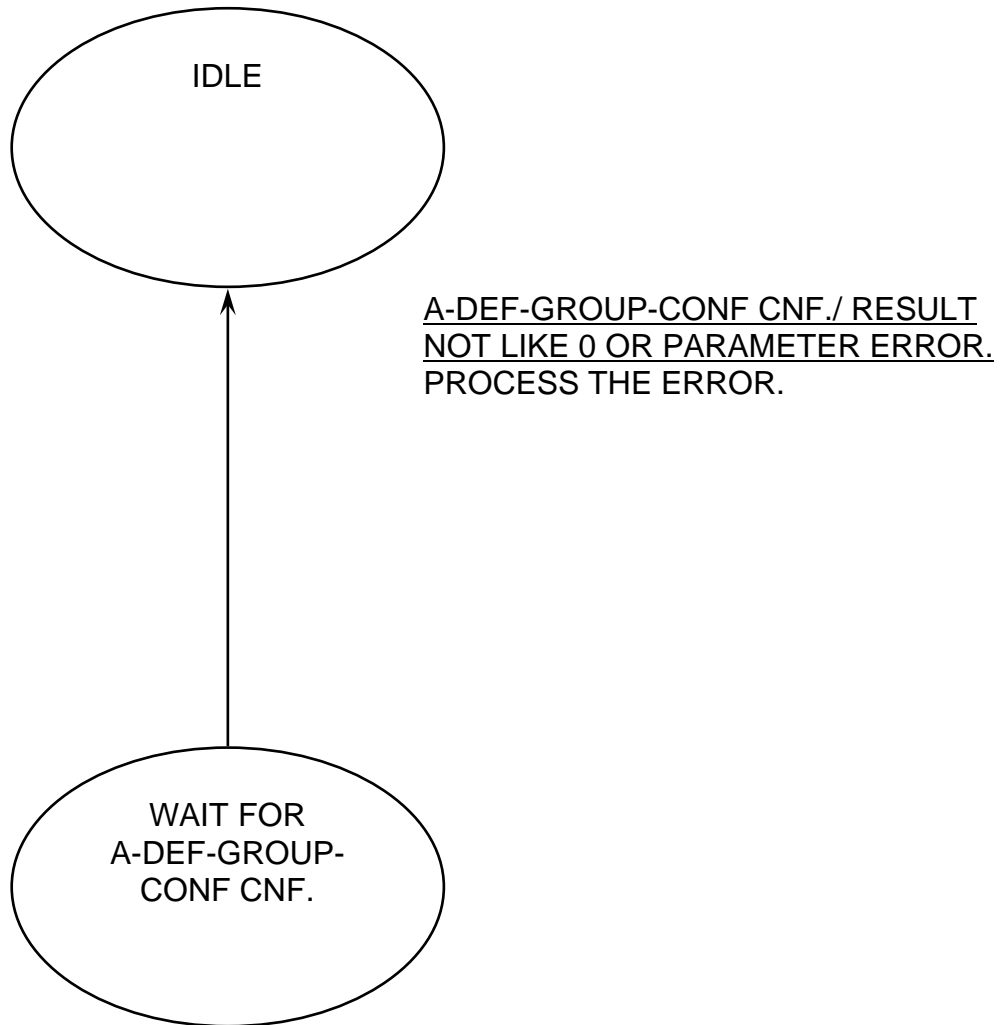


GMFU-04: Responder part, error handling

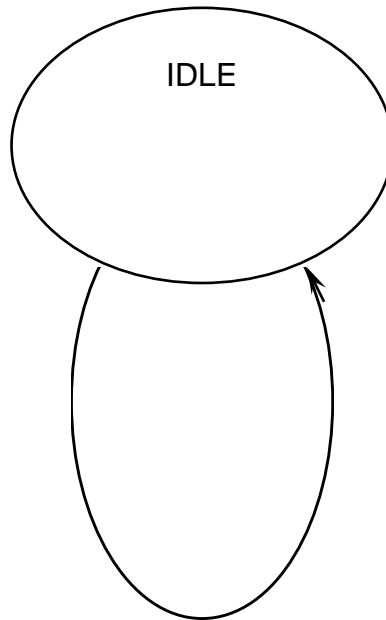
5. Group Definition FU – GDFU



GDFU-01: Initiator part, error-free operation



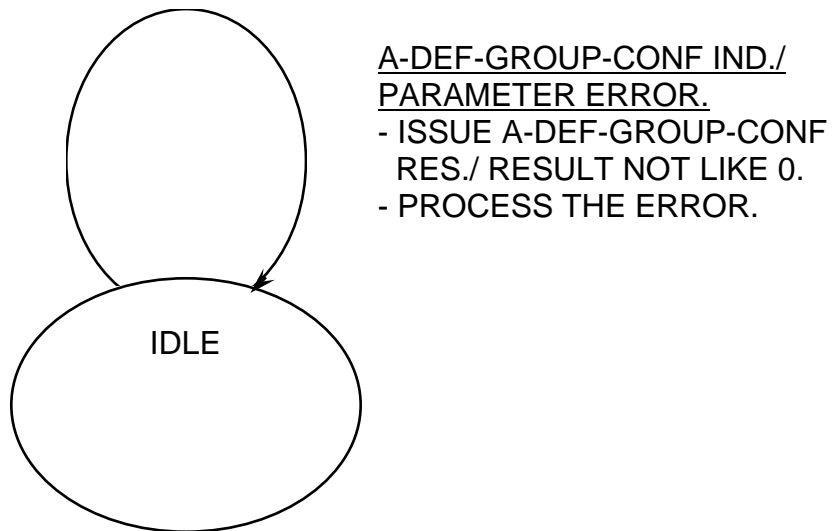
GDFU-02: Initiator part, error handling



A-DEF-GROUP-CONF IND.

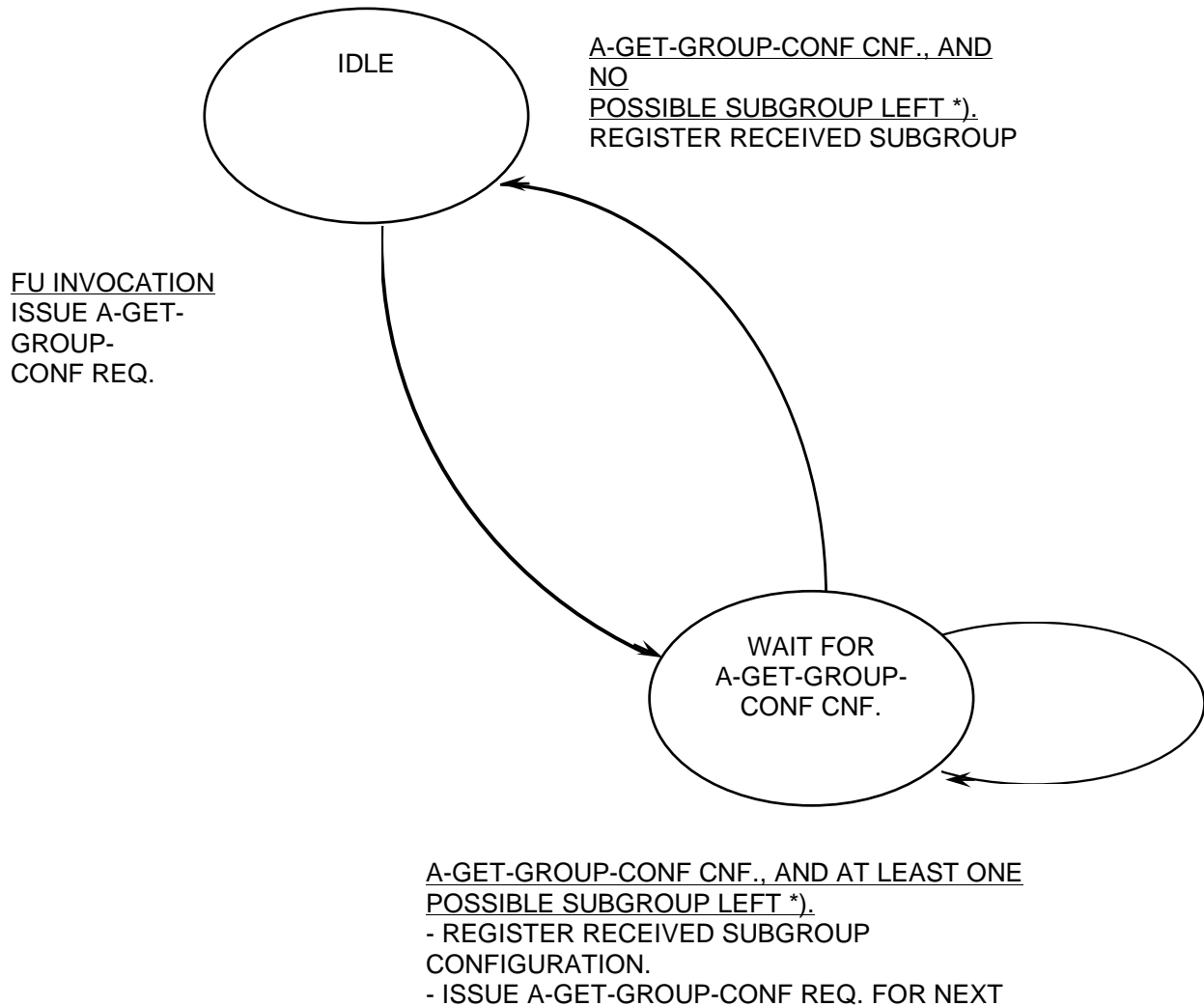
- UPDATE GROUP DEFINITION FOR RECEIVED SUBGROUP.
- UPDATE CONFIGURATION INTEGRITY CONTROL PARAMETER FOR PARTNER.
- ISSUE A-DEF-GROUP-CONF RES.

GDFU-03: Responder part, error-free operation



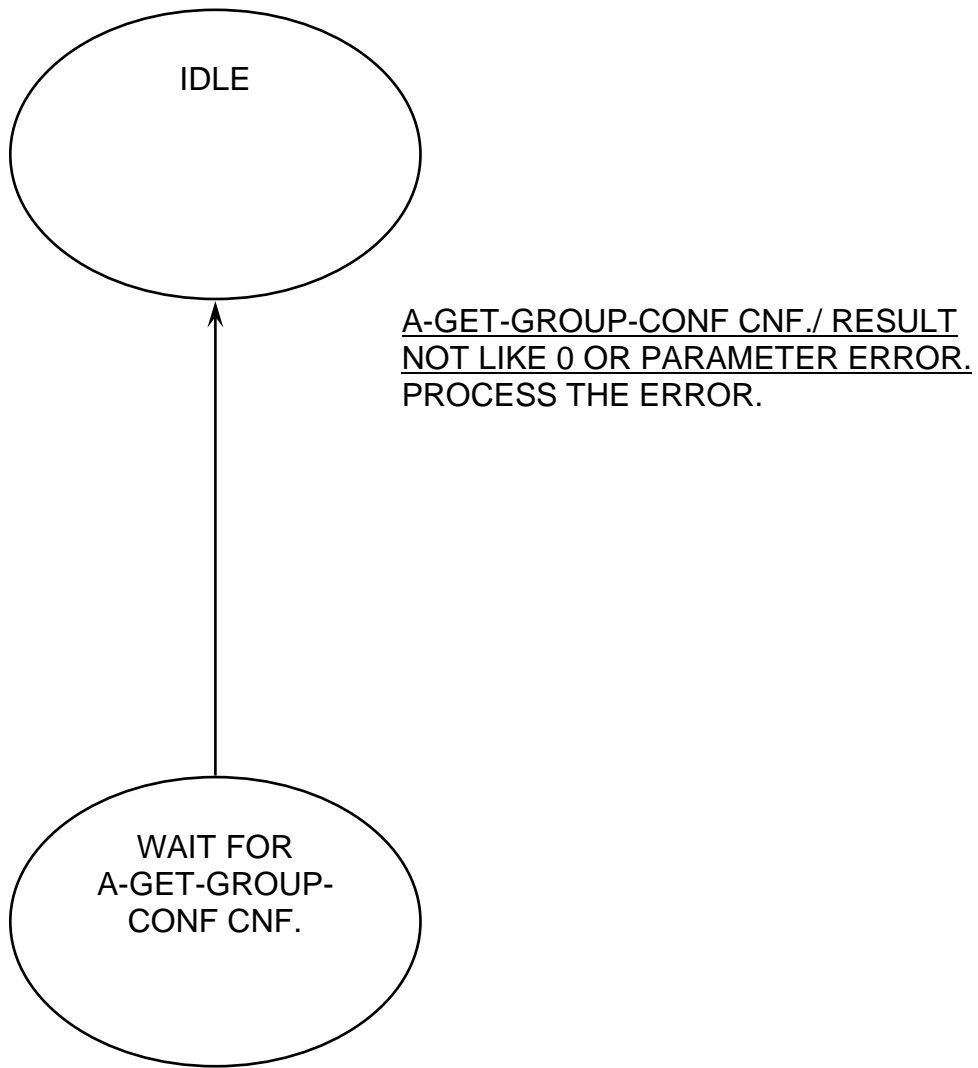
GDFU-04: Responder part, error handling

6. Group Readout FU – GRFU

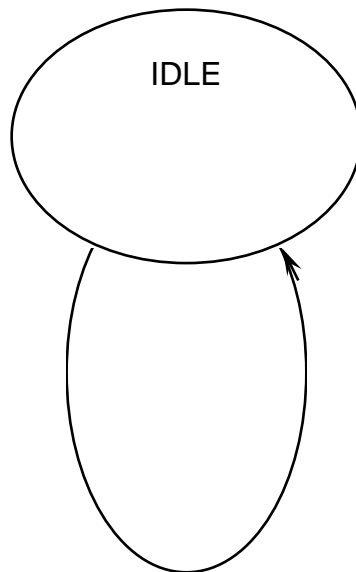


*) According to parameter GSIZE from 1st. A-GET-GROUP-CONF CNF. and when parameter OBJID from most recent A-GET-GROUP-CONF CNF. Done when GSIZE objects done,

GRFU-01: Initiator part, error-free operation

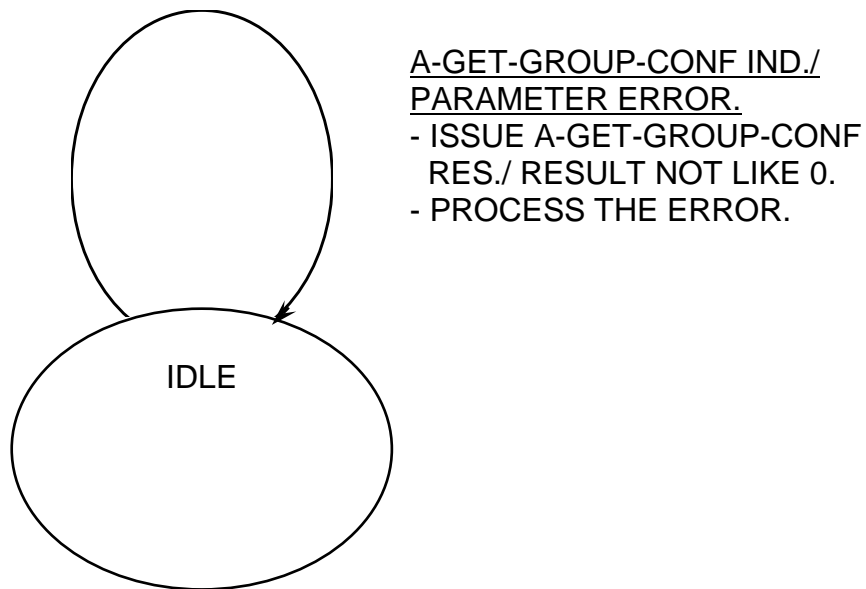


GRFU-02: Initiator part, error handling



A-GET-GROUP-CONF IND.
ISSUE A-GET-GROUP-CONF RES.
WITH THE REQUESTED DATA.

GRFU-03: Responder part, error-free operation



GRFU-04: Responder part, error handling

checksum may come out of the encipherment process. An alternative is to use a hashing function to generate the checksum before encipherment is done for the whole data parameter (including the checksum).

For field authentication and encipherment it is possible to ensure the integrity of the sequence of data fields by using cryptographic chaining. The CBC (Cipher Block Chaining) option of DES offers this.

9.6 Definition of the security information field

The *security information field* is used to select the security option and to convey the authentication information to the peer entity. The security information field is a part of the *user data field* in the *A-Connect* primitives. The maximum size of the security information field is 66 octets and is divided into three sub-fields:

- Length of security information field
- Security options field
- Authentication information field

The security information field may be absent. In this case security class 0 is chosen. If the length of the security information field is zero, security class 0 is chosen, as well.

If the authentication information field is absent (i.e. the security information field is 2 octets), the security options field must indicate security class 0 when sent by the INITIATOR UE. If not, the RESPONDER UE will reject the call with Result = *authentication-failure*.