IEC 61158-5-7

Edition 1.0 2007-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 5-7: Application Layer Service definition – Type 7 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-7: Définition des services des couches d'application – Éléments de Type 7

IEC 61158-5-7:2007-12(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

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

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

IEC Catalogue - webstore.iec.ch/catalogue
The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 14 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
More than 55 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de l'IEC
La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC
Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Catalogue IEC - webstore.iec.ch/catalogue
Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub
La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,…). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished
Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org
Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 14 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary
Plus de 55 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

IEC 61158-5-7

Edition 1.0 2007-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Industrial communication networks – Fieldbus specifications – Part 5-7: Application Layer Service definition – Type 7 elements

Réseaux de communication industriels – Spécifications des bus de terrain – Partie 5-7: Définition des services des couches d'application – Éléments de Type 7

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XH

ICS 25.040.40; 35.100.70

ISBN 978-2-8322-2000-9

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-7: Application Layer Service definition – Type 7 elements

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

NOTE Use of some of the associated protocol types is restricted by their intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in Type combinations as specified explicitly in the IEC 61784 series. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

International Standard IEC 61158-5-7 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This first edition and its companion parts of the IEC 61158-5 subseries cancel and replace IEC 61158-5:2003. This edition of this part constitutes an editorial revision.

This edition of IEC 61158-5 includes the following significant changes from the previous edition:

a) deletion of the former Type 6 fieldbus for lack of market relevance;
b) addition of new types of fieldbuses;
c) partition of part 5 of the third edition into multiple parts numbered -5-2, -5-3, ...

This bilingual version (2014-12) corresponds to the monolingual English version, published in 2007-12.

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

FDIS	Report on voting
65C/475/FDIS	65C/486/RVD

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

The French version of this standard has not been voted upon.

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

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

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

NOTE The revision of this standard will be synchronized with the other parts of the IEC 61158 series.

The list of all the parts of the IEC 61158 series, under the general title *Industrial communication networks – Fieldbus specifications*, can be found on the IEC web site.

INTRODUCTION

This part of IEC 61158 is one of a series produced to facilitate the interconnection of automation system components. It is related to other standards in the set as defined by the “three-layer” fieldbus reference model described in IEC/TR 61158-1.

The application service is provided by the application protocol making use of the services available from the data-link or other immediately lower layer. This standard defines the application service characteristics that fieldbus applications and/or system management may exploit.

Throughout the set of fieldbus standards, the term “service” refers to the abstract capability provided by one layer of the OSI Basic Reference Model to the layer immediately above. Thus, the application layer service defined in this standard is a conceptual architectural service, independent of administrative and implementation divisions.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-7: Application Layer Service definition – Type 7 elements

1 Scope

1.1 Overview

The fieldbus Application Layer (FAL) provides user programs with a means to access the fieldbus communication environment. In this respect, the FAL can be viewed as a "window between corresponding application programs."

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment. The term "time-critical" is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard defines in an abstract way the externally visible service provided by the Type 7 fieldbus Application Layer in terms of

a) an abstract model for defining application resources (objects) capable of being manipulated by users via the use of the FAL service,

b) the primitive actions and events of the service;

c) the parameters associated with each primitive action and event, and the form which they take; and

d) the interrelationship between these actions and events, and their valid sequences.

The purpose of this standard is to define the services provided to

1) the FAL user at the boundary between the user and the Application Layer of the Fieldbus Reference Model, and

2) Systems Management at the boundary between the Application Layer and Systems Management of the Fieldbus Reference Model.

This standard specifies the structure and services of the IEC fieldbus Application Layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI Application Layer Structure (ISO/IEC 9545).

FAL services and protocols are provided by FAL application-entities (AE) contained within the application processes. The FAL AE is composed of a set of object-oriented Application Service Elements (ASEs) and a Layer Management Entity (LME) that manages the AE. The ASEs provide communication services that operate on a set of related application process object (APO) classes. One of the FAL ASEs is a management ASE that provides a common set of services for the management of the instances of FAL classes.

Although these services specify, from the perspective of applications, how request and responses are issued and delivered, they do not include a specification of what the requesting and responding applications are to do with them. That is, the behavioral aspects of the applications are not specified; only a definition of what requests and responses they can send/receive is specified. This permits greater flexibility to the FAL users in standardizing such object behavior. In addition to these services, some supporting services are also defined in this standard to provide access to the FAL to control certain aspects of its operation.

1.2 Specifications

The principal objective of this standard is to specify the characteristics of conceptual application layer services suitable for time-critical communications, and thus supplement the OSI Basic Reference Model in guiding the development of application layer protocols for time-critical communications.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of services standardized as the various types of IEC 61158.

This specification may be used as the basis for formal application programming interfaces. Nevertheless, it is not a formal programming interface, and any such interface will need to address implementation issues not covered by this specification, including

a) the sizes and octet ordering of various multi-octet service parameters, and

b) the correlation of paired request and confirm, or indication and response, primitives.

1.3 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems.

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through implementation of conforming application layer protocols that fulfill the Type 7 application layer services as defined in this standard.

2 Normative references

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

IEC 60559, *Binary Floating-point Arithmetic for Microprocessor Systems*

IEC/TR 61158-1 (Ed.2.0), *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-3-7, *Industrial communication networks – Fieldbus specifications – Part 3-7: Data-link layer service definition – Type 7 elements*

IEC 61158-4-7, *Industrial communication networks – Fieldbus specifications – Part 4-7: Data-link layer protocol specification – Type 7 elements*

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model — Part 1: The Basic Model*

ISO/IEC 7498-3, *Information technology – Open Systems Interconnection – Basic Reference Model — Part 3: Naming and addressing*

ISO/IEC 8822, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824, *Information Technology – Abstract Syntax notation One (ASN-1): Specification of basic notation*

ISO/IEC 9545, *Information technology – Open Systems Interconnection – Application Layer structure*

ISO/IEC 10731, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

3 Terms, definitions, symbols, abbreviations and conventions

For the purposes of this document, the following terms as defined in these publications apply.

3.1 ISO/IEC 7498-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 7498-1 apply:

a) application entity
b) application process
c) application protocol data unit
d) application service element
e) application entity invocation
f) application process invocation
g) application transaction
h) real open system
i) transfer syntax

3.2 ISO/IEC 8822 terms

a) abstract syntax
b) presentation context

3.3 ISO/IEC 9545 terms

a) application-association
b) application-context
c) application context name
d) application-entity-invocation
e) application-entity-type
f) application-process-invocation
g) application-process-type
h) application-service-element
i) application control service element

3.4 ISO/IEC 8824 terms

a) object identifier
b) type

3.5 Fieldbus data-link layer terms

The following terms as defined in IEC 61158-3-7 and IEC 61158-4-7 apply.

a) acknowledgement response DLPDU
b) basic cycle
c) basic transaction

d) bus-arbitrator (BA)
e) control field
f) destination address
e) DL-segment, local link
g) DLCEP-identifier
h) DLCEP-iedentifier DLPDU
i) end of message transaction indication DLPDU
j) identified variable (or simply « variable »)
k) invalid DLCEP-identifier
l) macrocycle
m) message DLPDU identifier
n) message response DLPDU
o) periodic scanning of variables
p) published identified varaible
q) request response DLPDU
r) source address
s) subscribed identified variable
t) triggered message scanning
u) triggered periodic scanning of messages
w) triggered periodic scanning of variables
x) triggered scanning of variables
y) turnaround time
z) variable response DLPDU

The following symbols and abbreviations as defined in IEC 61158-3-7 and IEC 61158-4-7 apply.

a) BA
b) B_Dat_cons
c) B_Dat_Prod
d) B_Req1/2
e) ID_DAT
f) ID_MSG
g) ID_RQ1/2
h) PRT
i) Q_IDMSG
j) Q_IDRQ1/2
k) Q_Msg_Aper
l) Q_Req1/2
m) Q_RPRQ
n) RQ_Inhibit
o) RP_ACK
p) RP_DAT
q) RP_DAT_MSG

r) RP_DAT_RQ1/2
s) RP_DAT_RQ1/2_MSG
t) RP_MSG_ACK
u) RP_MSG_NOACK
v) RP_NAK
w) RP_END
x) STT
y) TI

3.6 Fieldbus application layer specific definitions

For the purposes of this standard, the following terms and definitions apply.

3.6.1
access protection
limitation of the usage of an application object to one client

3.6.2
active connection control object
instance of a certain FAL class that abstracts the interconnection facility (as Consumer and Provider) of an automation device

3.6.3
address assignment table
mapping of the client's internal I/O-Data object storage to the decentralised input and output data objects

3.6.4
allocate
take a resource from a common area and assign that resource for the exclusive use of a specific entity

3.6.5
application
function or data structure for which data is consumed or produced

3.6.6
application layer interoperability
capability of application entities to perform coordinated and cooperative operations using the services of the FAL

3.6.7
application objects
multiple object classes that manage and provide a run time exchange of messages across the network and within the network device

3.6.8
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.6.9
application process identifier
distinguishes multiple application processes used in a device

3.6.10
application process object
component of an application process that is identifiable and accessible through an FAL application relationship

NOTE Application process object definitions are composed of a set of values for the attributes of their class (see the definition for Application Process Object Class Definition). Application process object definitions may be accessed remotely using the services of the FAL Object Management ASE. FAL Object Management services can be used to load or update object definitions, to read object definitions, and to dynamically create and delete application objects and their corresponding definitions.

3.6.11
application process object class
a class of application process objects defined in terms of the set of their network-accessible attributes and services

3.6.12
application relationship
cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation

NOTE This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities.

3.6.13
application relationship application service element
application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.6.14
application relationship endpoint
context and behavior of an application relationship as seen and maintained by one of the application processes involved in the application relationship

NOTE Each application process involved in the application relationship maintains its own application relationship endpoint.

3.6.15
attribute
description of an externally visible characteristic or feature of an object

NOTE The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behaviour of an object. Attributes are divided into class attributes and instance attributes.

3.6.16
behaviour
indication of how an object responds to particular eventss

3.6.17
bit-no
designates the number of a bit in a bitstring or an octet

3.6.18
channel
single physical or logical link of an input or output application object of a server to the process

3.6.19
class
a set of objects, all of which represent the same kind of system component

NOTE A class is a generalisation of an object; a template for defining variables and methods. All objects in a class are identical in form and behaviour, but usually contain different data in their attributes.

3.6.20
class attributes
attribute that is shared by all objects within the same class

3.6.21
class code
unique identifier assigned to each object class

3.6.22
class specific service
service defined by a particular object class to perform a required function which is not performed by a common service

NOTE A class specific object is unique to the object class which defines it.

3.6.23
client
a) object which uses the services of another (server) object to perform a task

b) initiator of a message to which a server reacts

3.6.24
communication objects
components that manage and provide a run time exchange of messages across the network

EXAMPLES: Connection Manager object, Unconnected Message Manager (UCMM) object, and Message Router object

3.6.25
configuration identifier
representation of a portion of I/O Data of a single input- and/or output-module of a server

3.6.26
connection
logical binding between application objects that may be within the same or different devices

NOTE 1 Connections may be either point-to-point or multipoint.

NOTE 2 The logical link between sink and source of attributes and services at different custom interfaces of RT-Auto ASEs is referred to as interconnection. There is a distinction between data and event interconnections. The logical link and the data flow between sink and source of automation data items is referred to as data interconnection. The logical link and the data flow between sink (method) and source (event) of operational services is referred to as event interconnection.

3.6.27
connection channel
description of a connection between a sink and a source of data items

3.6.28
connection ID (CID)
identifier assigned to a transmission that is associated with a particular connection between producers and consumers, providing a name for a specific piece of application information

3.6.29
connection path
octet stream that defines the application object to which a connection instance applies

3.6.30
connection point
buffer which is represented as a subinstance of an Assembly object

3.6.31
consume
act of receiving data from a producer

3.6.32
consumer
node or sink that is receiving data from a producer

3.6.33
consuming application
application that consumes data

3.6.34
consumerID
unambiguous identifier within the scope of the ACCO assigned by the consumer to recognize the internal data of a configured interconnection sink

3.6.35
control commands
action invocations transferred from client to server to clear outputs, freeze inputs and/or synchronise outputs

3.6.36
conveyance path
unidirectional flow of APDUs across an application relationship

3.6.37
cyclic
repetitive in a regular manner

3.6.38
data consistency
means for coherent transmission and access of the input- or output-data object between and within client and server

3.6.39
dedicated AR
AR used directly by the FAL User

NOTE On Dedicated ARs, only the FAL Header and the user data are transferred.

3.6.40
device
physical hardware connected to the link

NOTE A device may contain more than one node.

3.6.41
device profile
collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.6.42
end node
producing or consuming node

3.6.43
endpoint
one of the communicating entities involved in a connection

3.6.44
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.6.45
error class
general grouping for related error definitions and corresponding error codes

3.6.46
error code
identification of a specific type of error within an error class

3.6.47
event
instance of a change of conditions

3.6.48
FIFO variable
a Variable Object class, composed of a set of homogeneously typed elements, where the first written element is the first element that can be read

NOTE On the fieldbus only one, complete element can be transferred as a result of one service invocation.

3.6.49
frame
denigrated synonym for DLPDU

3.6.50
group
a) <general> a general term for a collection of objects.

b) <addressing> when describing an address, an address that identifies more than one entity

3.6.51
interface
a) shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate

b) collection of FAL class attributes and services that represents a specific view on the FAL class

3.6.52
interface definition language
syntax and semantics of describing service parameters in a formal way

NOTE This description is the input for the ORPC model, especially for the ORPC wire protocol.

3.6.53
interface pointer
key attribute that unambiguously addresses an object interface instance

3.6.54
invocation
act of using a service or other resource of an application process

NOTE Each invocation represents a separate thread of control that may be described by its context. Once the service completes, or use of the resource is released, the invocation ceases to exist. For service invocations, a service that has been initiated but not yet completed is referred to as an outstanding service invocation. Also for service invocations, an Invoke ID may be used to unambiguously identify the service invocation and differentiate it from other outstanding service invocations.

3.6.55
index
address of an object within an application process

3.6.56
instance
actual physical occurrence of an object within a class that identifies one of many objects within the same object class

EXAMPLE California is an instance of the object class state.

NOTE The terms object, instance, and object instance are used to refer to a specific instance.

3.6.57
instance attributes
attribute that is unique to an object instance and not shared by the object class

3.6.58
instantiated
object that has been created in a device

3.6.59
logical device
certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

3.6.60
manufacturer ID
identification of each product manufacturer by a unique number

3.6.61
management information
network-accessible information that supports managing the operation of the fieldbus system, including the application layer

NOTE Managing includes functions such as controlling, monitoring, and diagnosing.

3.6.62
member
piece of an attribute that is structured as an element of an array

3.6.63
method
synonym for an operational service which is provided by the server ASE and invoked by a client

3.6.64
module
hardware or logical component of a physical device

3.6.65
multipoint connection
connection from one node to many

NOTE Multipoint connections allow messages from a single producer to be received by many consumer nodes.

3.6.66
network
a set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.6.67
object
abstract representation of a particular component within a device, usually a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behaviour

3.6.68
object remote procedure call
model for object oriented or component based remote method invocation

3.6.69
object specific service
service unique to the object class which defines it

3.6.70
originator
client responsible for establishing a connection path to the target

3.6.71
peer
role of an AR endpoint in which it is capable of acting as both client and server

3.6.72
physical device
an automation or other network device

3.6.73
point-to-point connection
connection that exists between exactly two application objects

3.6.74
pre-defined AR endpoint
AR endpoint that is defined locally within a device without use of the create service

NOTE Pre-defined ARs that are not pre-established are established before being used.

3.6.75
pre-established AR endpoint
AR endpoint that is placed in an established state during configuration of the AEs that control its endpoints

3.6.76
process data
object(s) which are already pre-processed and transferred acyclically for the purpose of information or further processing

3.6.77
produce
act of sending data to be received by a consumer

3.6.78
producer
node that is responsible for sending data

3.6.79
provider
source of a data connection

3.6.80
publisher
role of an AR endpoint that transmits APDUs onto the fieldbus for consumption by one or more subscribers

NOTE A publisher may not be aware of the identity or the number of subscribers and it may publish its APDUs using a dedicated AR.

3.6.81
resource
processing or information capability of a subsystem

3.6.82
server
a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request

b) object which provides services to another (client) object

3.6.83
service
operation or function than an object and/or object class performs upon request from another object and/or object class

3.6.84
subscriber
role of an AREP in which it receives APDUs produced by a publisher

3.7 Abbreviations and symbols

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
APO	Application Object
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
AR	Application Relationship
AREP	Application Relationship End Point
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
CID	Connection ID
CIM	Computer Integrated Manufacturing
CIP	Control and Information Protocol
Cnf	Confirmation
COR	Connection originator
CR	Communication Relationship
CREP	Communication Relationship End Point
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLCEP	Data Link Connection End Point

DLL	Data Link Layer
DLM	Data Link-management
DLSAP	Data Link Service Access Point
DLSDU	DL-service-data-unit
DNS	Domain Name Service
DP	Decentralised Peripherals
FAL	Fieldbus Application Layer
FIFO	First In First Out
HMI	Human-Machine Interface
ID	Identifier
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
Ind	Indication
IP	Internet Protocol
ISO	International Organization for Standardization
LDev	Logical Device
LME	Layer Management Entity
ORPC	Object Remote Procedure Call
OSI	Open Systems Interconnect
PDev	Physical Device
PDU	Protocol Data Unit
PL	Physical Layer
QoS	Quality of Service
Req	Request
Rsp	Response
RT	Runtime
SAP	Service Access Point
SCL	Security Level
SDU	Service Data Unit
SMIB	System Management Information Base
SMK	System Management Kernel
STD	State transition diagram, used to describe object behaviour
S-VFD	Simple Virtual Field Device
VAO	Variable Object

3.8 Conventions

3.8.1 Overview

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of two parts, its class specification, and its service specification.

The class specification defines the attributes of the class. The attributes are accessible from instances of the class using the Object Management ASE services specified in Clause 5 of this standard. The service specification defines the services that are provided by the ASE.

3.8.2 Conventions for class definitions

Class definitions are described using templates. Each template consists of a list of attributes for the class. The general form of the template is shown below:

FAL ASE:			**ASE Name**
CLASS:	**Class Name**		
CLASS ID:			#
PARENT CLASS:			Parent Class Name
ATTRIBUTES:			
1	(o)	Key Attribute:	numeric identifier
2	(o)	Key Attribute:	name
3	(m)	Attribute:	attribute name(values)
4	(m)	Attribute:	attribute name(values)
4.1	(s)	Attribute:	attribute name(values)
4.2	(s)	Attribute:	attribute name(values)
4.3	(s)	Attribute:	attribute name(values)
5.	(c)	Constraint:	constraint expression
5.1	(m)	Attribute:	attribute name(values)
5.2	(o)	Attribute:	attribute name(values)
6	(m)	Attribute:	attribute name(values)
6.1	(s)	Attribute:	attribute name(values)
6.2	(s)	Attribute:	attribute name(values)
SERVICES:			
1	(o)	OpsService:	service name
2.	(c)	Constraint:	constraint expression
2.1	(o)	OpsService:	service name
3	(m)	MgtService:	service name

(1) The "FAL ASE:" entry is the name of the FAL ASE that provides the services for the class being specified.

(2) The "CLASS:" entry is the name of the class being specified. All objects defined using this template will be an instance of this class. The class may be specified by this standard, or by a user of this standard.

(3) The "CLASS ID:" entry is a number that identifies the class being specified. This number is unique within the FAL ASE that will provide the services for this class. When qualified by the identity of its FAL ASE, it unambiguously identifies the class within the scope of the FAL. The value "NULL" indicates that the class cannot be instantiated. Class IDs between 1 and 255 are reserved by this standard to identify standardized classes. They have been assigned to maintain compatibility with existing national standards. CLASS IDs between 256 and 2048 are allocated for identifying user defined classes.

(4) The "PARENT CLASS:" entry is the name of the parent class for the class being specified. All attributes defined for the parent class and inherited by it are inherited for the class being defined, and therefore do not have to be redefined in the template for this class.

NOTE The parent-class "TOP" indicates that the class being defined is an initial class definition. The parent class TOP is used as a starting point from which all other classes are defined. The use of TOP is reserved for classes defined by this standard.

(5) The "ATTRIBUTES" label indicate that the following entries are attributes defined for the class.

a) Each of the attribute entries contains a line number in column 1, a mandatory (m) / optional (o) / conditional (c) / selector (s) indicator in column 2, an attribute type label

in column 3, a name or a conditional expression in column 4, and optionally a list of enumerated values in column 5. In the column following the list of values, the default value for the attribute may be specified.

b) Objects are normally identified by a numeric identifier or by an object name, or by both. In the class templates, these key attributes are defined under the key attribute.

c) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting is used to specify

i) fields of a structured attribute (4.1, 4.2, 4.3),

ii) attributes conditional on a constraint statement (5). Attributes may be mandatory (5.1) or optional (5.2) if the constraint is true. Not all optional attributes require constraint statements as does the attribute defined in (5.2).

iii) the selection fields of a choice type attribute (6.1 and 6.2).

(6) The "SERVICES" label indicates that the following entries are services defined for the class.

a) An (m) in column 2 indicates that the service is mandatory for the class, while an (o) indicates that it is optional. A (c) in this column indicates that the service is conditional. When all services defined for a class are defined as optional, at least one has to be selected when an instance of the class is defined.

b) The label "OpsService" designates an operational service (1).

c) The label "MgtService" designates an management service (2).

d) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting within the list of services is used to specify services conditional on a constraint statement.

3.8.3 Conventions for service definitions

3.8.3.1 General

This standard uses the descriptive conventions given in ISO/IEC 10731.

The service model, service primitives, and time-sequence diagrams used are entirely abstract descriptions; they do not represent a specification for implementation.

3.8.3.2 Service parameters

Service primitives are used to represent service user/service provider interactions (ISO/IEC 10731). They convey parameters which indicate information available in the user/provider interaction.

NOTE 1 See the note under 3.8.3.3 relative to the non-inclusion of service parameters that are appropriate to a protocol specification or programming interface specification or implementation specification, but not to an abstract service definition.

This standard uses a tabular format to describe the component parameters of the service primitives. The parameters that apply to each group of service primitives are set out in tables throughout the remainder of this standard. Each table consists of up to six columns: a column for the name of the service parameter, and a column each for those primitives and parameter-transfer directions used by the service. The possible six columns are:

1) the parameter name;

2) the request primitive's input parameters;

3) the request primitive's output parameters;

NOTE 2 This is a seldom-used capability. Unless otherwise specified, request primitive parameters are input parameters.

4) the indication primitive's output parameters;

5) the response primitive's input parameters; and

6) the confirm primitive's output parameters.

NOTE 3 The request, indication, response and confirm primitives are also known as requestor.submit, acceptor.deliver, acceptor.submit, and requestor.deliver primitives, respectively (see ISO/IEC 10731).

One parameter (or component of it) is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the column:

M parameter is mandatory for the primitive

U parameter is a User option, and may or may not be provided depending on dynamic usage of the service user. When not provided, a default value for the parameter is assumed.

C parameter is conditional upon other parameters or upon the environment of the service user.

— (blank) parameter is never present.

S parameter is a selected item.

Some entries are further qualified by items in brackets. These may be

a) a parameter-specific constraint:
"(=)" indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.

b) an indication that some note applies to the entry:
"(n)" indicates that the following note "n" contains additional information pertaining to the parameter and its use.

3.8.3.3 Service procedures

The procedures are defined in terms of

- the interactions between application entities through the exchange of fieldbus Application Protocol Data Units, and
- the interactions between an application layer service provider and an application layer service user in the same system through the invocation of application layer service primitives.

These procedures are applicable to instances of communication between systems which support time-constrained communications services within the fieldbus Application Layer.

NOTE The IEC 61158-5 series of standards define sets of abstract services. They are neither protocol specifications nor implementation specifications nor concrete programming interface specifications. Therefore there are restrictions on the extent to which service procedures can be mandated in the parts of IEC 61158-5. Protocol aspects that can vary among different protocol specifications or different implementations that instantiate the same abstract services are unsuitable for inclusion in these service definitions, except at the level of abstraction that is necessarily common to all such expressions.

For example, the means by which service providers pair request and reply PDUs is appropriate for specification in an IEC 61158-6 protocol specification standard but not in an IEC 61158-5 abstract service definition standard. Similarly, local implementation methods by which a service provider or service user pairs request and confirm(ation) primitives, or indication and response primitives, is appropriate for an implementation specification or for a programming interface specification, but not for an abstract service standard or for a protocol standard, except at a level of abstraction that is necessarily common to all embodiments of the specifying standard. In all cases, the abstract definition is not permitted to over-specify the more concrete instantiating realization.

Further information on the conceptual service procedures of an implementation of a protocol that realizes the services of one of the IEC 61158-5 abstract service definitions can be found in IEC/TR 61158-1, 9.6.

4 Concepts

The common concepts and templates used to describe the application layer service in this fieldbus are detailed in IEC/TR 61158-1, Clause 9.

5 Data type ASE

5.1 Overview

An overview of the data type ASE and the relationships between data types is provided in IEC/TR 61158-1, 10.1

5.2 Formal definition of data type objects

The template used to describe the data type class is detailed in IEC/TR 61158-1, 10.2. This includes the specific ASE structure and the definition of its attributes.

5.3 FAL defined data types

5.3.1 Fixed length types

Primitive types that are selected and their characteristics are mostly specified in the ISO 8824 standard.

The Used Data types and the discrepancies between the 8824 and this standard are listed hereafter.

5.3.1.1 BOOLEAN

Conform to the 8824 Standard.

5.3.1.2 FLOATING POINT

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	Not used
2	Data type Name	=	FloatingPoint
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	Boolean

This primitive type defines value representation for positive and negative real numbers, including, zero, negative infinite and positive infinite.

The values at the limits (0, +• -•), as well as the various representation formats are defined in IEC 60559.

For this type, the attribut Octet length, of boolean type, indicates whether the representation format is simple precision “4 octets” (FALSE) or double precision “8 octets” (TRUE).

5.3.1.3 BCD

Conform to the ISO/IEC 8824.

5.3.1.4 INTEGER

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	Not used
2	Data type Name	=	Integer
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	n

The definition of this primitive type is the same as the integer type definition in ISO 8824 standard. For this type, the Octete lentgh attribute defines the number of bits that are necessary to have a twos complement representation of all possible values.

Ex : Octet lentgh = 1 if -128 ≤ n ≤ 127 ; 2 if -32768 ≤ n ≤ 32767; .....

5.3.1.5 UNSIGNED

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	Not used
2	Data type Name	=	Unsigned
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	n

The definition of this primitive type is the same as the integer type definition in ISO 8824 with the exclusion of negative numbers. For this type, the Octet Length attribute defines the number of bits that are necessary to have a twos complement representation of all possible values.

Ex : 1 : for unsigned8 ; 2 : for unsigned16 ; 4 : for unsigend32 ; 8 for unsigned64

5.3.1.6 GENERALISED TIME

Conform to the ISO/IEC 8824.

5.3.1.7 BINARY TIME

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	Not used
2	Data type Name	=	BinaryTime
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	n

This primitive type is defined as the unsigned type of the present standard, the value of which corresponds to a number of elementary times.

For this type, each value of the *Octet Lentgh* attribute defines the value of an elementary time as well as a number of bits used to represent any possible binary time value, as show in the Table 1.

Table 1 – Binary time coding

Size	10 ms	0,1 ms	1 ms	10 ms
16 Bits	0	1	2	3
32 Bits	4	5	6	7
48 Bits	8	9	—	—

5.3.2 String types

5.3.2.1 BIT STRING

Conform to the ISO/IEC 8824.

5.3.2.2 OCTET STRING

Conform to the ISO/IEC 8824.

5.3.2.3 VISIBLE STRING

Conform to the ISO/IEC 8824.

5.3.3 Array types

There are defined as user defined data types for the specific application. User data types are supported by this standard as instances of the data types classes.

User defined types are specified in the same manner that all FAL objects are specified. They are defined by providing values for the attributes specified for their class.

5.3.4 Structures types

There are defined as user defined data types for the specific application. User data types are supported by this standard as instances of the data types classes.

User defined types are specified in the same manner that all FAL objects are specified. They are defined by providing values for the attributes specified for their class.

6 Communication model specification

6.1 Concepts

6.1.1 AL environment

This general model presents the relationship between the AL environment and the application layer services which is the purpose of this standard.

The AL environment is both process control and discrete parts manufacturing. In those systems, manufacturing devices are controlled by a system that performs the automation functions.

6.1.2 Application services

Application services can be divided into two sets. See Figure 1.

The standard specifies the MPS (Manufacturing Periodic/aperiodic Services) application service set. The other service set corresponds to a subset (subMMS) of that specified in the MMS standard.

MPS services are particularly well suited to the real time requirements of a distributed application in the fact that they support:

– The periodic broadcast update of a distributed database in the various control system devices.

– The elaboration of temporal qualifiers associated with the data base.

– The elaboration of consistency qualifiers associated with sets of data within this database.

SubMMS and MCS are essentially considered to fulfil the requirements for configuration and operating mode management within the distributed application.

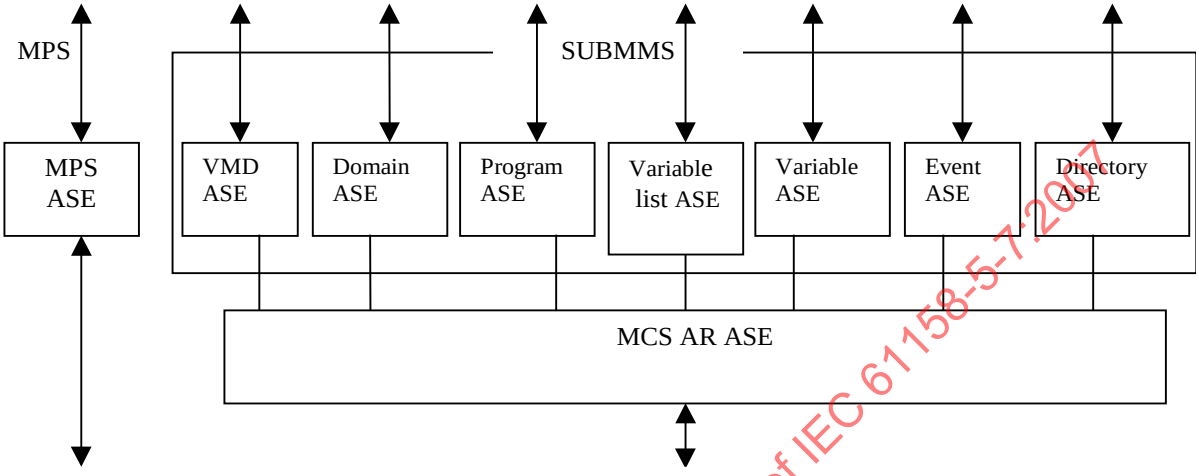


Figure 1 – Organisation of the ASEs and ARs

A distributed application is characterized by various information processing tasks on various devices of a communication system for the requirements of check a production system control application.

Cooperation between the tasks belonging to different devices is modelled in the form of interactions between application processes (APs).

A distributed application is composed of two or more APs, each of the devices of the communication system which can have zero, one or several of them.

As provided for by this standard, the communication between two APs passes via the network.

NOTE The breaking down of a distributed application in several AP's at the level of a device is decided by the user according to his own criteria (Methodology, Functional structuring of the application, Setting into service facilities, etc.).

6.1.3 Application process

6.1.3.1 Application process overview

An AP is modelled by one or several information processing tasks in a device for the requirements of a particular function of the distributed application.

The aspects of an AP, taken into account by the AL, only concern its communication capacities which are modelled by Application Entities (AEs). Each AP can have at the most one AE of a given type.

On the other hand, the cooperation between APs in the framework of a distributed application are performed via the setting up of the relations between AP invocations (APIs) which represent its activities. An AP can, at a given instant, zero, one or several APIs. Besides, each API has one particular utilization of the AE (AEI) which satisfies the communication requirements.

6.1.3.2 AP model

6.1.3.2.1 AP model description

The purpose of the AP model is to show all the aspects of an AP which are taken into account by the AL communication.

NOTE The other aspects concerning the type of information, processing and execution of the latter are not modeled, since they are not involved at the communication level.

***Class*: APPLICATION PROCESS (AP)**

Key Attribute: AP title

Attribute: AP type

Attribute: List of Inherit from Application Entity

Attribute: List of Reference API

6.1.3.2.2 Attributes:

AP title:

A title is attributed to an AP to identify it, among all the other APs, in a non-ambiguous manner in a multisegment AL network. For the requirements of the universal identification of an AP in the AL environment, this AP tile is involved as a constituent of the ISO name proper to AL user site.

This identification can be structured from the name of the device to which it belongs.

NOTE To structure this identification, this AP should remain permanently in the equipment.

AP type:

The type of an AP helps to designate a set of APs to which it belongs.

NOTE This type is not covered by a title which can be handled in the AL environment.

List of Inherit from Application Entity

The Application Process class inherits an Application Entity class list which gives it the capacity to communicate in the AL environment.

Thus an AP has one or several application entities, which are necessarily of different types.

List of Reference API:

Designates the lists of APIs which model the AP activities. This list can vary dynamically but at a given instant zero, can designate one or several APIs.

6.1.3.3 API model

6.1.3.3.1 API model description

An API model describes the activities of an AP.

***Class*: APPLICATION PROCESS INVOCATION (API)**

Key Attribute: API identification

Attribute: Reference Application Process

Attribute: *List of Reference* AEI

6.1.3.3.2 Attributes:

API identification:

Identifies an Application Process Invocation in a non-ambiguous manner in the given AP field. It is underlined that the scope of this identification is global for the AP and independent of the rules for naming the AP title.

Reference **Application process**

The purpose of this attribute is to show that this API belongs to a given AP.

The lifetime of an API is necessarily less than or equal to that of the AP to which it belongs.

List of Reference AEI:

This list designates AE uses which are reserved for the API requirements.

An API can use zero, one or several AEIs for each AE of the AP to which it belongs.

The number of referenced AEIs can vary dynamically in time. Consequently, an API, at a given moment, can have one or several AEIs (Complete definition of an AEI in the following paragraph).

6.1.4 Application entity

An AE represents a set of communication capacities for the requirements of an AP. These communication capacities are defined by a set of Application Service Elements (ASEs). The particular uses of the communication capacities of an AE are represented by AE invocations (AEIs).

An AEI consequently models the communication functions and their state for the particular activities of an API. Communications between APIs take place via AEIs which, as required, may or may not be related by Application Associations (AAs) or Application Relationship (AR) in the AL context.

An AEI in the AL environment gives an API the exclusive possibility of communicating with or without association. An AEI can only participate in one and in only one association application.

The communications without association allow an AEI to exchange information with one or several other AEIs.

An association application allows an AEI to communicate with one or several other AEIs in the framework of a common Context Application (CA).

This context can be negotiated or predefined by configuration for each AEI, for a point to point association which makes use of a pair of AEIs. The context is necessarily prenegotiated for each AEI.

The state of an AEI is thus directly linked to the fact that the latter participates or not in an association and to the state of the association if it participates in a negotiated association.

Generally it can be said that the lifetime of an AEI is equivalent to that of the service exchanged for the communications without association and equivalent to that of the association for communications with associations. However, for a negotiated association, the lifetime of the pair of AEIs is lengthened by their opening and closing transients.

The lifetime of an AEI is controlled by the API which it represents in the AL environment. An API, on the other hand, can have a much longer lifetime than that of its AEIs, equal to that of the AP to which it belongs.

Particular AR are predefined by configuration, these AR are only used by the MPS ASE to exchange plain data (variable). All the parameters relative to the variable are pre-established at the configuration, these parameters never change. Only a new configuration can cancel or modify these particular AR. The identification of this peculiar AR is made by a number unique on the network, this number refer to the whole set of parameters of the AR and the DLL resources implied for the communication. This number similar to a DLCEP Id is called Identifier.

6.1.5 AE model

6.1.5.1 AE model desription

Description of the communication capacities offered by an AE.

***Class*: APPLICATION ENTITY (AE)**
Attribute: AE type
Key Attribute: Qualifier
Attribute: List of Inherit from ASE
Attribute: List of Reference AEI

6.1.5.2 Attributes:

Qualifier:

An AE qualifier identifies an AE non-ambiguously in the field of an application process. The combination of this AE qualifier with the AP title forms an AE title which identifies this AE non-ambiguously in the AL environment.

This AE title allows recovery from the directory of the addressing information, the link addresses.

An AE title can have "Synonyms of the AE title" to be distinguished from one or several other AEs by different titles.

An AE can be globally identified with other AEs with the help of an "AE group designation".

AE type:

The AE type designates a set of AEs sharing the same communication characteristics. Various AEs of the same type inherit the same list of application service elements.

List of Inherit from ASE

The Application Entity (AE) class inherits an ASE class list which gives it the specificity of its communication functions. An AE can support one or several ASEs, the latter being necessarily of different types.

List of Reference AEI

Designates the list of AEIs which model the AE uses. This list, which can vary dynamically , can designate at a given moment, zero, one or several AEIs.

6.1.6 AEI model

6.1.6.1 AEI model description

An AEI model is used to describe a particular use of the AE.

***Class*: APPLICATION ENTITY INVOCATION (AEI)**
Key attribute: AEI identification
Attribute: Reference AE
Attribute: *Reference* API
Attribute: Local AE address
Attribute: Remote AE address
Attribute: Communication mode
[ASSOCIATED; NON-ASSOCIATED]
Constraint: Communication mode = ASSOCIATED
Attribute: Association state
[OPENING;
ESTABLISHED;
CLOSED]
Attribute: Association type
[NEGOTIATED; PRENEGOTIATED]
Attribute: Association type
[NEGOTIATED; PRENEGOTIATED]
Attribute: Local AEI access point
Attribute: Remote AEI access point
Attribute: *Inherit from* Application context

6.1.6.2 Attributes:

AEI identification:

Allows non-ambiguous identification of an application entity invocation in the sub-field of an AE reserved for an API of a given AP.

The Non-ambiguous designation of an AEI in the environment requires indicating: AP title / API identification / AE qualifier /AEI identification.

Reference AE:

This reference is used to designate the AE object to which this AEI belongs,

The lifetime of an AEI is necessarily less than or equal to that of the AE to which it belongs.

Reference API:

This reference helps to designate the API object which is represented in the AL environment by this AEI. An AEI can represent one and only one API in the AL environment.

Local AE address:

Corresponds to the link address which allows locating the AE to which this AEI belongs.

This address is completed with the help of directory functions during the creation of this AEI and remains static for its lifetime.

Remote AE address:

Corresponds, either to a link address which allows locating the AE with which this AEI is corresponding, for point to point exchanges, or with a link address which helps to locate an AE group with which this AEI is corresponding for multipoint exchanges.

This address is filled at the AEI level during the opening of an association with one other AEI or when exchanging data in a non-associated mode.

Communication mode [ASSOCIATED; NON-ASSOCIATED]

(The communication mode in the ASSOCIATED state indicated that this AEI allows an API to communicate in a application context which was previously defined or negotiated. The type of exchanges between AEIs, possible in such a communication mode, is point to point or multipoint.

When the communication mode is NON-ASSOCIATED, this AEI allows an API to communicate outside all pre-established context. The nature of the possible exchanges in a non-associated communication mode is point to point or multipoint.

Association state

[OPENING;
ESTABLISHED;
CLOSING]:

In the associated mode, the possibilities of communication of an AEI with another AEI depends on the state of the association. The various states correspond to the phases in the life of an association application:

OPENING:

Corresponds to an AEI which, at the request of the API which it represents, is negotiating an application context with another AEI, for the requirements of an association. In this state, no exchanges can take place outside those reserved for negotiation of the context.

ESTABLISHED:

Corresponds to an AEI which offers the API which it represents, the possibility of communicating with another API via one of its determined AEIs, within the framework of an application context. This application context has been obtained either by negotiation between the AEI pair or by local predefinition at the AEIs.

CLOSING:

Corresponds to an AEI which, at the request of the API which it represents, or with which it had been associated, is being freed from its application context.

Association type

[NEGOTIATED;
PRENEGOTIATED]

This type indicates if the association has been negotiated or not. This attribute, if it takes the PRENEGOTIATED value, imposes on the association to remain exclusively in the ESTABLISHED state.

Local AEI access point:

Each AEI, in the ASSOCIATED mode, can be directly localized by its AEI access point. This addressing information is filled with the help of the directory functions during the creation of an AEI instance and remains static for its full lifetime.

AEI remote access point:

Each AEI, which is in the ASSOCIATED mode, has this addressing information which localizes the AEI with which it is in correspondence for point to point exchanges. This information is filled in for an AEI at the opening of an association.

This information locates the AEIs with which it is in correspondence for multipoint exchanges.

Inherit from Context Application

This inheritance allows designating the object of the application context which conditions communications with this AEI.

The attributes of this application context are valued during installation of an AEI on the basis of the values proposed by the user in the negotiated associated mode and the non associated mode, whereas they are filled by configuration in the prenegotiated associated mode.

6.1.7 Application context

6.1.7.1 Application context overview

A pair of AEIs require common knowledge of the rules which govern their communications. This set of rules is called Context Application. The application context which can be supported by an AEI is necessarily derived from the possibilities offered by the various ASEs possessed by the AE to which it belongs.

6.1.7.2 Application service elements

6.1.7.2.1 ASE description

The possibilities offered by the various ASEs of an AE are defined by the specification (in ASN1) of one or more set of Application Protocol Data Units (APDU) which form abstract syntaxes. Besides, the use of an abstract syntax to dialogue between two AEs requires the implementation of a transfer syntax which defines the rules for encoding of the APDUs on the bus.

NOTE Among the abstract syntaxes associated with the ASE MMS, the general abstract syntax ISO-9506-MMS-1 can be noted as well as other abstract syntaxes such as those meant for digital controls ISO-9506-NCCS-1 and process control ISO-9506-PROCESS-1.

Class: ELEMENT SERVICE APPLICATION

Attribute: ASE type

Attribute: List of Abstract syntax

Attribute: List of Transfer syntax

6.1.7.2.2 Attributes:

ASE type:

Several ASE types can be met with in the AL environment. This standard is covered by the SubMMSE definition derived from the ASE MMS defined by the ISO.

The MCSE (Messaging Common Service Element) service corresponds to the control ASE which should necessarily be used to project another ASE application on the link layer messaging. This service element has unique encoding rules which are universally known in the AL messaging environment.

NOTE The MCS encoding rules are not given in ASN1. They are directly given in the representation used for the transfer.

List of Abstract Syntax:

An application service element can have one or several abstract syntaxes. Each of them correspond to a set of APDUs described in an ASN1 module universally identified in the AL environment.

NOTE For example, an abstract syntax defined in ASN1 has the following appearance:

<module name> <module universal identifier>

Definitions::= BEGIN{

IMPORTS <..>, <..>, ...FROM <external module>

EXPORTS <..>, <..>, ...

PDU module ::= CHOICE {

{<production name> [N°] <production>} +

}}

END

This standard keeps AL SUBMMS-1, derived from the reference core, as one of its abstract syntaxes for the MMS service element.

List of Transfer syntax:

A transfer syntax defines the abstract syntax encoding rules used. An ASE could support several transfer syntaxes within the framework of an AE, which could be used according to the requirements depending on the possibilities of their correspondents.

One of the transfer syntaxes taken into account by this AL environment standard for the implementation of the AL abstract syntax, AL SUBMMS-1, is proposed by the ISO basic encoding rules (BER).

Like the abstract syntaxes, the transfer syntaxes are universally identified in the AL environment.

6.1.7.3 Type of application context

6.1.7.3.1 Application context description

The application context belonging to an AEI conditions the possible communications with it. In the information contained in the application context, some is specific to the AE architecture, i.e. with the type of ASE resting on MCS as well as with the abstract transfer syntax selected. Other information is specific to the choice performed on the API level and corresponding to the restrictions to the negotiated abstract syntax and to the quantity of communication resources used.

***Class*: APPLICATION CONTEXT**

Attribute: Context name

Key Attribute: Context identification

Attribute: Reference ASE

Attribute: Abstract syntax

Attribute: Transfer syntax

Constraint: Com. mode = ASSOCIATED

Attribute: Application reductions

Attribute: Service quality

6.1.7.3.2 Attributes:

Identification of the context:

Allows identifying an application context in the field of an AE. It thus has an exclusive local role.

Context name:

Used to identify, in the AL environment, the bases of a context consisting of the selected ASE, an abstract syntax and a transfer syntax. This name is used during the association negotiation or during a transfer of data in the non-associated mode.

Reference ASE:

One of the AE application service elements which is selected by this application context.

Abstract syntax:

An abstract syntax among those of the ASE within the framework of the AE, which is selected by this context.

Transfer syntax:

The transfer syntax selected for encoding the ASE PDUs, implemented within the framework of this AE, which is selected by this context.

Application reductions:

The abstract syntax selected at the level of an AEI application context for a given service element can be used in a reduced manner. This possibility is used only for transmission in the associated mode.

Service quality:

The selected service quality.

6.1.7.4 Directory

6.1.7.4.1 Directory overview

In order to communicate mutually, the AEIs make use of directory functions which given them the addressing information which they require. These directory functions are necessary exclusively during the opening of association or during a transmission in the mode without association.

6.1.7.4.2 Addressing model

The link layer offers data transfer services in the non-connected mode and an addressing space allowing location of the application entities. This addressing space allows locating the entities Individually and/or by groups, and on the other hand to locate them with the help of physical or logic addresses. These two types of addresses respectively allow taking into account or not of the network topology in the location of application entities.

The application layer has a double location requirement, that of the AEs during negotiation of the associations or transmission of data without association, and that of the AEIs during transmission of data within the framework of an association.

Consequently, the AE addresses and the AEI access points which need to be located are addressed with the link addresses from the single link addressing space.

In the case of point to point exchanges, the AE addresses and the AEI access points are designated with the help of individual link addresses. In the case of multipoint exchanges, the AE address and the AEI access point locating the transmitter are designated by individual link addresses whereas the AE address and the AEI access point locating the receivers are designated by group link addresses.

The rules allocating the liaison addresses used for the AE addresses and the AEI access points are not defined in this standard.

The abbreviations used to speak about these two uses of link addresses are: ADAE for those which are associated with AE addresses and ADAEI for those which are associated with AEI access points.

NOTE The ADAEIs are allocated by the directory functions of a device to the AEIs, in the sub-set which was attributed to an AE.

The address values contained in this sub-set can be, for example selected by dividing up the addressing space according to the network architecture or well chosen according to other criteria better satisfying the communication requirements of the user application.

Addressing rules:

Rule_1: Each AE is identified by one and only one "AE title" which is in a bi-univocal relation with an ADAE, allowing it to be located in the AL environment.

Rule_2: Each AE can be designated by one or several synonyms, each being in relation with one and only one address.

Rule_3: Each AE can be designated in the framework of one or several "AE group designations", each being in relation with one and only one address.

NOTE It is important to note that biunivocity is not a property of rules 2 and 3.

Rule_3: Each AEI which is identified by an AEI identification in the field of an AE, an API, of a given AP, is in biunivocal relation with an ADAEI, allowing it to be located in the AL environment.

6.1.7.4.3 Directory functions

The projection between the names allowing identification of the various objects (AEs and AEIs) in the application, and the link addresses (respectively ADAE and ADAEI) used to locate them in the AL environment, is managed by the directory service.

For any communication initiative at the level of an AEI, the latter calls the directory functions which give it the addresses necessary for the exchanges.

— Initiator Directory Function (IDF):

This function is implemented for an AEI initiating an exchange concerning an association opening or a transmission of data in the non-associated mode.

IDF is used to:

– obtain an ADAE pair allowing location of the local and remote AEs according to their AE titles,

– allocate an ADAEI for the local AEI access point and optionally of another for the remote AEI access point. This allocation only takes place when setting up an association.

<Input parameters> ::=

<Local AE title>
<Remote AE title
| Synonym of the AE title
| Designation of the AE group>
<Transmission mode>

with <Transmission mode>::=<ASSOCIATED ï NON-ASSSOCIATED>;

<Output parameter> ::=

<Local ADAE>

<Remote ADAE>

[<Local ADAEI>] (*)

[<Remove ADAE≥] (**)

(*) This ADAEI is only returned in the case of an ASSOCIATED" transmission mode.

(**) This ADAEI is returned or not according to the remote AE title.

Responder Directory Function (RDF):

This function is implemented at the level of a called AE, for the opening of an association.

RDF:

Used to allocate an ADAE at the level of the called entity during the response phase for an association opening request.

<Input parameters> ::=

<Local ADAE>

<Remote ADAE>

<Output parameters> ::=

<Local ADAEI>

NOTE This function is not used on the level of a responder if the initiator of the communication had already performed the allocation by use of the function.

6.1.8 Coordination of the AEIs

6.1.8.1 AEI communication description

The initiative and the type of communications belong to the APIs. The four types of communication are:

– Information exchanges in the non-associated mode.
– Association opening requests.
– Information exchanges in the associated mode.
– Association closing requests

NOTE For each of these types of communication, the AL environment objects are implemented in a specific manner which is covered by the following paragraphs.

6.1.8.2 Information exchanges in the non-associated mode

When an API wishes to initialize a data transfer in the non-associated mode, it proceeds in the following manner:

Exchange scenario:

• It addresses, as applicable:

– either the source AE identified by its <AE title>, indicating the addressee (either by its <AE title>, or by an <AE synonym title>, or by an <AE group designation> and the <Application context>,

– or an AEI whose mode is NON-ASSOCIATED, if the latter has been created to allow a series of exchanges.

• The local procedure which takes place at the AE after this request by the API is as follows:

If the API addresses the AE, the following local procedure is executed:

– Creation of an AEI object in the NON-ASSOCIATED mode,

– Entering of the attributes of this object:

– <Application context> takes the value indicated by the request,

– <Remote AE address> takes the value returned by IDF for the non-associated communication mode,

– <Local AE address>: takes the value returned by ID for the non-associated communication mode.

– Then whatever the addressing which took place, a data transfer service in the MCSE non-associated mode takes place within the framework of this AEI. The user data encapsulated in this data transfer is that of the MCS user application service which was requested.

– The AEI thus used could be maintained or revoked after this exchange. In case of revocation, this will be effective immediately if the service is not confirmed or after reception of the response and passage to API of the data contained in it, if the service is confirmed.

• The following procedure is executed at the addressee AE:

– Depending on whether an AEI does not yet exist or exists already, the following operations are executed or not:

– Creation of an AEI object in the NON-ASSOCIATED mode.

• Filling of the attributes of this object:

– <Application context>: takes the value indicated in the received PDU, if it is supported by the AE.

– <Remote AE address> takes the value contained in the received PDU.

– <Local AE address> takes the value contained in the PDU.

– Identification of an API object designated implicitly or explicitly in the PDU.

– Passage of the <Application service> contained in the PDU received at the API identified for the execution.

– The AEI thus used can be revoked or preserved at the end of this exchange. In case of revocation, this will be effective immediately if the service is not confirmed or after transmission of the response if it is confirmed.

6.1.8.3 Association opening request

If the API wishes to initialize an association opening request, it proceeds in the following manner:

Opening scenario:

• The API locally addresses the calling AE identified by its <AE title>, indicating:

- the called <AE title>,
- the <opening service> (Initiate Rq in MMS) which contains the <Application reductions> of the association, which will be negotiated,
- the <Application context> description other than the application reductions which will be negotiated at the MCS level.

• The local procedure which will be executed at the AE level after this API request, is the following:

- Creation of an AEI object in the ASSOCIATED mode in the OPENING state.
- Entering of the attributes of this object:
- <Remote AE address; local AEI access point>: take the value returned by IDF.
- Consideration of the application context requested by the user, and possible reduction of this, if its local AE cannot offer it.
- A_Associate.Rq service call by MCSE in the framework of this AEI with the application context given by the user as well as the opening request user PDU.
- On reception of the response, the other AEI attributes can be filled:
- <Remote AEI access point>: takes the value contained in the PDU.
- <Application context> tales the value contained in the response which can correspond to a reduction of that requested.
- Passage of the AEI state to ESTABLISHED,

• The procedure which takes place at the responder AE level on reception of an opening request transmitted by the source procedure is the following:

- Creation of an AEI object in the ASSOCIATED mode in the OPENING state.
- Filling of the attributes of this object:
- <Application Context>: takes the value indicated in the PDU, or a reduction, if the AE does not support it.
- <Local AE address> tales the value received in the PDU.
- <Local AEI access point> can take the optional value contained in the PDU or that returned by the RDF (Responder Directory Function).
- <Remote AE address; Remote access point> take the values received in the PDU.
- Addressing of the API from its implicit or explicit identification in the PDU.
- Passing to the user of the opening request PDU.
- When the API gives its response, a call of the A_ASSOCIATE.Rp service of MCSE in the framework of this AEI, with the PDU user, as a response to the opening request and possible updating of the context if this has been reduced by the API.
- Passage of the AEI state to ESTABLISHED.

6.1.8.4 Exchange of information in the associated mode

When an API wishes to initialize an associated mode exchange, it proceeds as follows.

Exchange scenario:

- It addresses the AEI locally in the ESTABLISHED state, identified by its <AEI identifier> relative to its area by indicating:

  – the <application service> which should be executed with the called entity.

- The local procedure which should be performed at the AEI level is as follows:

  - During reception of the response, for a confirmed service, sending of this to the user.

  – Call of the data transfer service in the MCSE associated mode in the AEI framework, with as user data the PDU associated with the service.

- The procedure executed at the responder AEI on reception of the service request is as follows:

  – Passing of the <Application service> contained in the API PDU.

  – Service call and information transfer in the MCSE associated mode, in the framework of the AEI, with as user data the response PDU associated with this service.

6.1.8.5 Request for termination of an association

When an API wishes to initialize a request for termination of an association, it proceeds as follows:

Closing scenario:

• It locally addresses the AEI identified by its <AEI identification> in its area, indicating:

  – the <Closing application service>.

• The local procedure which is executed at the AEI initiating this closing is as follows:

  – Setting of the AEI to the CLOSING state (Blocking of the data transfers beyond what has been engaged).

  – MCSE A-RELEASE service call to transmit the closing request service in the framework of this AEI.

  – On reception of the positive response, revocation of the AEI object, return to the ESTABLISHED state in the other cases;

- The remote procedure which is performed at the responder AEI level is as follows:

  – Setting of the AEI to the CLOSING state.

  – (Blocking of the data transfers outside those engaged).

  – Passing of the closing request service to the user.

  – At a positive response from the user, revocation of the AEI object and sending of the response to the initiator using the MCSE.

6.1.9 Concepts for sub_MMS

6.1.9.1 General

Sub-MMS communications can be established through negotiated or predefined association as well as without association.

The assignment of access rights to association as well as the protections of objects, allow and the protection of objects authorizes selective access to the objects.

6.1.9.2 Sub-MMS communication channels

1- A Sub-MMS communication can be achieved without association:

These communication can take place in point to point (confirmed/unconfirmed services), in multipoint or in distribution (unconfirmed services).

In this case a service request (other than Initiate, Conclude and Abort) should be refused if the service is not supported, if the resources are insufficient, if it is not in compliance with the protocol, etc.

The Initiate, Conclude and Abort services have no significance.

This type of communication does not support the access protection mechanism on the objects.

2- A Sub-MMS communication can be performed in a predefined association:

These communication can take place in point to point (confirmed/unconfirmed services), in multipoint or in distribution (unconfirmed services).

In this case a service request (other than Initiate, Conclude and Abort) should be refused if the service is not supported, if it is not in compliance with the protocol or if the resources reserved for the association are insufficient, etc.

The Initiate, Conclude and Abort services have no significance.

This type of communication does/does not support the access protection mechanism on the objects.

3- A Sub-MMS communication can be performed in a negotiated association:

These communication can take place in point to point (services confirmed/unconfirmed),

In this case a service request (other than Initiate, Conclude and Abort) should be refused if the service is not supported, if it is not in compliance with the protocol, or if the resources reserved for the association are insufficient, etc.

This type of communication does/does not support the access protection mechanism on the objects.

6.1.9.3 Access protection mechanism

6.1.9.3.1 Protection level introduction

The access to Sub-MMS object can be refused to some clients. For this we define the protection levels assigned to an object and access right assigned to the association.

6.1.9.3.2 Protection level assigned to an object

The definition of objects other than the VMD object includes an OPTIONAL attribute called "ACCESS PROTECTION" which determines if an object is protected or not.

An object which is not protected has its "ACCESS PROTECTION" attribute forced to the value "FALSE".

An object which is protected has its "ACCESS PROTECTION" attribute forced to the value "TRUE".

1 – Password,
2 – Access Groups,
3 – Access Rights.

Password:

The "Password" attribute allows restricting access only to clients holding the password.

Access Groups:

"Access Groups" attribute allows to restrict the access only to clients who belong to one of the groups defined by the attribute.

Access Rights:

The "Access Rights" attribute indicates the various possibilities to obtain access to the object.

It also indicates for each of these access possibilities, the services which can be executed as well as the conditions to be fulfilled, as shown in Table 2.

Table 2 – Access protection

Object class	**Executable operations**
Domain	Load, Upload, Delete, Connect to a program invocation
Program invocation	Execute, Initialize, Stop, Delete
Variable	Read, Write
Variable-List	Read, Write, Delete
Event	Read, Alter, Acknowledge
Access granted because of password	**Conditions to be fulfilled**
Yes	a) The association password is identical to that of the object.
No	b) The value of the password of the association is ZERO; condition (a) is not fulfilled
Access granted because of group membership	**Conditions to be fulfilled**
Yes	a) The group word for the association and the object have at least one identical group No
No	b) The value of the association group word is ZERO; condition (c) is not fulfilled
Access granted to all	**Condition to be fulfilled**
Yes	e) The "Access Rights" attribute authorizes access to all clients.

6.1.9.3.3 Level of the access rights assigned to an association

It is possible to assign access rights to predefined or negotiated associations. In this case these access rights are characterized by the parameters "Password" and "Access Groups".

The "Password" and "Access Group" parameters are granted implicitly to the predefined association.

The "Password" and "Access Group" parameters are granted to the negotiated association by the initiate service.

The "Password" and "Access Group" parameters thus characterize the Client's access rights.

6.1.9.3.4 Access control mechanisms for protected objects

Access to the objects can be granted:

a) either when the "Password" attribute of the object is identical to the association "Password" parameter

b) when the "Access Groups" attribute of the object and the "Access Groups" parameter of the association have at least one group No. in common.

c) or to all the clients when the "Access Rights" attribute permits.

NOTE No access restriction is imposed for the use of the following services:
- Get Name List,

- Get Domain Attributes,
- Get Program Invocation Attributes,
- Get Variable Access Attributes,
- Get Variable List Attributes,
- Get Event Condition Attributes,
- Get OD Header/Data type Attributes.

6.1.9.4 Directory management

6.1.9.4.1 General

All the sub-MMS object describers supported by an application form a directory.

The arrangement of the describers in a directory can be either standardized or free.

The standardization of the directory is performed by an object called "OD" (Object Dictionary).

6.1.9.4.2 Use of the "OD" object

It is not mandatory for a device to support the "OD" object.

6.2 ASEs

6.2.1 MPS ASE (periodic/aPeriodic manufacturing services)

6.2.1.1 Overview

In a MPS environment, a certain number of application objects are implemented. These objects come directly from the instantiation of the different classes previously described.

All these application objects representing the network configuration should be described partially or globally in order to allow:

- the user to have access to the application configuration (e.g.: Access to the variable type),
- the checking of the configuration consistency of the application between several subscribers (e.g.: configuration consistency between the producer and the consumers of a variable).

Moreover, this description is specified in this standard in order to ensure the interoperability in the configuration interchanges between different implementations.

6.2.1.2 Object concepts

6.2.1.2.1 Application objects

The description of periodic/aperiodic MPS services leads to describe all the resources involved in the application layer. This specification defines the static part of the specification.

The specification reactive aspects describe the various service elements actions on the application layer resources:

MPS environment application objects are divided into two sets:

• objects related to variables,

• objects related to variable lists.

Variables are abstract elements that associate a transfer syntax and a semantics to a data.

Variables, and related information, are provided to the user by the communication system, either individually, or by lists.

Other variables are provided with an event characteristic to which the user may associate actions relative to the execution of the distributed application.

6.2.1.2.2 Application objects addressing

The key-attributes (A_Name) used to address the MPS application layer objects all belong_to the same addressing space.

Each of these names in the addressing space is defined within a maximum length of 16 characters.

The characters belong to the set defined for the "visible string" type of the abstract syntax notation ASN1 (ISO 8824).

Within that set, the following characters are to be used:

Upper case letters (A..Z)

Lower case letters (a..z)

Digits (0..9)

Underscore (_)

Currency symbol ($)

The "space" character is not to be used.

Upper and lower case letters are considered as different characters.

An A_Name should not begin with a digit.

The "underscore" character as a second character defines a standard-reserved name.

6.2.1.3 Variable access sub-ASE

6.2.1.3.1 Concepts

6.2.1.3.1.1 MPS environment variables

An MPS environment variable is an abstraction in the MPS environment, in which it is referenced by a unique name or description. The data associated with this variable corresponds to the representation handled by the data link services, which is exchanged on the network.

6.2.1.3.1.2 Network memory

Control system devices provide an AP with a value of each variable concerned an image of which is stored in local network memory.

6.2.1.3.1.3 Producer/consumer/third party concepts

Application entities that recognise a variable are provided with a local image of it.

Within a distributed application, one unique application entity is declared as the producer of the variable value whereas one or several application entities are declared as consumers of that value.

Moreover, some application entities, while being neither producer, nor consumer of the variable value, may have a knowledge of it in order to invoke the updating of its value. Such application entities are called third party entities.

6.2.1.3.1.4 Periodicity and aperiodicity concepts

The exchange of variable value between the producing entity and the consuming entities can be performed periodically or aperiodically.

A periodic updating is initiated by the network and is defined during the configuration step.

The updating period is determined by

• the user requirements to fulfil the distributed application,

• constraints related to the network.

Aperiodic updating is initiated by a user, such as the producing entity, a consuming entity, or a third party entity.

6.2.1.3.1.5 Synchronous and asynchronous properties

Application entities handling variables elaborated according to production validity, for the produced variables, and transmission validity for the consumed variables.

Definitions:

Production or transmission validity associated with each variable have an asynchronous character as they are elaborated independently from the network activity.

Production or transmission validity associated with each variable have a synchronous character when they involve synchronization orders issued from the network.

A synchronization order corresponds to the event associated with the emission or the reception of a variable of class SYNCHRONIZATION.

6.2.1.3.2 Variable class specification

6.2.1.3.2.1 Variable class modelisation

A variable in the MPS environment is modelled using the following objects (see Figure 2):

- one *Produced Variable Local Image* object that includes all the attributes necessary to a producer user.
- one or several *Consumed Variable Local Image* object that includes all the attributes necessary to one or several Consumer users.

Those two object types are issued from inheritance from several classes:

The *Produced Variable Local Image* objects are instantiated from the *Produced Variable* class.

The *Produced Variable Class* includes all the attributes proper to a producing entity, and inherits from the *Identified Variable* class. That latter class includes all the attributes that are common to every local image associated with that variable.

Moreover, the *Produced Variable* class optionally inherits from two classes *Refreshment* and *Punctual Refreshment* that include all the attributes necessary to the elaboration of informations relative to the production validity associated with that variable, within the producing application entity.

The Consumed Variable Local Image objects are instantiated from the Consumed Variable class.

The *Consumed Variable* class includes all the variable attributes proper to a consuming entity, and inherits from the *Identified Variable* class. That latter class includes all the attributes common to every local image associated with that variable.

Moreover, the *Consumed Variable* class optionally inherits from two classes *Promptness* and *Punctual Promptness* that includes the attributes necessary to the elaboration of the informations relative to the transmission validity associated with the variable within the consuming application entities.

Each of the classes *Produced Variable* and *Consumed Variable* optionally inherits respectively form a class *Production Resynchronization* and Consumption Resynchronization that include the attributes relative to the double memorisation of a variable in order to allow the providing of the variable:

- to the network by a producing entity,
- to the consuming entities by the network, upon a synchronization order issued from the network.

All the common attributes of a variable are described in a *Variable* generic class.

The variable type is described in a *Type Constructor* class which is referred by the variable and that allow the elaboration of all the type that may exist in the MPS environment.

Concerning the access to a variable, the *Variable Access* class refers to a variable and indicates the access mode that is used.

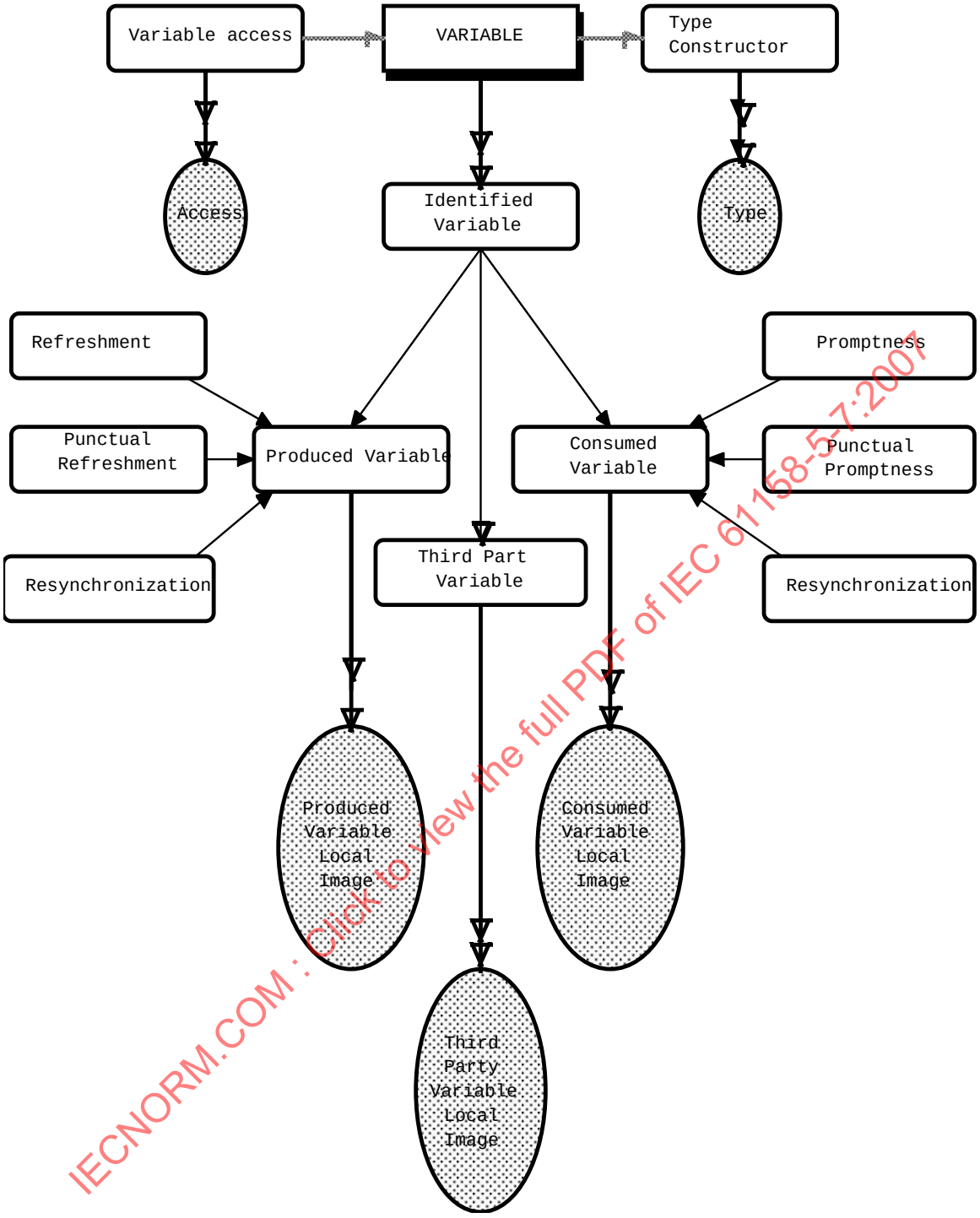


Figure 2 – Object model of the MPS ASE

6.2.1.3.2.2 *Variable* generic class specification

6.2.1.3.2.2.1 Variable generic class model

FAL ASE: **MPS ASE**
CLASS: Variable
CLASS ID: not used
PARENT CLASS: not used
Generic Class: VARIABLE
Class Attributes:

Key-Attribute : A_Name
Attribute : Reference Type Constructor
Attribute : Transmitted status [TRUE, FALSE]
Constraint : Transmitted status = TRUE
Attribute : Significant Status
Attribute : Identifier
Attribute : Transmission mode [PERIODIC,APERIODIC]]
Constraint : Transmission mode = PERIODIC
Attribute : Network period
Attribute : Class [NORMAL,SYNCHRONIZATION,DESCRIPTION]
Constraint : Class = SYNCHRONIZATION
Attribute : Consistency variable [TRUE, FALSE]
Constraint : Consistency variable = TRUE
Attribute : *Reference* Variable List

Instance Attributes:

Attribute : Public value
Constraint : Transmitted status = TRUE
Attribute : Transmitted status value
Attribute : Universal services requested [TRUE, FALSE]
Constraint : Universal services requested = TRUE
Attribute : Universal services scope [LOCAL,DISTANT]
Constraint : Universal services scope = DISTANT
Attribute : Priority [NORMAL,URGENT]

6.2.1.3.2.2.2 Attributes

A_Name:

This attribute uniquely identifies, a variable at the interface between the application layer and the user. The A_Name of a variable belongs to the MPS addressing space.

Moreover, for a variable for which the Class attribute has the value NORMAL, or SYNCHRONIZATION the A_Name length is limited to 14 characters.

***Reference* Type Constructor:**

This reference indicates the A_Name of a global type associated with the variable.

The type is described by a *Type Constructor* object that supports any variable type that may exist in MPS environment.

Transmitted status:

When this boolean attribute has the value TRUE ,status relative to the variable production validity are broadcast to the respective consumers.

Significant status:

For the variables with information relative to their production validity, it is necessary to specify which status are effectively elaborated by the producer.

This attribute is of type Boolean array , each TRUE element indicates, that the production validity status of the associated the variable is significant.

The associated production validity status is defined for the *Transmitted status Value* array element with the same index

The various elements are

S1: Elaboration by the producer of a synchronous or asynchronous refreshment status.

S2: Elaboration by the producer of a punctual refreshment status.

Identifier:

This attribute corresponds to a data link address that refers a data link layer object.

There is a one to one correspondence between a variable A_Name and the associated identifier. Moreover, the relationship between an identifier and a variable A_Name is a local issue committed during the network configuration generation step.

Transmission mode:

This attribute defines the exchanges of the value of the variable it is associated with.

The exchanges may be PERIODIC or APERIODIC, depending on the user requirements, and are defined during the network configuration step.

Network period:

When the transmission mode is PERIODIC, this attribute specifies the updating period of the variable by the network.

This period is expressed in milliseconds.

Class:

This attribute permits to associate a functionality to a variable. It may have one of the following values:

NORMAL: NORMAL class variables indicate process representation information.

SYNCHRONIZATION: SYNCHRONIZATION class variable indicates:

- Synchronization information of the distributed application,
- information necessary to the spatial consistency mechanism for a variable list.

DESCRIPTION: DESCRIPTIONclass variables features the description of all or part of the attributes relative to a variable , to a variable access, to a variable type, or to a variable list.

Consistency variable:

When this attribute has the value TRUE , the described variable corresponds to a consistency variable involved with the spatial consistency mechanism of a variable list.

Reference Variable list:

This reference to the *Variable List* generic class indicates, for a CONSISTENCY class variable, the variables the reception of which, within an application entity, takes part in the elaboration of the consistency variable value when a spatial consistency mechanism is involved

Public value:

This attribute corresponds to the value of the variable provided to the consumers by the producer.

After instantiation, that attributes includes:

- the public value provided to the network,
- the last public value received from the network

regarding that the instance corresponds to a producer's one or a consumer's one.

At a given time, it is to be considered the value available at the consumer, the value transmitted during the last exchange, and the last value received by each consumer.

The last value received by a consumer is not always identical to the last transmitted value because of possible transmission faults.

Transmitted status value:

This attribute corresponds to the array of production validity status , relative to a variable, and transmitted with that variable value. Each status within the array is a boolean.

All the production validity status associated with the variable value are not necessarily significant.

As for the variable public value, that attribute instantiation includes:

- the status values provided to the network,
- the last status values received from the network

regarding that the instance corresponds to a producer's one or a consumer's one.

The handling at a producer and at the consumers of the set consisting of the variable value and its associated status values is atomic.

Status array elements:

S1 Synchronous/asynchronous refreshment status

S2 Punctual refreshment status

NOTE The abstract syntax defines the array of status.

Requested universal services:

When this boolean attribute has the value TRUE , universal services may be invoked for the variable. in consideration.

Universal services scope:

This attribute is used by the variable access extended services, and defines the service category -local or distant- to use for the universal read/write services.

When that attribute has the value LOCAL , universal services that are used correspond to local services; when it has the value DISTANT , they correspond to distant services.

Priority:

This attribute is used by the variable access extended services and defines the priority level used to request variable updating when the universal services correspond to distant services.

When that attribute has the URGENT value, flow control guarantees that an urgent updating request will be processed before a normal request initiated after it.

6.2.1.3.2.3 *Identified Variable* class specification

The *Identified Variable* class is issued from the instantiation of the *Variable* generic class.

6.2.1.3.2.4 *Produced Variable* class specification

6.2.1.3.2.4.1 Produced variable class model

This class gathers all the attributes relative to a variable within a producer application layer.

FAL ASE: **MPS ASE**
CLASS: Produced variable
CLASS ID: not used
PARENT CLASS: not used
Class: PRODUCED VARIABLE

Attribute	: *Inherit from* Identified Variable
Attribute	: Refreshment elaborated [TRUE, FALSE]
Constraint	: Refreshment elaborated = TRUE
Attribute	: *Inherit from* Refreshment
Attribute	: Punctual refreshment elaborated [TRUE, FALSE]
Constraint	: Punctual refreshment elaborated = TRUE,
Attribute	: *Inherit from* Punctual Refreshment
Attribute	: Resynchronized Variable [TRUE,FALSE]
Constraint	: Resynchronized Variable = TRUE
Attribute	: *Inherit from* Production Resynchronization
Attribute	: Emission Indication[TRUE,FALSE]

6.2.1.3.2.4.2 Attributes

***Inherit from* Identified Variable:**

Produced Variable class inherits from *Identified Variable* class that provides it with the attributes common to all local copies of the same variable within MPS environment.

Refreshment elaborated:

Refreshment is an information relative to a variable production validity.

This information, which is optionally elaborated, tells the consumer users about the production period of the producer user.

The status associated with this information is elaborated within the producer application layer and provided to the network in the status value associated with the variable value.

When this Boolean attribute has the value TRUE , a refreshment status is elaborated within the producer application entity.

***Inherit from* Refreshment:**

The *Produced Variable* class inherits from the *Refreshment* class.

The *Refreshment* class includes all the attributes relatives to the elaboration of the refreshment information associated with a variable, in order to qualify its production.

Punctual Refreshment elaborated:

Punctual refreshment is an information relative to a variable production validity.

This information, that is optionally elaborated, tells consumer users about the respect by the producer user of a writing operation within a time slot associated with a synchronization order issued from the network.

The status associated with that information is elaborated within the producer application layer and provided to the network in the status value associated with the variable value.

When this boolean attribute has the value TRUE , a punctual refreshment status is elaborated within the producer application entity.

Inherit from Punctual Refreshment:

The Produced Variable class inherits from the Punctual Refreshment class.

The Punctual Refreshment class includes all the attributes relative to the elaboration of the punctual refreshment information associated with the variable, in order to qualify its production.

Resynchronized Variable:

Some users produce variables asynchronously with respect to the network.

These variables can be resynchronized within the application layer by the means of a double buffer.

The private buffer is the one provided to the asynchronous producer user. The public buffer is the one provided to the network.

Resynchronization of such a variable consists of transferring from Private Buffer to Public Buffer upon the reception of a synchronization order from the network.

This attribute supports the chosen option for the variable concerned.

When this boolean attribute has the value TRUE , the concerned variable is resynchronized in production.

***Inherit from* Production Resynchronization:**

The Produced Variable class inherits from the Production Resynchronization class.

The Production Resynchronization class includes all the attributes relative to a variable resynchronization.

Emission Indication:

When it has the value TRUE this boolean attribute indicates that an emission indication (A_SENT.ind) has to be returned to the user once the variable has been sent on the network.

6.2.1.3.2.5 Produced variable local image class specification

Produced Variable Local Image object is issued from the instantiation of the *Produced Variable* Class.

6.2.1.3.2.6 Consumed variable class specification

6.2.1.3.2.6.1 Consumed variable class model

This class gathers all the attributes relative to a variable within a consumer application layer.

FAL ASE: **MPS ASE**
CLASS: Consumed variable
CLASS ID: not used
PARENT CLASS: not used
Class: CONSUMED VARIABLE
Attribute : *Inherit from* Identified Variable
Attribute : Promptness elaborated [TRUE, FALSE]

Constraint : Promptness elaborated = TRUE
Attribute: *Inherit from* Promptness
Attribute : Punctual Promptness elaborated [TRUE, FALSE]
Constraint : Punctual Promptness elaborated = TRUE
Attribute: *Inherit from* Punctual Promptness
Attribute : Resynchronized Variable [TRUE, FALSE]
Constraint : Resynchronized Variable = TRUE
Attribute: *Inherit from* Consumption resynchronization
Attribute : Reception Indication [TRUE, FALSE]

6.2.1.3.2.6.2 Attributes

***Inherit from* Identified Variable:**

Consumed Variable class inherits from *Identified Class* that provides it with all the attributes common to every local copy of the same Identified Variable within MPS environment.

Promptness elaborated:

Promptness is an information relative to a variable transmission validity.

This information, that is optionally elaborated, tells a consumer user about the reception of a variable respecting a consumption period

The status associated with this information is elaborated within the consumer application layer and provided to the consumer user.

When this boolean attribute has the value TRUE, a promptness status is elaborated within the consuming application entity.

***Inherit from* Promptness:**

Consumed Variable class inherits from Promptness class.

Promptness class includes all the attributes relative to the elaboration of a promptness status associated with a consumed variable.

Punctual Promptness elaborated:

Punctual promptness is an information relative to a variable transmission validity.

This information, that is optionally elaborated, tells a consumer user about the reception of a variable within a time slot associated with a synchronization order issued from the network.

The status associated with this information is elaborated in the application layer and provided to the consumer user.

When this boolean attribute has the value TRUE , a punctual promptness status is elaborated within the consuming application entity.

***Inherit from* Punctual Promptness:**

Consumed Variable class inherits from the Punctual Promptness class.

Punctual Promptness class includes all the attributes relative to the elaboration at the consumer level of a punctual promptness status associated with a variable.

Resynchronized Variable:

Some users consume variables asynchronously with respect to the network.

These variables can be resynchronized within the application layer by the means of a double buffer.

The private buffer is the one provided to the asynchronous consuming user. The public buffer is the one receiving the value from the network.

Resynchronization of such a variable consists in transferring the public buffer into the private buffer upon a synchronization order from the network.

This attribute supports the chosen option for the variable concerned in consumption resynchronization.

When this attribute has the value TRUE , the associated variable is resynchronized in consumption.

***Inherit from* Consumption Resynchronization:**

Consumed Variable class inherits from the Consumption resynchronization.

Consumption Resynchronization includes all the attributes relative to a variable resynchronization in consumption.

Reception Indication:

When it has the value TRUE this boolean attribute indicates that a reception_indication (A_RECEIVED.ind) has to be returned to the user once the variable has been received from the network.

6.2.1.3.2.7 Consumed variable local image class specification

Consumed Variable Local Image object is issued from the instantiation of *Consumed Variable* class.

6.2.1.3.2.8 Third party variable class specification

6.2.1.3.2.8.1 Third party variable class model

FAL ASE: **MPS ASE**
CLASS: Third party variable
CLASS ID: not used
PARENT CLASS: not used
Class: THIRD PARTY VARIABLE
Attribute : Inherit from IdentifiedVariable

6.2.1.3.2.8.2 Attributes

Inherit from IdentifiedVariable

Third Party Variable class inherits from *Identified Variable* class this provides it with the attributes necessary to know the variable.

Consequently, this variable is only provided with the pair (A_Name, Identifier) of the identified variable it inherits from.

6.2.1.3.2.9 Third party variable local image object class specification

Third Party Variable Local Image object is issued from instantiation of Third Party Variable class.

6.2.1.3.2.10 *Type Constructor* class specification

6.2.1.3.2.10.1 Type constructor class model

FAL ASE: **MPS ASE**
CLASS: Type constructor
CLASS ID: not used
PARENT CLASS: not used
Class : TYPE CONSTRUCTOR
Key–*Attribute* : A_Name
Attribute : Construction
[SIMPLE, ARRAY, STRUCTURE,PREDEFINED, EXPLICIT]
Constraint : Construction = SIMPLE
Attribute : Primitive type
Attribute : Size
Constraint : Construction = ARRAY
Attribute : Compressed [TRUE,FALSE]
Attribute : Dimension
Attribute : *Reference* Type Constructor
Constraint : Construction = STRUCTURE
Attribute : List of
Attribute : Named field [TRUE,FALSE]
Constraint : Named field = TRUE
Attribute : field name
Attribute : *Reference* Type Constructor

6.2.1.3.2.10.2 Attributes

A-Name:

This attribute uniquely identifies an object within the application layer. An_A_Name belongs to the MPS addressing space Moreover, a Type Constructor A_Name should no be longer than 14 characters.

An A_Name beginning with the characters K_ indicates a type reserved by the standard.

Construction:

The specified type can be of SIMPLE construction, and is then directly derived from the primitive types.

The specified type can be of ARRAY construction, and represents an ordered set of elements of the same type; that latter type can be itself of construction SIMPLE, ARRAY, or STRUCTURE. ARRAY construction type elements are increasingly numbered from zero (0) for the first element.

The specified type can be of STRUCTURE construction, and represents a complex type composed of an ordered set of one or several components. These components may have different types of SIMPLE, ARRAY, or STRUCTURE construction.

The specified type can be of PREDEFINED construction, and is then defined within the standard.

PREDEFINED construction types cannot be used to compose types of ARRAY or STRUCTURE.

The specified type can be of EXPLICIT construction in order to characterise a variable the type of which can change dynamically.

EXPLICIT construction types cannot be used to compose types of ARRAY or STRUCTURE construction.

SIMPLE construction types characteristics
Primitive Type:

Primitive types used.

Size:

This attribute has a value and purpose that depends on the value of the Primitive Type attribute defined above. The size, described for each of the primitive types , for Type Constructor class instances, describes a maximum value to the variable.

ARRAY construction types characteristics
Compressed:

The application layer transfer syntax allows to define, for some ARRAY type variables, compressed encoding rules.

Compressed encoding is specified using a boolean attribute.

When this attribute has the value TRUE , the associated variable value encoding is compressed, each time it is possible depending on the array elements type; when this attribute has the Value FALSE, the associated variable value encoding is not compressed.

Dimension:

This attribute defines the number of elements within the ARRAY construction type.

***Reference* Type Constructor:**

An ARRAY construction type is composed of elements of any one type either of construction SIMPLE, or ARRAY, or STRUCTURE.

This attribute references the "Type" object, instantiated from the *Type Constructor* class associated with the array elements

STRUCTURE construction types characteristics
Named field:

When a type is STRUCTURE , each component composing the type corresponds to the semantic representation of the associated variable. In order to permit a partial access on these components they may be designated with a name. When this attribute has the value TRUE , a name is associated with the corresponding component; when it has the value FALSE, no name is associated with the component.

Field name:

This attribute is a character string that uniquely identifies a structured variable component.

NOTE The scope of component names is local to a structure.

***Reference* Type Constructor:**

A STRUCTURE construction type is composed of components of any type. i.e. SIMPLE, ARRAY, or STRUCTURE.

This attribute denotes that the instance of *Type Constructor* class associated with each component is of type STRUCTURE

PREDEFINED construction types characteristics

PREDEFINED construction types are mostly used for description variables and refer to types that are reserved by the standard.

PREDEFINED construction types cannot be explicitly defined because they need a grammatical extension in order to use optional types, alternative types, or variable length types.

NOTE Grammatical extensions that are involved are specified in ISO 8824 standard.

EXPLICIT construction types characteristics

EXPLICIT construction types allow the associated variables to be provided with an explicit transfer syntax, in order to carry the type semantics with the data.

6.2.1.3.2.10.3 Primitive types

Primitive types that are selected, and their characteristics are mostly specified in the ISO 8824 standard.

NOTE The only access mode available for a primitive type variable is global access.

• BOOLEAN

The definition of this primitive type is the same as the boolean type definition in the ISO 8824 standard. For this type, the *Size* attribute should be omitted.

• BIT STRING

The definition of this primitive type is the same as the bit string definition in ISO 8824. For this type the *Size* attribute defines the number of bits in the bit string.

• INTEGER

The definition of this primitive type is the same as the integer type definition in ISO 8824 standard. For this type, the *Size* attribute defines the number of bits that are necessary to have a twos complement representation of all possible values.

• UNSIGNED

The definition of this primitive type is the same as the integer type definition in ISO 8824 with the exclusion of negative numbers. For this type, the *Size* attribute defines the number of bits that are necessary to have a twos complement representation of all possible values.

• OCTET STRING

The definition of this primitive type is the same as the octet string type definition in ISO 8824 standard. For this type, the *Size* attribute defines the number of octets in the string.

• VISIBLE STRING

The definition of this primitive type is the same as the visible string type definition in ISO 8824 standard. For this type, the *Size* attribute defines the number of characters in the string.

• GENERALISED TIME

This primitive type is defined with a fourteen-digits visible string (YYYYMMDDHHMMSS); it is equivalent to a particular form of the generalised time type in ISO 8824 standard. For this type, the *Size* attribute should be omitted.

• FLOATING POINT

This primitive type defines value representation for positive and negative real numbers, including, zero, negative infinite and positive infinite.

The values at the limits (0, +• -•), as well as the various representation formats are defined in IEC 60559.

For this type, the *Size* attribute, of boolean type, indicates whether the representation format is simple precision (FALSE) or double precision (TRUE).

• BINARY TIME

This primitive type is defined as the unsigned type of the present standard, the value of which corresponds to a number of elementary times.

For this type, each value of the *Size* attribute defines the value of an elementary time as well as a number of bits used to represent any possible binary time value, as shown in Table 3.

Table 3 – Binary time coding

Size	10 ms	0,1 ms	1 ms	10 ms
16 Bits	0	1	2	3
32 Bits	4	5	6	7
48 Bits	8	9	—	—

• BCD

This type is used to represent a set of values corresponding to the numeric representation of digits.

For this type, the *Size attribute* should be omitted.

6.2.1.3.3 Basic mechanisms

6.2.1.3.3.1 Time-out class specification

6.2.1.3.3.1.1 Time-out class model

The elaboration, by producing application entities ,of information relative to production validity involves time-out resources. Likewise for consuming application entities.

The description of this resource can be modelled with an attribute object, whose purpose is to define visible aspects of time-out used by MPS environment services.

FAL ASE: **MPS ASE**
CLASS: Time-out
CLASS ID: not used
PARENT CLASS: not used
Class: TIME-OUT
Attribute : Preset
Attribute : Current value
Attribute : State [RUNNING, IDLE]

6.2.1.3.3.1.2 Attributes

Preset:

Preset corresponds to the duration associated with the time-out when it is armed, whenever the time-out is running or expired.

Current value:

Current value corresponds to the instantaneous value of the time-out.

That value is equal to

- the period remaining before expiration, when the time-out is running
– "0" when the time-out has expired.

State:

The time-out state indicates whether the time-out is active or not.

The two states are as follows:

RUNNING:Corresponds to a running time-out within the time interval between preset value and expiration.

IDLE: Corresponds to an expired time-out.

NOTE As soon as the time-out current value is forced with a non-zero value, the time-out is automatically set to RUNNING state.

6.2.1.3.3.1.3 Time-out mechanism

The dynamic working of a time-out is defined by the evaluation net shown in Figure 3.

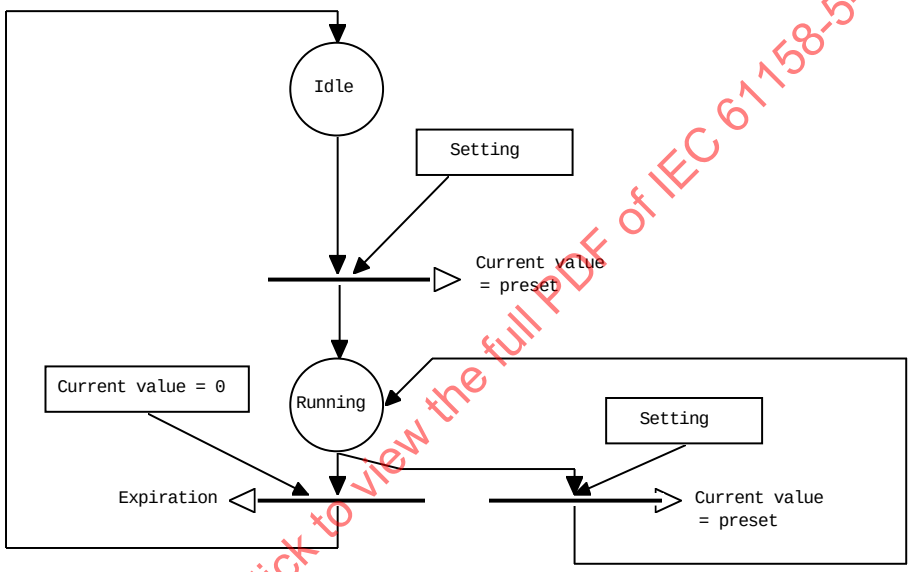


Figure 3 – Time-out evaluation net

6.2.1.3.3.2 Variable addressing

6.2.1.3.3.2.1 Variable addressing description

User access to application variables in MPS environment can be carried out in various ways.

Access to a variable is carried out by means of

- a variable A_Name, in the case of variable global access,
- an access path in the case of explicit partial access to a structured variable named field or to an array element,
- an access A_Name, in the case of renamed partial access to a variable or to one of its components (Structure field, array element…).

Access A_Name belong all to the same MPS addressing space provided to the user at the interface with the application layer.

6.2.1.3.3.2.2 Access modes

6.2.1.3.3.2.2.1 Global access

This access mode allows the user to globally access a variable.

The access to the variable is made with the Application name of the variable that is accessed. In this case, the application name corresponds to the *A_Name* attribute of *Produced Variable Local Image,* or *Consumed Variable Local Image,* or *Third Party Variable Local Image*. It belongs to the MPS addressing space provided to the user at the interface between the application layer and the user.

6.2.1.3.3.2.2.2 Explicit partial access

This access mode allows the user the access to a named field of a structured type variable or to an element of an array type variable.

The explicit partial access is made with an access path corresponding to the enumeration of all successive fields in the structure tree or array elements index; This enumeration should begin from the variable root, to the component that is accessed.

6.2.1.3.3.2.2.3 Renamed access

Renamed access consists in the projection of an A_Name from the MPS addressing space on a variable A_Name, or on a structured type variable access path, or on an array type variable element.

The names that are used in renamed partial access belong to the same addressing space as variable global names; this means they belong to the addressing space provided to the user at the interface with the application layer.

Consequently, renamed access allows the user access to a structure field or to an array element in the same way as for a global variable.

NOTE This access mode is provided to improve transmission efficiency, by grouping several user defined variables in the same structure.

This means that the transmission efficiency improvement can lead to group in a minimum number of DLPDUs variables exchanged between APs in an elementary distributed application. This is done in order to minimise control data in the data link layer.

Optimisation structures created at the application layer are transparent to the user with the renamed access, but they have to be declared in all the application entities that have to access all or part of the user defined variables they represent.

NOTE on optimisation structure elaboration: Only variables which do not need production validity and for which transmission validity is elaborated with the same value of the consumption period can be grouped within the same optimisation structure.

6.2.1.3.3.2.3 Variable renamed access modelization

6.2.1.3.3.2.3.1 *Variable Access* class specification

Class: VARIABLE ACCESS

Key-Attribute	: A_Name
Attribute	: Reference Variable
Attribute	: Access Mode [PARTIAL, GLOBAL]
Constraint	: Access Mode = PARTIAL
Attribute	: Access Path

A_Name:

This attribute allows a unique specification of an access name at the interface between the application layer and the user. The A_Name belongs to the MPS addressing space

Moreover, a variable access A_Name should not be longer than 14 characters.

***Reference* Variable:**

The *Variable Access* class refers to the *Variable* class.

Variable class indicates the application variable with which the access A_Name is associated.

NOTE A variable renamed access name is global, this means that the same access name within several different application entities should refer to the same application layer variable.

Access Mode:

This attribute indicates which access mode is to be used.

When it has the value TRUE , this attributes indicates a global renamed access to an application layer variable.

When it has the Value FALSE, it indicates a partial access to a field of a structured variable or to an array element.

Access Path:

The access path to a structured variable field or to an array element refers, by increasing depth order structure field names or array element index.

NOTE In order to reach a full specification of the access path to a variable component, it is necessary that all the *Named Field* attributes of the *Type Constructor* objects at the various level of the structure have the value TRUE.

6.2.1.3.3.3 Variable transmission validity specification

6.2.1.3.3.3.1 Variable transmission overview

Consumer users may be provided with information relative to transmission validity for a variable.

Two sets of transmission validity information are defined:

- Promptness; relative to transmission validity regarding a user validity delay for the variable that is received: this validity delay is called the consumption period. Semantics associated with this information depends on the synchronous or asynchronous nature of the promptness
- Punctual promptness; that is relative to the transmission validity regarding a time slot associated with a synchronization order issued from the network.

6.2.1.3.3.3.2 Promptness

6.2.1.3.3.3.2.1 Promptness definitons

Promptness attributes associated with variable are optional.

Promptness is relative to the validity of availability of a variable value provided to the user by the network. This validity is elaborated with a maximum consumption period which is fixed by the user.

In the case of synchronous promptness, the semantics of that validity information takes into account the respect of availability by the network of a synchronization order associated with the variable.

Asynchronous Promptness:

When it has the value TRUE, this boolean information indicates in a consumer entity that the variable value has been updated by the network for a delay that is smaller than the variable consumption period.

Synchronous Promptness:

When it has the value TRUE, this boolean information indicates in a consumer entity that the synchronization order has been received and has been followed by (at least) one updating of the associated variable by the network, and that the updating was within the consumption period.

6.2.1.3.3.3.2.2 *Promptness* class specification

This class gathers all the attributes relative to the elaboration of a promptness information associated with a variable within a consuming application entity.

Class:	PROMPTNESS	
	Attribute	: Consumption period
	Attribute	: *Inherit from* Time-out
	Attribute	: Promptness characteristic
		[SYNCHRONOUS, ASYNCHRONOUS]
	Constraint	: Promptness characteristic = SYNCHRONOUS
	Attribute	: *Reference* (Synchronization) Variable
	Attribute	: Promptness status [TRUE,FALSE]

Consumption period:

The consumer user declares a consumption period corresponding with the needs_of the distributed application.

This period corresponds to the minimal period to which the consumer user will access the variable.

Inherit from Time-out:

Promptness class inherits from Time-out class.

Time-out class includes all the attributes relative to a time-out. This resource is mandatory to elaborate an information relative to a variable promptness.

Promptness characteristics:

Promptness may have a characteristic which is proper to the consumer, and that influences the promptness status semantics.

When the characteristic is ASYNCHRONOUS, promptness is elaborated exclusively with attributes proper to the variable.

When the characteristic is SYNCHRONOUS, promptness takes into account not only the variable attributes, but also an event related to the reception of a synchronization order.

***Reference* (Synchronization) Variable:**

This attribute references a *Variable* class for which the *Class* attribute has the value SYNCHRONIZATION. This allows reference to the synchronization order which sets the time-out in the case of synchronous promptness.

Promptness status:

Variable promptness is characterized by a status.

This status is of type boolean and is locally elaborated within each entity having a *Consumed Variable Local Image* associated with the variable.

Promptness status transition conditions from one state to the other are elaborated by the means of

- events corresponding to the variable reception or to the occurrence of the synchronization order associated with the variable,
- the expiration of the time-out associated with the variable,
- the state of the time-out associated with the variable for the processing of synchronous promptness.

6.2.1.3.3.3.2.3 Mechanism

The asynchronous promptness status associated with a variable within a consuming entity is dynamically processed by the mechanism described in Figure 4 and Table 4.

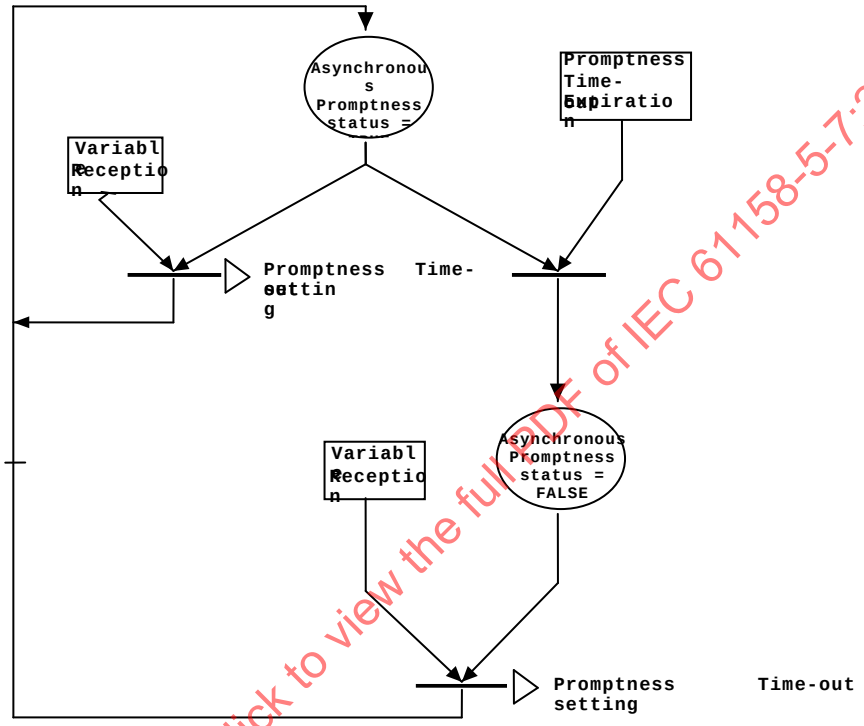


Figure 4 – Asynchronous promptness status evaluation net

Table 4 – Asynchronous promptness events and actions

Initial state: **Asynchronous Promptness status = FALSE**

Promptness time-out state = IDLE

Requests: **Variable Reception:**

Variable available from the network

Promptness time-out expiration:

Transition of the time-out associated with the mechanism elaborating the asynchronous promptness status from RUNNING state to IDLE state.

Actions: **Promptness time-out setting**

Transition of the time-out associated with the mechanism elaborating the asynchronous promptness status from IDLE state to RUNNING state with a preset equal to the *Consumption period* attribute value.

Synchronous promptness status associated with a variable within a consuming application entity is processed by the mechanism described in Figure 5 and Table 5.

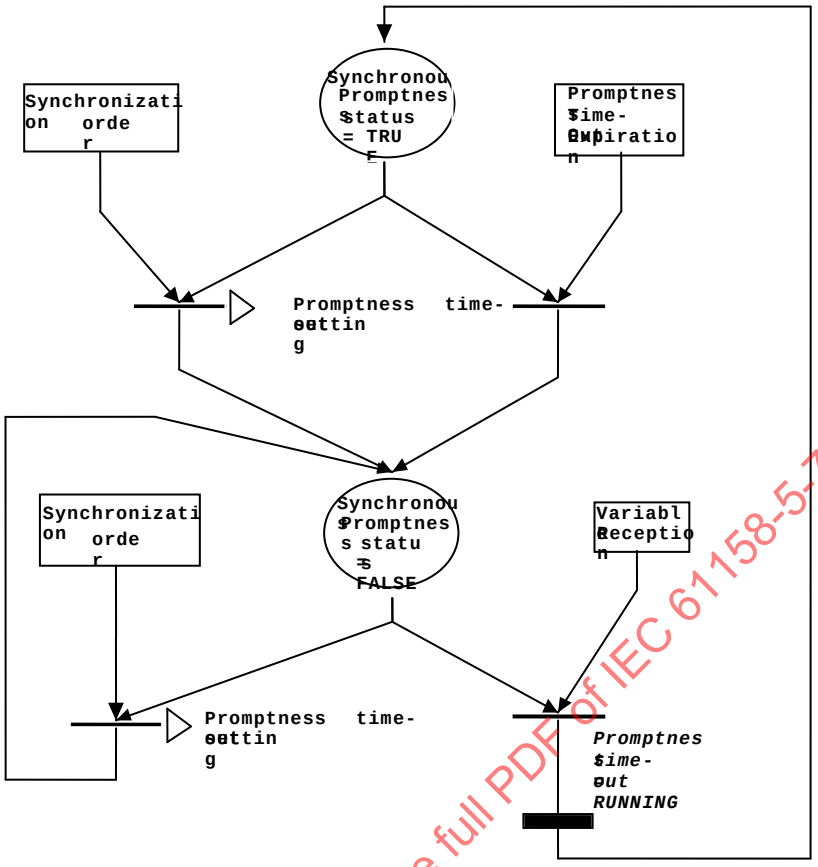


Figure 5 – Synchronous promptness status evaluation net

Table 5 – Synchronous promptness events and actions

Initial state: **Synchronous promptness status = FALSE**

Promptness time-out state = IDLE

Requests: **Variable reception:**

Variable available from the network

Synchronization order:

Occurrence of a synchronization order associated with the synchronous promptness status elaboration mechanism.

Promptness time-out expiration:

Transition of the time-out associated with the mechanism elaborating the synchronous promptness status from RUNNING state to IDLE state.

Actions: **Promptness time-out setting**

Transition of the time-out associated with the mechanism elaborating the synchronous promptness status from IDLE state to RUNNING state with a preset equal to the *Consumption period* attribute value.

6.2.1.3.3.3 Punctual promptness

6.2.1.3.3.3.1 Punctual promptness definition

Punctual promptness information associated with a variable is optional.

Punctual promptness is relative to a validity of availability of the variable value to the consumer user from the network. That validity is elaborated by the means of a consumption time-slot that is defined during the configuration step. This time-slot is activated upon reception of a synchronization order from the network.

Punctual Promptness:

When it has the value TRUE this boolean informs a consuming application entity that the variable value has been updated by the network (at least) once in the time slot following the reception of a synchronization order associated with the variable. It indicates also that the variable value has not been updated outside the time slot.

6.2.1.3.3.3.2 *Punctual Promptness* class specification

This class gathers all the attributes relative to the elaboration of a punctual promptness information associated with a variable within a consuming application entity.

Class: PUNCTUAL PROMPTNESS

Attribute	: Consumption time slot
Attribute	: *Inherit from* Time-out
Attribute	: *Reference* (Synchronization) Variable
Attribute	: Punctual Promptness Status [TRUE,FALSE]

Consumption time-slot:

Punctual promptness is elaborated by the means of a consumption time slot associated with a variable by a consumer user.

Punctual promptness informs a variable consumer that it has received a new value (at least) once during the time slot associated with the last synchronization order that has been received.

Inherit from Time-out:

Punctual Promptness class inherits from Time-out class.

Time-out class includes all the attributes relative to a time-out (preset, state).

***Reference* (Synchronization) Variable:**

This attribute refers to a *Consumed Variable* class the *Class* attribute of which has the value SYNCHRONIZATION , and that defines the synchronization order that sets the time-out in the case of a punctual promptness.

Punctual promptness status:

A variable punctual promptness is characterized by a status.

That status is of type boolean and is locally elaborated within each entity where a *Consumed Variable Local Image* is declared for the variable.

Promptness status transition conditions from one state to the other are elaborated by the means of

- events corresponding to the reception of the variable or to the occurrence of a synchronization order associated with that variable,
- state of the time-out associated with the variable for the punctual promptness processing.

6.2.1.3.3.3.3 Mechanism

Punctual promptness status associated with a variable within a consuming application entity is processed by the mechanism described in Figure 6 and Table 6.

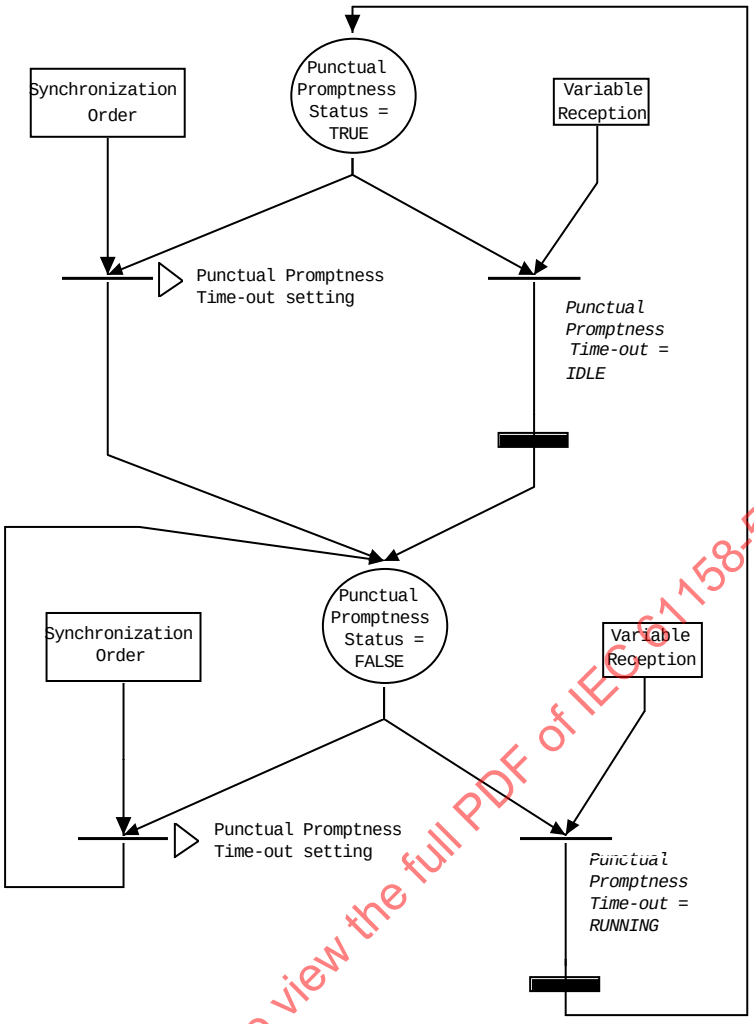


Figure 6 – Punctual promptness status evaluation net

Table 6 – Punctual promptness events and actions

Initial state: **Punctual Promptness status = FALSE**

Promptness time-out state = IDLE

Requests: **Variable Reception:**

Variable available from the network

Synchronization order:

Occurrence of the synchronization order associated with the punctual promptness elaboration mechanism.

Actions: **Punctual Promptness time-out setting**

Transition of the time-out associated with the mechanism elaborating the punctual promptness status from IDLE state to RUNNING state with a preset equal to the *Consumption Time slot* attribute value.

6.2.1.3.3.4 Variable production validity specification

6.2.1.3.3.4.1 Variable production validity definition

Production validity may be associated with any variable. Validity information is elaborated by a producing application entity and provided to the consuming entities. Those consuming entities only see a sampled value of the information.

Two categories of production validity information are defined:

- Refreshment, relative to a production validity regarding a maximum delay fixed by the user. Refreshment semantics depends on its characteristic, synchronous or asynchronous, within a producing application entity.
- Punctual refreshment relative to production validity regarding a production time slot associated with a synchronization order issued from the network.

6.2.1.3.3.4.2 Refreshment

6.2.1.3.3.4.2.1 Refreshment definition

This production validity information is elaborated as an option and is characterized with a status.

Refreshment is relative to validity of availability to the network by a producer user of a variable value. This validity is elaborated by the means of a production period that is fixed by the user.

In the case of a synchronous refreshment, the semantics of this validity information denotes the respect of availability by the network of the synchronization order associated with the variable.

That refreshment information, carried with the variable value by the network, is provided to consumer users.

Asynchronous refreshment:

When it has the value TRUE this boolean indicates in a producing application entity that the variable value has been provided to the network by the producer user, with a delay less than the production period.

Synchronous Refreshment:

When it has the value TRUE this boolean indicates within a producing application entity that a synchronization order has been received and followed by a (at least) one sourcing of a new variable value to the network from the producer user with a delay less than the production period.

6.2.1.3.3.4.2.2 *Refreshment* class specification

This class gathers all the attributes relative to the elaboration of a refreshment information associated with a variable within a producing application entity.

Class:	REFRESHMENT	
	Attribute	: Production period
	Attribute	: *Inherit from* Time-out
	Attribute	: Refreshment characteristic [SYNCHRONOUS,ASYNCHRONOUS]
	Constraint	: Refreshment characteristic = SYNCHRONOUS
	Attribute	: *Reference* (Synchronization) Variable
	Attribute	: Refreshment Status [TRUE, FALSE]

Production period:

The producer user may declare a production period corresponding to the needs of its distributed application.

This period corresponds to the maximum interval during which the producer user will provide the variable to the network.

Inherit from Time-out:

Refreshment class inherits from *Time-out* class.

Time-out class includes all the attributes relative to a time-out. This resource is mandatory to the elaboration of information relative to variable refreshment.

Refreshment characteristic:

Refreshment may be provided with a characteristic proper to a producer and that influences the refreshment status semantics.

When the characteristic is ASYNCHRONOUS, refreshment is elaborated exclusively with attributes proper to the variable.

When the characteristic is SYNCHRONOUS, refreshment takes into account not only the variable attributes, but an event related to the reception of a synchronization order.

***Reference* (Synchronization) Variable:**

The attribute refers to a *Variable* class, the *Class* attribute of which has the value SYNCHRONIZATION. This defines the synchronization order which sets the time-out in the case of a synchronous refreshment.

Refreshment Status:

The variable refreshment provided to a user is characterized by a status.

This status is of type boolean and is locally elaborated within each application entity where a *Produced Variable Local Image* is declared.

The value of the status is associated with the variable value during the exchange on the bus.

Refreshment status is defined by two states, and the transition conditions from one state to the other are elaborated by means of:

- events corresponding to the production of the variable,
- events related to the occurrence of the synchronization order associated with the variable,
- expiration of the time-out associated with the variable,
- state of the Time-out associated with the variable for the processing of synchronous refreshment.

6.2.1.3.3.4.2.3 Mechanism

Asynchronous refreshment status associated with a variable within a producing application entity is elaborated by the mechanism described in Figure 7 and Table 7.

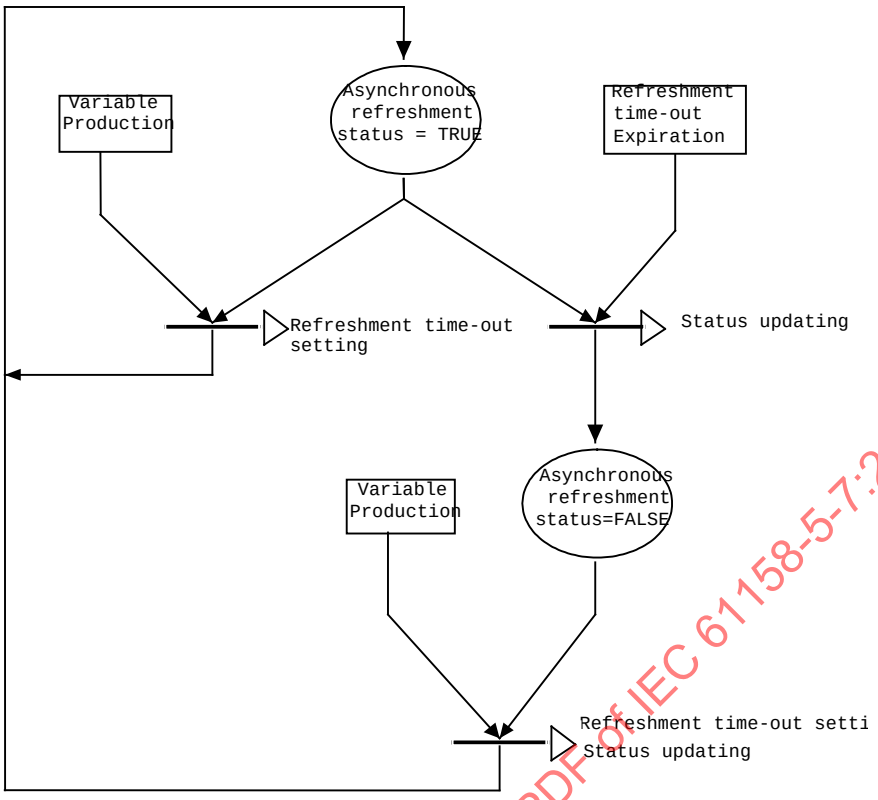


Figure 7 – Asynchronous refreshment status evaluation net

Table 7 – Asynchronous refreshment events and actions

Initial state:	**Asynchronous refreshment status = FALSE**
	Refreshment time-out state = IDLE
Requests:	**Variable Production:**
	A_WRITELOC, A_WRITEFAR or A_WRITE service invocation by the user
	Refreshment time out expiration:
	Transition of the time-out associated with the asynchronous refreshment evaluation mechanism from the state *RUNNING* to the state *IDLE*
Action:	**Refreshment time-out setting:**
	Transition of the time-out associated with the asynchronous refreshment mechanism from the state *IDLE* to the state *RUNNING,* with a preset equal to the *Production Period* attribute value.
	Status updating:
	Providing of the asynchronous refreshment status value to the network.

Synchronous refreshment status associated with a variable within a producing application entity is elaborated by the mechanism described in Figure 8 and Table 8.

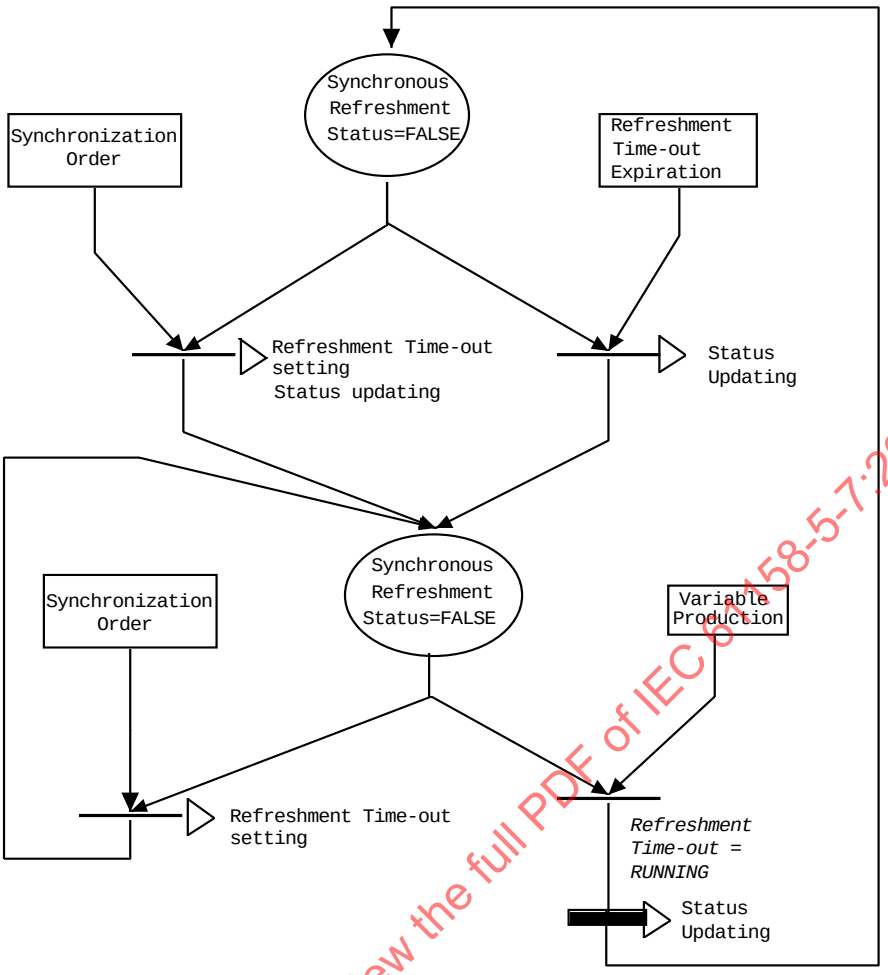


Figure 8 – Synchronous refreshment status evaluation net

Table 8 – Synchronous refreshment events and actions

Initial state: **Synchronous refreshment status = FALSE**

Refreshment time-out state = IDLE

Requests: **Variable Production:**

A_WRITELOC, A_WRITEFAR or A_WRITE service invocation by the user

Refreshment time out expiration:

Transition of the time-out associated with the synchronous refreshment evaluation mechanism from the state *RUNNING* to the state *IDLE*

Synchronization order:

Occurrence of the synchronization order associated with the synchronous refreshment elaboration mechanism.

Action: **Refreshment time-out setting:**

Transition of the time-out associated with the synchronous refreshment mechanism from the state *IDLE* to the state *RUNNING,* with a preset equal to the *Production Period* attribute value.

Status updating:

Providing of the asynchronous refreshment status value to the network.

6.2.1.3.3.4.3 Punctual refreshment

6.2.1.3.3.4.3.1 Punctual refreshment definition

Production validity information is optional.

Punctual refreshment is relative to the validity of availability to the network by the producer user of a variable value. This validity is elaborated by the means of a time-slot fixed by the user. This time slot is activated upon the reception of a synchronization order from the network.

Punctual Refreshment:

When it has the value TRUE this boolean indicates within a producing application entity that a variable value has been provided to the network by the producer user (at least) once during the time slot following the reception of a synchronization order. The variable value should not have been provided outside the time slot.

Punctual refreshment status is associated with the variable value and is provided to the network.

6.2.1.3.3.4.3.2 *Punctual Refreshment* class specification

This class gathers all the attributes relative to the elaboration of a punctual refreshment information associated with a variable within a producing application entity.

Class: PUNCTUAL REFRESHMENT
Attribute : Production time slot
Attribute : *Inherit from* Time-out
Attribute : *Reference* (Synchronization) Variable
Attribute : Punctual Refreshment Status [TRUE,FALSE]

Production Time slot:

Punctual refreshment is elaborated by the means of a production time-slot associated with a variable by the producer user. Punctual refreshment allows a consumer user to check whether the variable producer guarantees its production within the time slot activated by the last synchronization order issued from the network, and associated with it.

Inherit from Time-out:

Punctual Refreshment class inherits from *time-out* class.

Time-out class includes all the attributes relative to a time-out (preset, state) that characterize the production time slot.

***Reference* (Synchronization) Variable:**

This attribute refers to a *Variable* class, the attribute *Class* of which has the value SYNCHRONIZATION, and that permits to define the synchronization order that sets the time-out in the case of a punctual refreshment.

Punctual Refreshment Status:

Variable punctual refreshment is characterized by a status.

This status is of type boolean and is locally elaborated within each entity owning a *Produced Variable Local Image* object of the variable.

Punctual refreshment status is characterized by two states, the transition conditions between which are elaborated by the means of

- events corresponding to the variable production, or to the occurrence of a synchronization order associated with the variable,
- state of the time-out associated with the variable for the processing of the punctual refreshment.

6.2.1.3.3.4.3.3 Mechanism

Punctual refreshment status associated with a variable, within a producing application entity, is elaborated by the mechanism described in Figure 9 and Table 9.

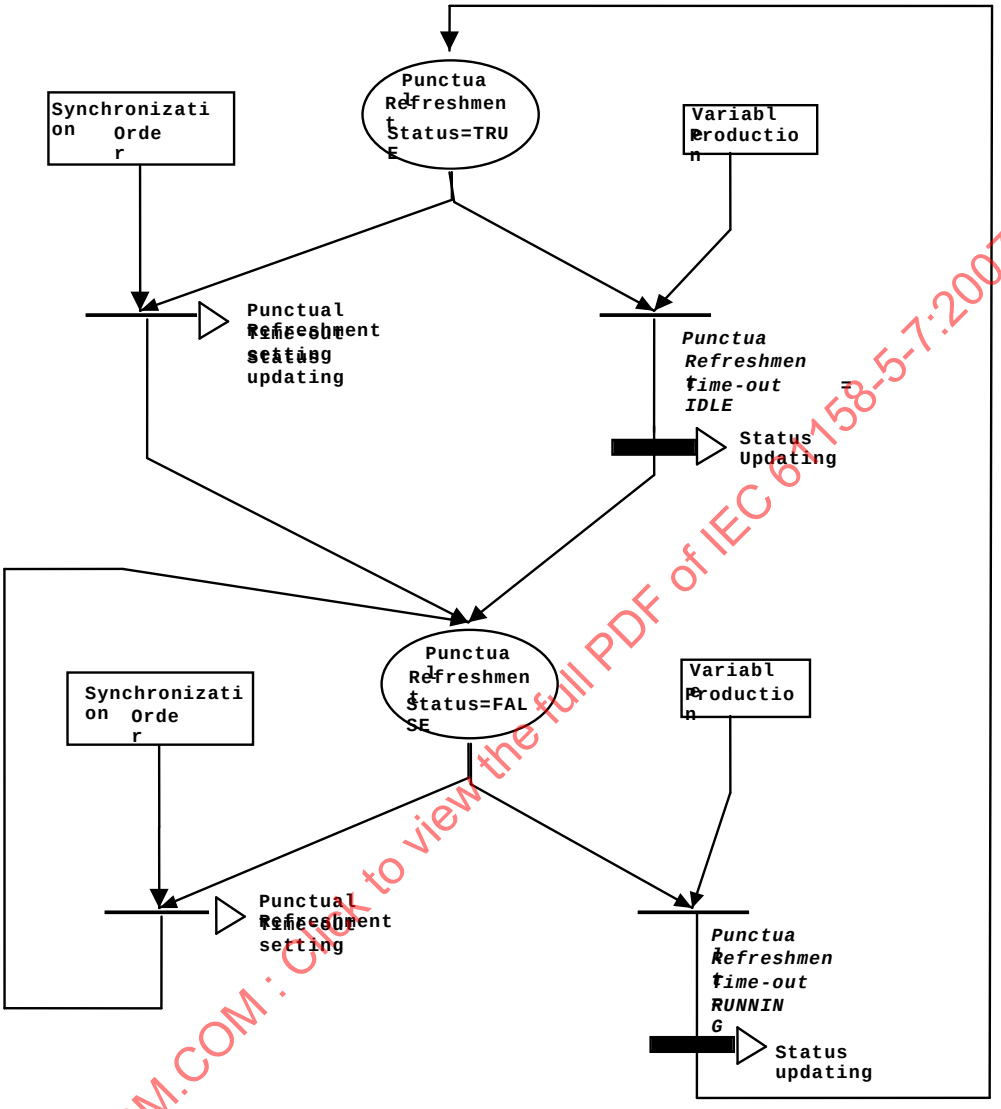


Figure 9 – Punctual refreshment status evaluation net

Table 9 – Punctual refreshment events and actions

Initial state:	**Punctual refreshment status = FALSE**
	Refreshment time-out state = IDLE
Requests:	**Variable Production:**
	A_WRITELOC, A_WRITEFAR or A_WRITE service invocation by the user
	Refreshment time out expiration:
	Transition of the time-out associated with the punctual refreshment evaluation mechanism from the state *RUNNING* to the state *IDLE*
	Synchronization order:
	Occurrence of the synchronization order associated with the punctual refreshment elaboration mechanism.
Action:	**Refreshment time-out setting:**
	Transition of the time-out associated with the punctual refreshment mechanism from the state *IDLE* to the state *RUNNING,* with a preset equal to the *Production Time Slot* attribute value.
	Status updating:
	Providing of the asynchronous refreshment status value to the network.

6.2.1.3.4 Basic services

6.2.1.3.4.1 Basic service overview

The variable access basic services in the MPS environment allow the user to access the associated application objects identified by their specification.

The variable specification corresponds:

- either to the application name for variables accessed in a global way. In this case the application name corresponds to the *A_Name* of the *Consumed Variable Local Image* or *Produced Variable Local Image* or *Third part Variable Local Image* objects.
- or to the partial explicit access name either for structured variables or array type variables partially accessible. This access name is built together with the application variable *A_Name* followed by the access path corresponding to the successive enumeration of field names of the tree of the structure and/or index elements of an array.
- or to the access path in the case of partial or global renamed access on a variable. This name corresponds to the *A_Name* attribute of the Variable *Access* object.

6.2.1.3.4.2 Local read service

6.2.1.3.4.2.1 Functionality

With this service the user can read a variable value or a field of a structured variable value or the element of an array type variable value which is available in the local communication entity.

6.2.1.3.4.2.2 Service primitives

A local read is performed by using A_READLOC service primitives, as shown in Table 10:

A_READLOC.Rq (Argument): Local read request

A_READLOC.Cnf (Result): Local read confirmation

Table 10 – A_Readloc service parameters

Parameter name	Req	Cnf
Argument		
Variable specification	M	M (=)
Result (+)		S
Value		M
Refreshment status		C
Refreshment status		C
Promptness status		C
Punctual Promptness status		C
Result (-)		S
Type of error		M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.		

Argument:

Information associated with a local read request.

Variable specification:

Corresponds to the identification of a *Produced Variable Local Image* object in the MPS addressing space.

Result (+):

Valid answer associated with a local read request

Value:

It is the value of the variable, or field, or array element specified in the request. This parameter corresponds to the attribute *Public value* of the *Consumed Variable Local Image* object for the non resynchronized variables. For resynchronized variables this parameter corresponds to the attribute *Private value* of the *Consumed Variable Local Image* object.

Refreshment status:

Punctual Refreshment status:

Information related to the production validity.

This information come from the attribute *Transmitted status value* of the object *Consumed Variable Local Image*.

Promptness status:

Punctual Promptness status:

Information related to the consuming validity.

This information came from the attributes with same name within the *Consumed Variable Local Image* object.

Result (-):

This indicates that the local read service failed.

Type of error:

Various error types may be returned in the confirmation of a read request.

Error_1: The variable value is temporarily not accessible.

Error_2: The variable value is not accessible by means of this service. It corresponds to a variable which is not locally declared, or which is accessible only by production services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

6.2.1.3.4.2.3 Service procedure

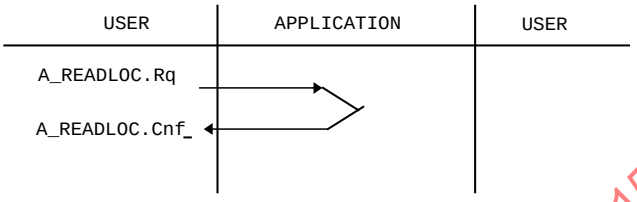


Figure 10 – A_Readloc service procedure

This service, shown in Figure 10, ensures the value integrity of the variable or the field or the element handled.

The value integrity of a variable is ensured in a read operation when all the binary elements of the data associated with this value come from the same update.

Furthermore requests on a variable are processed sequentially: a new request may be made only after reception of confirmation upon the previous one.

6.2.1.3.4.3 Local write service

6.2.1.3.4.3.1 Functionality

With this service the user can write a variable value or a field of a structured variable value or an element of an array type variable value in the local communication entity.

6.2.1.3.4.3.2 Service primitives

A local write is performed using A_WRITELOC service primitive, as shown in Table 11.

A_WRITELOC.Rq (Argument) : Local write request.

A_WRITELOC.Cnf (Result) : Local write confirmation.

Table 11 – A_Writeloc service parameters

Parameter name	Req	Cnf
Argument		
Variable specification	M	M (=)
Value	M	
Result (+)		S
Result (-)		S
Type of error		M

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument:

Information associated with a local write request

Variable Specification:

Corresponds to the identification of a *Consumed Variable Local Image* object in the MPS addressing space. The definition of this parameter corresponds to the one shown in paragraph 3.8.

Value:

Value to be written to the specified variable specification, or field of a structured variable, or element of an array type variable.

This parameter corresponds to the attribute *Public value* of the *Produced Variable Local Image* object for the non resynchronized variables. For the resynchronized variables this parameter corresponds to the attribute *Private value* of the *Produced Variable Local Image* object.

Result (+):

No parameters

Result (-):

This indicates that the local write service failed.

Type of error:

Various error types may be returned in the confirmation of a read request.

Different error types can be returned with the local write confirmation.

Error_1: The variable value is temporarily not accessible.

Error_2: The variable is not accessible with use of this service. It corresponds to a variable which is not locally declared, or which is accessible only by consuming services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

6.2.1.3.4.3.3 Service procedure

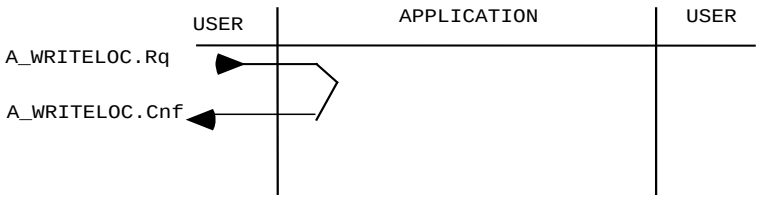


Figure 11 – A_Writeloc service procedure

This service, shown in Figure 11, ensures the integrity of the value of the handled variable.

The integrity of the written variable corresponds to provide the network with a consistent value coming from the same write operation.

All status elaborated by the producing entity and provided to the network should be consistent with the value of the transmitted variable.

Furthermore this service triggers the mechanism associated with the service elements which elaborate refreshment and punctual refreshment status of the *Produced Variable Local Image* object.

Partial access on a variable when writing a value triggers mechanism of production validity status elaboration associated with this variable.

Furthermore request on variable are processed sequentially; a new request can be made only after reception of confirmation upon the previous one.

6.2.1.3.4.4 Updating service

6.2.1.3.4.4.1 Functionality

This service supports the request (to the bus arbitrator) for the updating of a variable value at the consuming entities level. The variable value is locally available in the producer communication entity level.

This service can be used by the application entity which produces the variable, by an application entity which consumes it, or by an application entity which is neither producer nor consumer (ie: a third party entity).

6.2.1.3.4.4.2 Service primitives

An updating request is performed with the following service primitives, as shown in Table 12.

A_UPDATE.Rq (Argument) : Updating request.

A_UPDATE.Cnf (Result) : Updating confirmation.

Table 12 – A_Update service parameters

Parameter name	Req	Cnf
Argument		
Variable specification	M	M (=)
Priority	M	
Result (+)		S
Result (-)		S
Type of error		M

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Arguments:

Information associated with an updating request.

Variable Specification:

Corresponds to the identification of a *Produced Variable Local Image* object or a *Produced Variable Local Image* object or *Third party Variable Local Image* object in the MPS addressing space

Priority:

This parameter allows the user to choose between two flow control possibilities **Urgent** and **Normal** to request the updating of a variable the name of which has been specified in the request.

The flow control ensures that an urgent request, will trigger an updating before a subsequent normal request.

Result (+):

No parameters.

Result (-):

This indicates that the updating service failed.

Type of error:

Various error types may be returned in the service confirmation:

Error_1: The variable is not declared at the AP level of the request initiator.

Error_2: The request cannot be accepted given the local limits concerning the number of outstanding requests.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

6.2.1.3.4.4.3 Service procedure

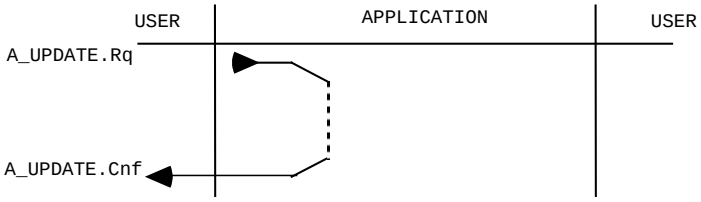


Figure 12 – A_Update service procedure

This service, shown in Figure 12, guarantees that the user will receive a confirmation after a request.

The service confirmation is a positive one when the request is positively confirmed at the data link level.

Requests may be chained together even before receiving all confirmations. However; confirmations are not always issued in the order of requests previously initiated.

For a SYNCHRONIZATION class variable, this service can be initiated only by the variable producer.

NOTE The maximum number of outstanding requests depends on the implementation, and saturation is specified in error codes associated with the confirmation.

6.2.1.3.4.5 Remote read service

6.2.1.3.4.5.1 Functionality

With this service the consumer user of a variable can take the initiative of an information exchange on the bus, before performing a read on the new variable value locally available.

6.2.1.3.4.5.2 Service primitives

Remote read is done with the A_READFAR service primitive, as shown in Table 13.

A_READFAR.Rq (Argument): Remote read request

A_READFAR.Cnf (Result): Remote read confirmation with return of the read value

Table 13 – A_Readfar service parameters

Parameter name	Req	Cnf
Argument		
Variable specification	M	M (=)
Priority	M	
Result (+)		S
Value		M
Refreshment Status		C
Punctual Refreshment Status		C
Promptness Status		C
Punctual Promptness status		C
Result (-)		S
Type of error		M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.		

Argument:

Information associated with a remote read request.

Variable specification:

Corresponds to the identification of a *Consumed Variable Local Image* object in the MPS addressing space.

Priority:

This parameter allows the user to choose between two flow control possibilities **Urgent** and **Normal** to request the updating of a variable the name of which has been specified in the request.

The flow control guarantees that an urgent request, will trigger an updating before a subsequent normal request.

Result (+):

This result indicates that the requested service has succeeded.

Value:

The value of the variable or field, or a array element specified in the request.

This parameter corresponds the attribute *Public value* of the object *Consumed Variable Local Image* for the non resynchronized variables. For resynchronized variables this parameter corresponds to the attribute *Private value* of the object *Consumed Variable Local Image*.

Refreshment status:

Punctual Refreshment status:

Information relative to the production validity This information comes from the attribute *Transmitted status value* of the *Consumed Variable Local Image* object.

Promptness status:

Punctual Promptness status:

Information relative to the consumption validity

This information comes from the attributes with the same name within the object *Consumed variable Local Image*.

Result (-):

This indicates that the requested service failed.

Type of error:

Various error types may be returned in the service confirmation:

Error_1: The variable value is temporarily not accessible.

Error_2: The variable value is not accessible by means of this service. It corresponds to a variable which is not locally declared or which is accessible only by production services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

Error_4: The request cannot be accepted given the local limits concerning the number of outstanding requests.

Error_5: The request has been aborted due to expiration of the time-out associated with the reception waiting of the variable value.

6.2.1.3.4.5.3 Service procedure

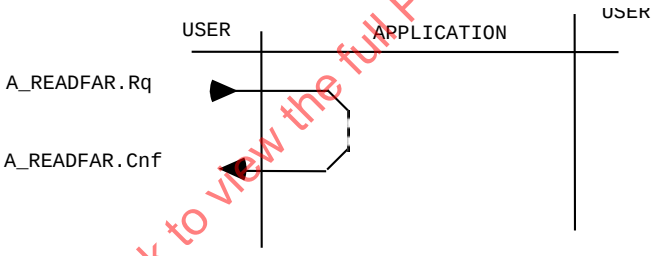


Figure 13 – A_Readfar service procedure

This service, shown in Figure 13, guarantees the integrity of the handled variable value, or field, or array element value.

Requests may be chained together. A new request can be issued without waiting for corresponding confirmations. However, confirmations are not always issued in the same order as the requests.

This service guarantees that the user will receive a confirmation after a request.

The value integrity of a variable value is ensured in a read operation when all the binary elements of the data associated with this value come from the same updating.

6.2.1.3.4.6 Remote write service

6.2.1.3.4.6.1 Functionality

With this service, the variable producer user may initiate an information exchange on the bus after providing the variable value to the network.

6.2.1.3.4.6.2 Service primitives

Remote write is performed by using the A_WRITEFAR service primitive, as shown in Table 14.

A_WRITEFAR.Rq (Argument): Remote write request

A-WRITEFAR.Cnf (Result): Remote write confirmation

Table 14 – A_Writefar service parameters

Parameter name	Req	Cnf
Argument		
Variable specification	M	M (=)
Priority	M	M (=)
Value	M	
Result (+)		S
Result (-)		S
Type of error		M

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Arguments:

Information associated with a remote request

Variable specification:

Corresponds to the identification of a *Produced Variable Local Image* object in the MPS space

Priority:

This parameter allows to the user to choose between two flow control possibilities **Urgent** and **Normal** to request the updating of a variable which the name of which has been specified in the request.

The flow control ensures that an urgent request, will trigger an updating before a subsequent normal request.

Value:

Upon a remote write the user associates with the variable specification the value he intends to affect it.

This parameter corresponds to the attribute *Public value* of the *Produced Variable Local Image* object for the non resynchronised variables. For the resynchronized variables this parameter corresponds to the attribute *Private value* of the *Produced Variable Local Image* object.

Result (+):

No parameters.

Result (-):

This indicates that the remote write service failed.

Type of error:

Different error types may be returned in the service confirmation:

Error_1: The variable value is temporarily not accessible.

Error_2: The variable is not accessible by means of this service. It corresponds to a variable which is not locally declared or which is accessible only by consuming services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

Error_4: The remote write request cannot be accepted due to local limits concerning the number of outstanding requests. However, the local variable value has been updated.

Error_5: The request has been withdrawn

6.2.1.3.4.6.3 Service procedure

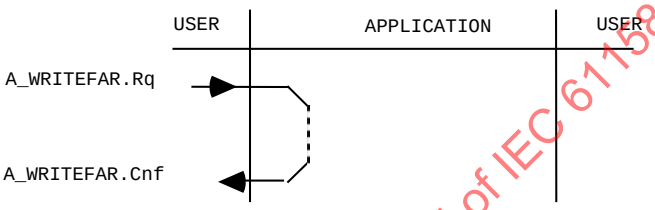


Figure 14 – A_Writefar service procedure

This service, shown in Figure 14, ensures the value integrity of the handled variable or the field, or the element.

Requests may be chained together. A new request can be issued without waiting for corresponding confirmations. However, confirmations are not always issued in the same order as the requests.

This service ensures that the user will receive a confirmation after a request.

The integrity of a written variable value corresponds to an updating of the network with a consistent value coming from only one write operation. This service also ensures the consistency between the produced variable value and statuses related to the production validity which are optionally associated.

Moreover, this service triggers the mechanisms associated with the service elements that elaborate the value of the refreshment status and the punctual refreshment status of the *Produced Variable Local Image* object.

NOTE The value provided to the consuming entities will be the one which was available at the producer level during its network updating.

6.2.1.3.4.7 Transmission indication service

6.2.1.3.4.7.1 Functionality

This service is optional, and indicates to a producer user that a variable has been transmitted on the network.

6.2.1.3.4.7.2 Service primitives

A transmission indication is performed using the A_SENT service primitive, as shown in Table 15.

A_SENT.Ind (Result): Transmission indication of a variable value

Table 15 – A_Sent service parameters

Parameter name	Ind
Result	
Variable specification	M
Result (+)	S
Result (-)	S
Type of error	M

Result:

Variable specification:

Corresponds to the identification of a *Produced Variable Local Image* object in the MPS addressing space.

Result(+):

No parameters

Result(-):

The results indicates that the service failed.

Type of error:

Various error types may be returned by the transmission indication service:

Error_1: The value of the transmitted variable is declared invalid.

6.2.1.3.4.7.3 Service procedure

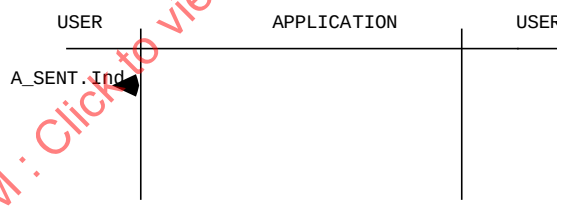


Figure 15 – A_Sent service procedure

This service, shown in Figure 15, informs the producer user of the broadcasting of a variable on the network.

When this variable or some of its fields are locally known at the producer user level by means of names coming from renamed partial access, this service provide an indication to the user for each of the names associated with the same variable.

6.2.1.3.4.8 Reception indication service

6.2.1.3.4.8.1 Functionality

This service informs a consumer user that a variable has been made available by the network.

6.2.1.3.4.8.2 Service primitives

The reception indication is performed by the use of the A_RECEIVED service primitive, shown in Table 16.

A_RECEIVED.Ind(Result): Variable reception indication.

Table 16 – A_Received service parameters

Parameter name	Ind
Result	
Variable specification	M
Result (+)	S
Result (-)	S
Type of error	M

Result:

Variable specification:

Corresponds to the identification of a *Consumed Variable Local Image* object in the MPS addressing space.

Result(+):

No parameters

Result (-):

This indicates that the indication service failed.

Type of error:

Different error type can be returned by the reception indication service:

Error_1: The value of the received variable is declared invalid by the producing application entity.

Error_2: The transmission of this variable failed.

6.2.1.3.4.8.3 Service procedure

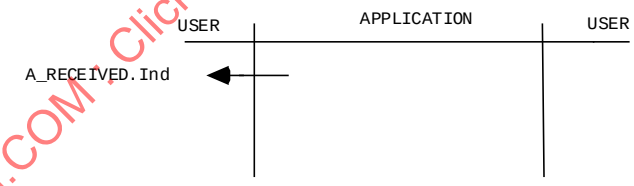


Figure 16 – A_Received service procedure

This service, shown in Figure 16, informs the user about the provision of a variable by the network.

6.2.1.3.5 Enhanced services

6.2.1.3.5.1 Enhanced services definition

Enhanced services are a further level of abstraction and build on the basic services to allow the user to access variables without the need to specify either the type of service (local or remote) or the priority used.

6.2.1.3.5.2 Universal read service

6.2.1.3.5.2.1 Functionality

This service allows the user to read a variable value without knowing the kind of read service (local or remote) to be used; the service is determined by the value of the attribute *Universal services range* of the class *Identified Variable.*

Furthermore when it is a remote read the priority used by the flow control is determined by the value of the attribute *Priority* of the class *Identified Variable.*

6.2.1.3.5.2.2 Service primitives

The universal read is performed using the A_READ service primitives, shown in Table 17.

A_READ.Rq (Argument): Universal read request

A_READ.Cnf (Result): Universal read confirmation with the read value

Table 17 – A_Read service parameters

Parameter name	Req	Cnf
Argument		
Variable specification	M	M (=)
Result (+)		S
Value		M
Refreshment Status		C
Punctual Refreshment Status		C
Promptness Status		C
Punctual Promptness status		C
Result (-)		S
Type of error		M
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.		

Argument:

Information associated with a universal read request.

Variable specification:

Corresponds to the identification of a *Consumed Variable Local Image* object in the MPS addressing space.

Result (+):

Valid answer associated with an universal read request.

Value:

It is the value of the variable, or field, or array element specified in the request. This parameter corresponds to the attribute Public value of the object *Consumed Variable Local Image* for non-resynchronized variables. For resynchronized variables this parameter corresponds to the attribute *Private value* of the object *Consumed Variable Local Image*.

Refreshment status:

Punctual Refreshment status:

Information related to the production validity.

This information is derived from the attribute *Transmitted status value* of the *Consumed Variable Local Image* object.

Promptness status:

Punctual promptness status:

Information related to the consuming validity.

This information is derived from the attributes with same name within the object *Consumed Variable Local Image*.

Result(-):

This indicates that the universal read service failed.

Type of error:

Various error types may be returned in the service confirmation:

Error_1: The variable value is temporarily not accessible.

Error_2: The variable value is not accessible with use of this service. It corresponds to a variable which is not locally declared, or which is accessible only by production services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

Error_4: The request cannot be accepted given the local limits concerning the number of outstanding requests.

6.2.1.3.5.2.3 Service procedure

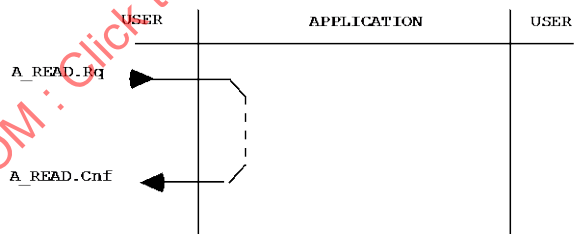


Figure 17 – A_Read service procedure

According to the value of the attribute *Universal services range* of the *Consumed Variable Local Image* object, the invocation of this service implies either the call of the local read service or the call of the remote read service. The universal read service procedure, Figure 17, is the one associated with the invoked service (local or remote).

The priority used in the case of a remote service invocation depends on the value of the attribute *Priority* of the class *Identified variable*.

Figure 18 shows the mechanism for chaining the basic services:

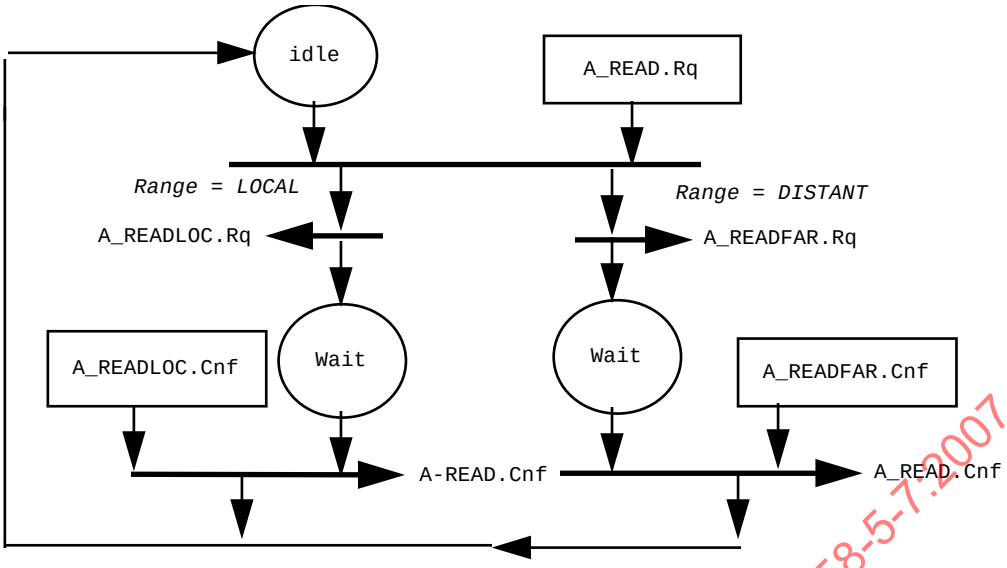


Figure 18 – A_Read service state machine

6.2.1.3.5.3 Universal write service

6.2.1.3.5.3.1 Universal write service functionality

This service allows the user to perform a write of the variable value without having to known the kind of write service (local or remote) to be used. The service is determined by the value of the attribute *Universal service range* of the *Produced Variable Local Image* object.

Furthermore when it is a remote write the priority used by the flow control is determined by the value of the attribute *Priority* of the class *Identified Variable.*

6.2.1.3.5.3.2 Service primitives

The universal write is performed by using the A_WRITE service primitives, shown in Table 18.

A_WRITE.Rq (Argument) : Universal write request

A_WRITE.Cnf (Result) : Universal write confirmation

Table 18 – A_Write service parameters

Parameter name	Req	Cnf
Argument		
Variable specification	M	M (=)
Priority	M	
Result (+)		S
Result (-)		S
Type of error		M

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Arguments:

Information associated with a universal write request.

Variable specification:

Corresponds to the identification of a Produced Variable Local Image object in the MPS addressing space.

Value:

It is the value of the variable, or field, or array element specified in the request. This parameter corresponds to the attribute *Public Value* of the *Produced Variable Local Image* object for non-resynchronized variables. For synchronized variables this parameter corresponds to the attribute *Private Value* of the *Produced Variable Local Image* object.

Result(+):

No parameters.

Result(-):

This indicates that the universal write service failed.

Type of error:

Different error types may be returned in the confirmation of a universal write:

Error_1: The variable value is temporarily not accessible.

Error_2: The variable value is not accessible by means of this service. It corresponds to variable which is not locally declared which is accessible only by consuming services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

Error_4: The request cannot be accepted due to local limits concerning the number of outstanding requests. However, the local variable value has been updated.

Error_5: The request has been withdrawn

6.2.1.3.5.3.3 Service procedure

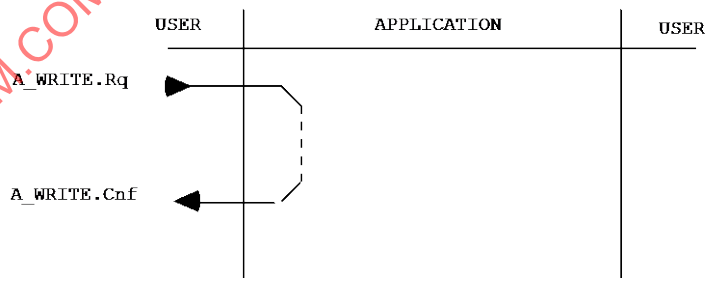


Figure 19 – A_Write service procedure

According to the value of the attribute *Universal services range* of the *Produced Variable Local Image* object, the invocation of this service implies either the call of the local write service or the call of the remote write service. The universal write service procedure, Figure 19, is the one associated with the invoked service (local or remote).

The priority used in the case of a remote service invocation depends on the value of the attribute *Priority* of the class *Identified variable*.

This service triggers the mechanism associated with the service elements that elaborate the refreshment status value and the punctual refreshment status value of the *Produced Variable Local Image* object.

Figure 20 shows the mechanism for chaining the basic services used by the universal write service:

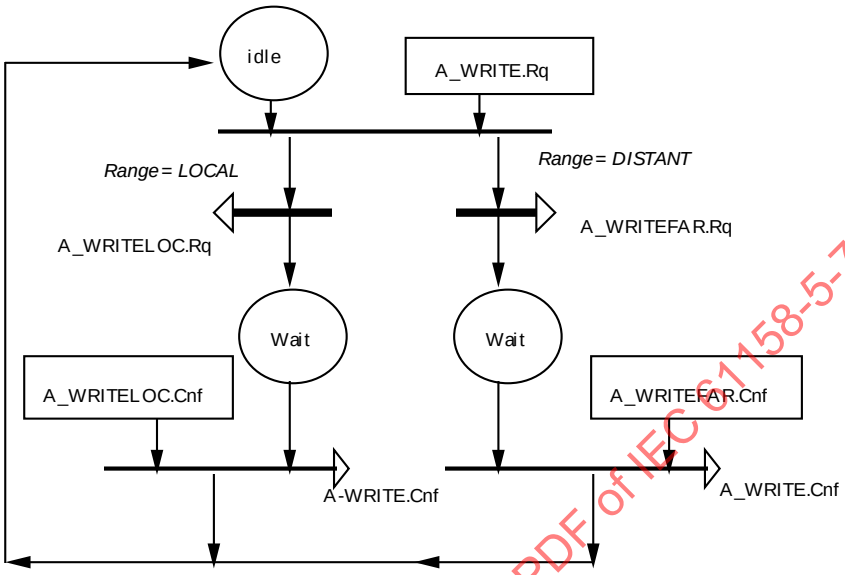


Figure 20 – A_Write service state machine

6.2.1.3.6 Advanced mechanisms

6.2.1.3.6.1 Synchronization

6.2.1.3.6.1.1 Synchronous application concept

The synchronization services provided by the MPS service elements were defined in order to allow the execution of distributed applications.

A distributed application is composed of several Application Processes (AP) in several devices of the control system.

Between the APs which participate in a distributed application, it is necessary to distinguish:

– those which are termed ASYNCHRONOUS, because their execution is independent of the network,

– those which are termed SYNCHRONOUS, because their execution is associated with indications issued from the network; these indications may be periodic or aperiodic.

Several synchronous APs, the execution of which are associated with the same indications, form a synchronous distributed application.

6.2.1.3.6.1.2 Resynchronization needs

Some APs can only support asynchronous operating mode, i.e. they are managed independent of the communication environment.

NOTE This exclusively asynchronous operating mode is usually the characteristic of APs located on devices whose age or simplicity does not offer the possibility of synchronization.

The resynchronization mechanism permits the sample/hold function of the variable values produced and consumed by an AP asynchronously with respect to the Network. This function allows asynchronous APs to participate in a synchronous distributed application.

6.2.1.3.6.1.3 Synchronization variables

The variables in the MPS environment described in the previous chapter correspond to passive elements which are exchanged on the network.

The synchronization variables are elements associated with actions concerning:

– synchronization processing at the level of the producing application entity and consuming application entity of this synchronization variable,
– procedure execution orders for different APs in a distributed application.

At the application entity level, some variables from the MPS environment take into account the event associated with the transmission of a synchronization variable to elaborate the synchronous validity informations, such as synchronous promptness and synchronous refreshment, or the punctual promptness and the punctual refreshment.

6.2.1.3.6.1.4 Synchronization at transmission and reception

Some services provided to the user may be used to synchronize the distributed application procedures using the network.

Among these services, it is necessary to distinguish those which allow:

– synchronization at emission time to ensure the activation of various local tasks inside a single AP,
– synchronization at reception time to ensure the activation management of various tasks of several APs located in several devices of the control system.

This type of synchronization is used by a synchronous distributed application.

6.2.1.3.6.1.5 Synchronization variable definition

6.2.1.3.6.1.5.1 Synchronisation model description

A synchronization variable in the MPS environment is modelled in the same manner as variables using the following objects:

– an object *Produced Variable Local Image* including all the attributes required by a synchronization variable in a producing application entity,
– an object *Consumed Variable Local Image* including all attributes required by a synchronization variable in a consuming application entity.

These two types of object correspond to special instantiations of the class *Identified variable* for which the attribute *Class* has the value SYNCHRONIZATION.

An object *Produced Variable Local Image* or *Consumed Variable Local Image* associated with a synchronization variable differs from the one associated with a normal variable since the value of some of its attributes are predetermined.

6.2.1.3.6.1.5.2 Predetermined attribute values of the *Identified variable* class

The *Identified variable* class associated with a synchronization variable is characterised by predetermined values for some of its attributes:

Class:

This attribute takes the value: *SYNCHRONIZATION*

All the other attribute values are not predetermined.

6.2.1.3.6.1.5.3 Predetermined attribute values of *Produced Variable Local Image* object

The *Produced Variable Local Image* object associated with a synchronization variable is characterised by predetermined values for some of its attributes:

Resynchronized variable:

This attribute take the value: *FALSE*

All the other attribute values are not predetermined.

6.2.1.3.6.1.5.4 Predetermined attribute values of *Consumed Variable Local Image* object

The *Consumed Variable Local Image* object associated with a synchronization variable is characterised by predetermined values for some of its attributes:

Resynchronized variable:

This attribute takes the value: *FALSE*

All the other attribute values are not predetermined.

6.2.1.3.6.1.5.5 Access to synchronization variables

The synchronization variables may be handled by a subset of services provided to the user for normal variables.

The services which provide access to synchronization variables are:

- A_WRITELOC / A_WRITEFAR / A_WRITE
- A_READLOC
- A_SENT
- A_RECEIVED
- A_UPDATE

All the others services offered to the user for a normal variable are not authorized for a synchronization variable.

6.2.1.3.6.1.5.6 Synchronization variable type

The synchronization variables may have a value of any type coming from the instantiation of the *Type constructor* class.

However, a synchronization variable used to meet the requirements of the application service element which elaborate a spatial consistency status, should contain in its value the network exchange number of the list of variables. In this case the variable has a predetermined type.

The exchange number is global data in the MPS environment, produced by a specific application entity. This exchange number qualifies the value of a variable exchanged in a periodic or aperiodic way, relative to the previous exchange.

This exchange number is accessed by the user in the same way as the value of a normal variable, and it is also used by an application service element for the elaboration of the spatial consistency status.

This predetermined type is described by the following instance of the *Type constructor* class.

A_type:	K_UNS16
Construction:	SIMPLE
Class of type:	UNSIGNED
Size:	16

6.2.1.3.6.2 Resynchronization

6.2.1.3.6.2.1 Resynchronisation mechanism principle

The resynchronization mechanism is applied to variables of NORMAL class in order to provide a sample/hold function on asynchronous produced or consumed variable values. This resynchronization mechanism requires that the network transmits synchronization orders in order to:

– trigger the sample/hold of the variables produced by the user.
– trigger the sample/hold of the variables consumed by the user.

Definition:

A resynchronization order corresponds to the synchronization order used by the resynchronization mechanism.

The resynchronization mechanism associates a second buffer with a variable at the application entity level, as shown in Figure 21. A resynchronized variable requires:

– A private buffer (TPP) exclusively accessible by the user.
– A public buffer (TPU) accessible by the network.

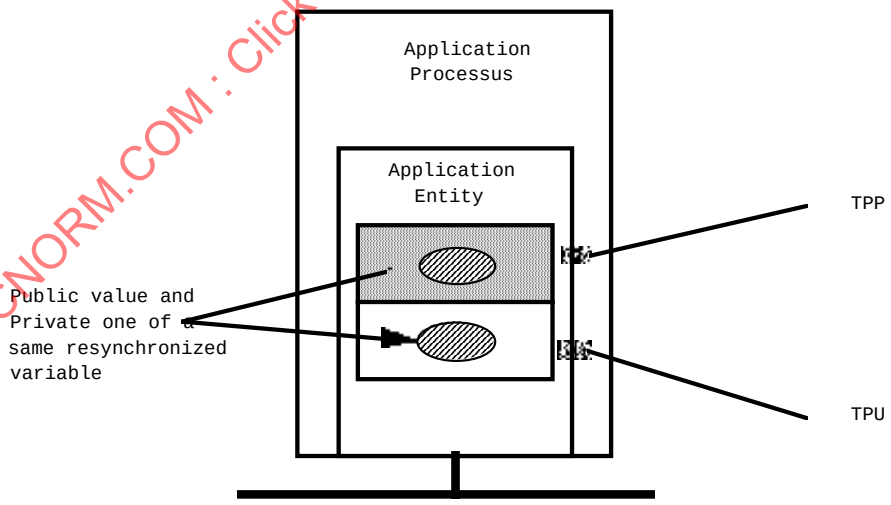


Figure 21 – Model of a resynchronised variable

6.2.1.3.6.2.2 Resynchronization of a produced variable

6.2.1.3.6.2.2.1 Principle

Assume an AP produces a variable in the private buffer (TPP) with a Tp production period. The availability of the value of this variable in the public buffer (TPU) is synchronized by the

network by means of a resynchronization order. This synchronization order is broadcast with a period Tr.

The resynchronization mechanism in production holds the value of the produced variable during the time interval between two occurrences of the same resynchronization order.

Figure 22 shows these principles.

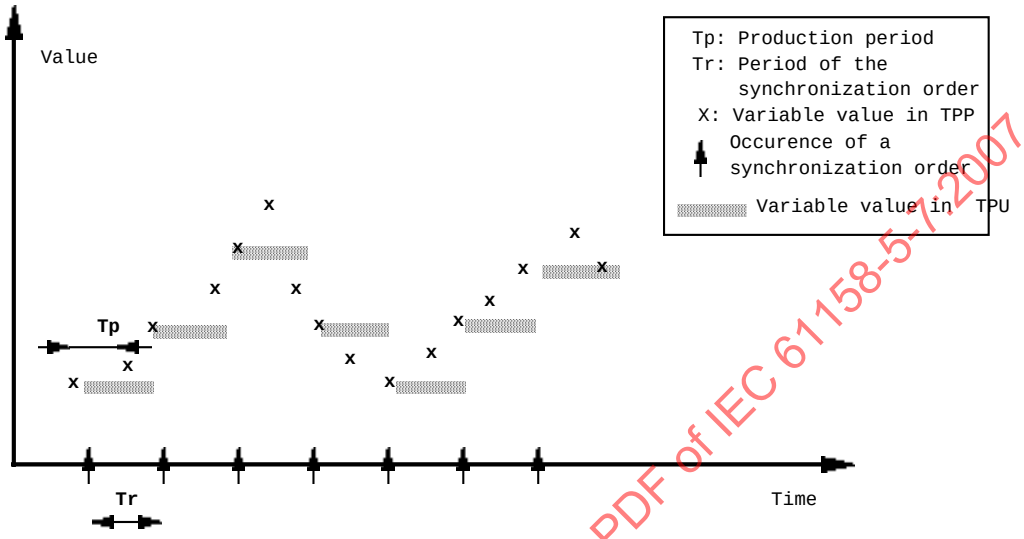


Figure 22 – Principles for resynchronisation of a produced variable

NOTE The resynchronization mechanism of a produced variable is useful only when $T_r >> T_p$.

The resynchronization service implies the use of extra resources like additional buffers for the private values associated with the public values of the resynchronized variables.

A refreshment status elaboration time out and a punctual refreshment elaboration time out are optionally associated with each public buffer of a resynchronized variable.

The resynchronization service requires the use of a private time-out associated with the private buffer in order to allow the production status to preserve their semantics when the variables are resynchronized.

The elaboration of the production validity status for a resynchronized variable at the producer entity level involves the use of two mechanisms associated with the public and private values of the resynchronized variable.

6.2.1.3.6.2.2.2 Specification of the resynchronization in production class

This class describes additional resources used to resynchronize a produced variable at the application entity level.

The class used to describe a resynchronized variable at the producing entity level inherits from the attributes of the *Resynchronization in production* class.

Class: RESYNCHRONIZATION IN PRODUCTION
Attribute : Private value
Constraint : Refreshment elaborated = TRUE
Attribute : Private Refreshment status
Attribute : *Inherit from* Time-out
Constraint : Punctual Refreshment elaborated = TRUE

Attribute : Private Punctual Refreshment status
Attribute : *Reference* Synchronization variable

Private value:

The resynchronization mechanism principle is based on data which is double_buffered. It is necessary to add a private value to the public value of the variable. The private value may now be written asynchronously.

Refreshment elaborated:

This attribute, both in the private context and in the public context, conditions the existence of a refreshment status associated with a variable.

When this attribute is TRUE, a resynchronized variable is provided with a refreshment status and a private refreshment status.

This conditional attribute is declared in the *Produced variable* class.

Private Refreshment status

This private status corresponds to internal information from the production entity of the resynchronized variable, from which is built the value of the refreshment status which will be transmitted with the public value of the variable.

The characteristic of this private status is SYNCHRONOUS or ASYNCHRONOUS according to the global declaration associated with the resynchronized variable in production which is made in the *Refreshment* class.

In the case of a synchronous private status the synchronization variable used by the private mechanism associated with this status is declared globally in the *Refreshment* class.

Inherit from Time-out:

This set of resources allows the user to maintain the semantics of the refreshment status associated with a resynchronized variable.

In fact, the use of a synchronous or asynchronous refreshment status requires the use of an extra time-out to preserve the whole semantics of this status.

Punctual Refreshment elaborated:

This attribute, both in the private context and in the public context, conditions the existence of a punctual refreshment status associated with a variable.

When this attribute is TRUE, a resynchronized variable is provided with a punctual refreshment status and a private punctual refreshment status.

This conditional attribute is declared in the *Produced variable* class.

Private Punctual Refreshment status:

This private status corresponds to internal information from the production entity of the resynchronized variable, from which is built the value of the refreshment status which will be transmitted with the public value of the variable.

In the case of a private punctual status, the synchronization variable used by the private mechanism associated with this status is globally declared in the *Punctual Refreshment* class.

***Reference* (Synchronization) variable:**

The reference to a synchronization variable indicates the resynchronization order associated with a resynchronized variable.

This resynchronization order conditions the transfer of variable values, production status values and current values of time-outs from the private domain to the public domain. This reference indicates the *A_Name* of the *Produced Variable Local Image* object.

6.2.1.3.6.2.2.3 Mechanism associated with the produced value

The resynchronization mechanism in production of the local variable value is made by the transfer from the private buffer to the public buffer, at the reception of resynchronization order.

The evaluation net of this service element which manages the resynchronization actions within a producing application entity for a resynchronized variable is shown in Figure 23.

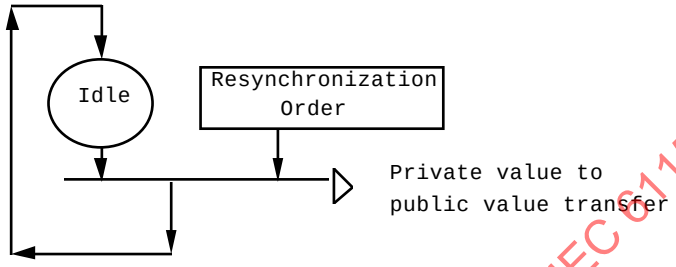


Figure 23 – Resynchronisation mechanism state machine for a produced variable

6.2.1.3.6.2.2.4 Mechanism associated with an asynchronous refreshment

The mechanism which elaborates the asynchronous refreshment status for a resynchronized variable may be described with two evaluation nets one of which describes the private asynchronous refreshment status and the other the asynchronous refreshment status associated with the public value of the variable.

Private mechanism:

The private time-out associated with refreshment is preset with the production period of the variable and set by the user with the act of writing a new private value, as shown in Figure 24 and Table 19.

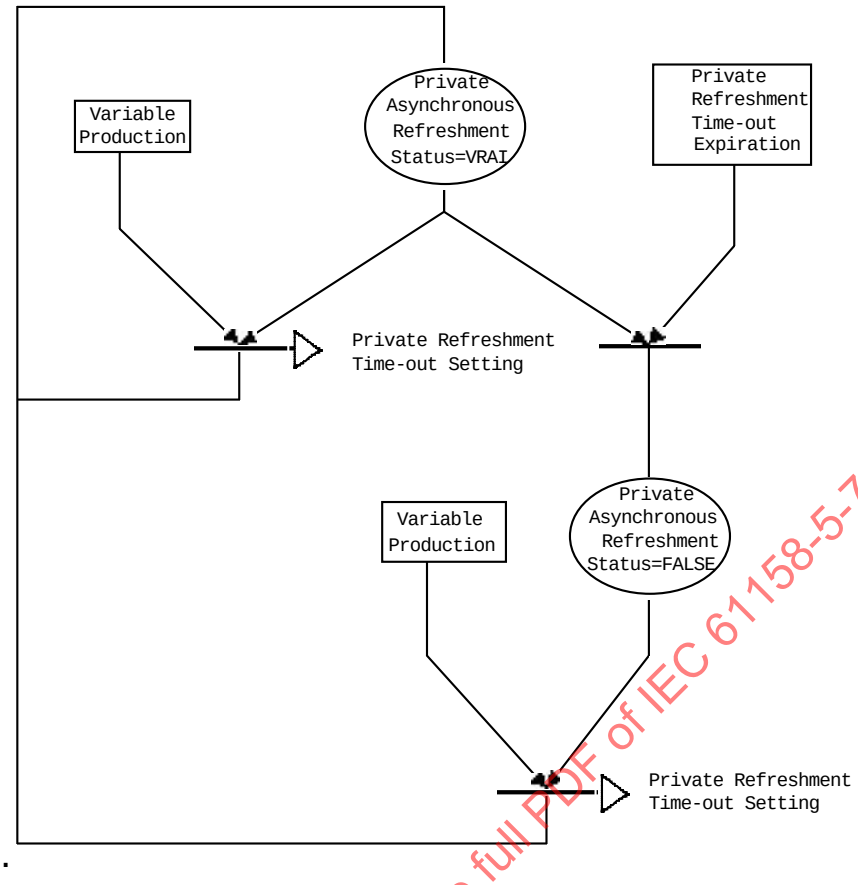


Figure 24 – Asynchronous refreshment private mechanism evaluation net

Table 19 – Asynchronous refreshment private mechanism events and actions

Initial state:	**Private asynchronous refreshment status = *FALSE***
	State of the private refreshment time-out = *IDLE*
Requests:	**Variable Production:**
	Invocation of the services A_WRITELOC, A_WRITEFAR or A_WRITE by the user
	Private Refreshment Time-out Expiration:
	Switch of the private time-out associated with the private asynchronous refreshment status from the *RUNNING* state to the *IDLE* state.
Action:	**Private Refreshment Time-out Setting**
	Switch of the private time-out associated with the private asynchronous status from the *IDLE* state to the *RUNNING* state, with the preset value equal to the Production *period* attribute value.

Public mechanism:

The public time-out is preset with the current value of the private time-out and set on the resynchronization order associated with this status.

The public mechanism to elaborate the asynchronous refreshment status of a resynchronized variable is represented by the following evaluation net, shown in Figure 25 and Table 20.

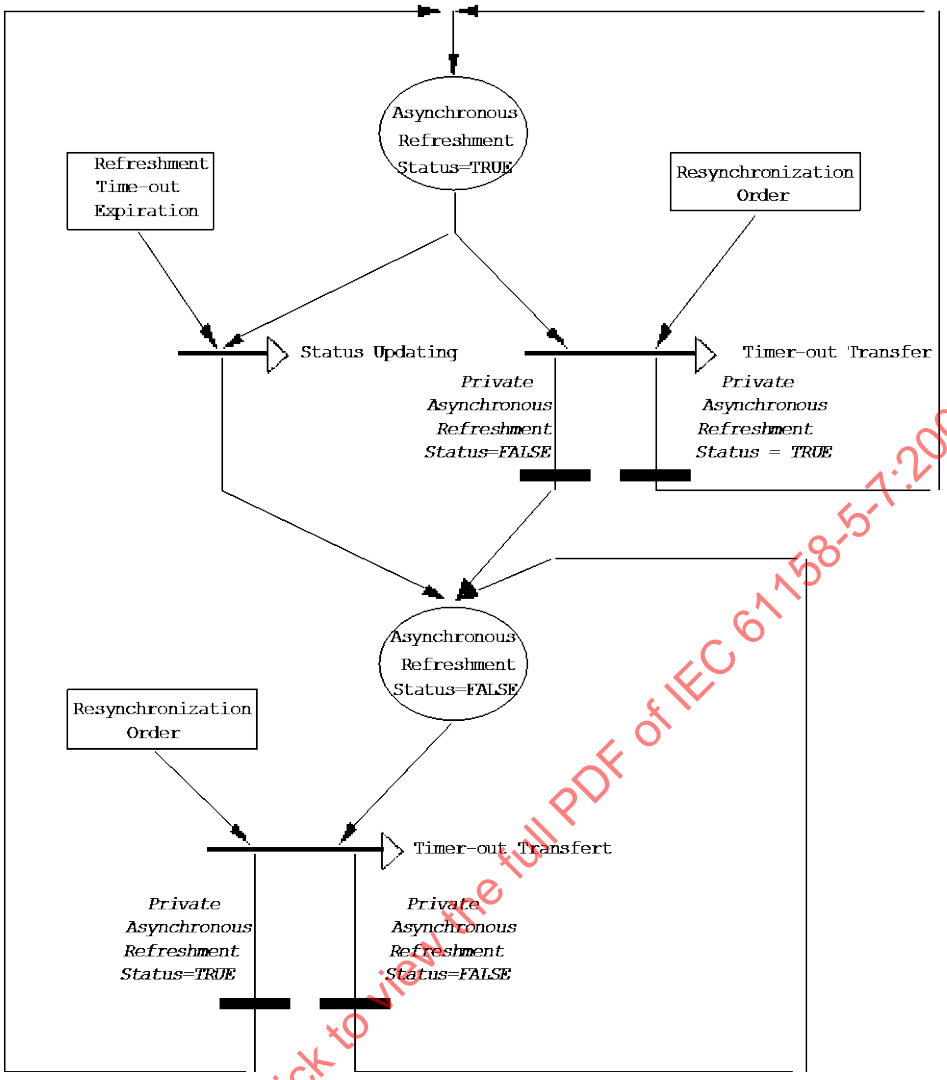


Figure 25 – Asynchronous refreshment public mechanism evaluation net

Table 20 – Asynchronous refreshment public mechanism events and actions

Initial state:	**Asynchronous refreshment status = *FALSE***
	State of the refreshment time-out = *IDLE*
Request:	**Refreshment Time-out Expiration:**
	Evolution of the public time-out associated with the asynchronous refreshment status from the *RUNNING* state to the *IDLE* state.
	Resynchronization Order
	Occurrence of the asynchronization order associated with the public resynchronization mechanism.
Action:	**Time-outs Transfer**
	Transfer in the *State* attribute value and *Current attribute value* of the private time-out of the private asynchronous refreshment status, of the attributes *State* and *Preset* belonging to the public time-out associated with the asynchronous refreshment status.

6.2.1.3.6.2.2.5 Mechanism associated with the synchronous refreshment

The mechanism which elaborates the synchronous refreshment status for a resynchronizes variable may be described with two evaluation nets which one describes the private synchronous refreshment status and the other the synchronous refreshment status associated with the public value of the variable.

Private mechanism:

The private time-out associated with the refreshment is preset with the production period of the variable and set at the receiving time of a new synchronization order associated with this status, as shown in Figure 26 and Table 21.

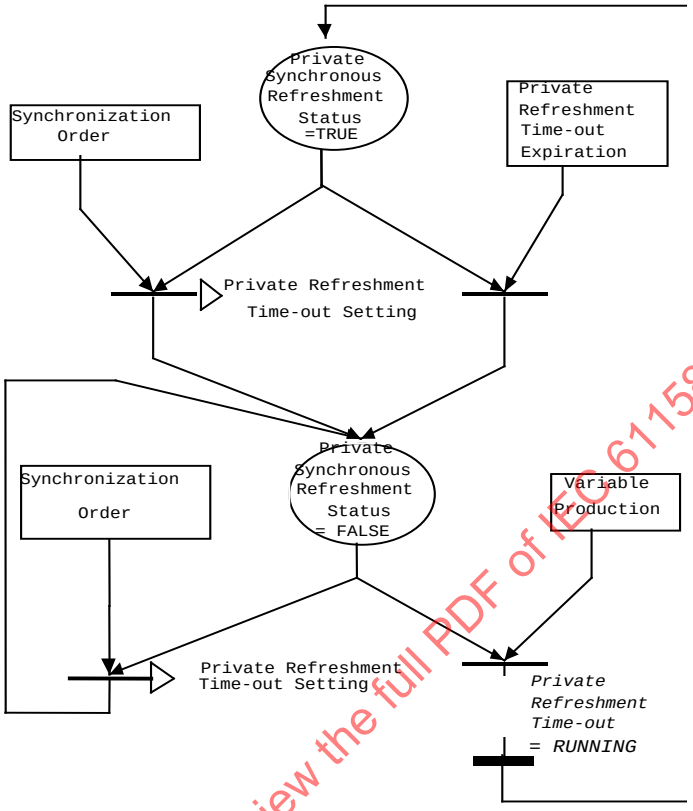


Figure 26 – Synchronous refreshment private mechanism evaluation net

Table 21 – Synchronous refreshment private mechanism events and actions

Initial state:	**Private synchronous refreshment status = FALSE**
	State of the private refreshment time-out = IDLE
Requests:	**Variable Production:**
	Invocation of the services A_WRITELOC, A_WRITEFAR or A_WRITE by the user
	Private Refreshment Time-out Expiration:
	Switch of the private time-out associated with the private synchronous status from the RUNNING state to the IDLE state.
	Synchronisation Order:
	Occurrence of a synchronization order associated with the public and private synchronous refreshment status.
Action:	**Private Refreshment Time-out Setting**
	Switch of the private time-out associated with the private asynchronous status from the *OUT* state to the *SET* state, with the preset value equal to the Production *period* attribute value.

Public mechanism:

The time-out associated with the public mechanism is preset with the current value of the private time-out and set on the occurrence of the resynchronization order.

The public mechanism to elaborate the synchronous refreshment status of a resynchronized variable is represented by the following evaluation net, shown in Figure 27 and Table 22.

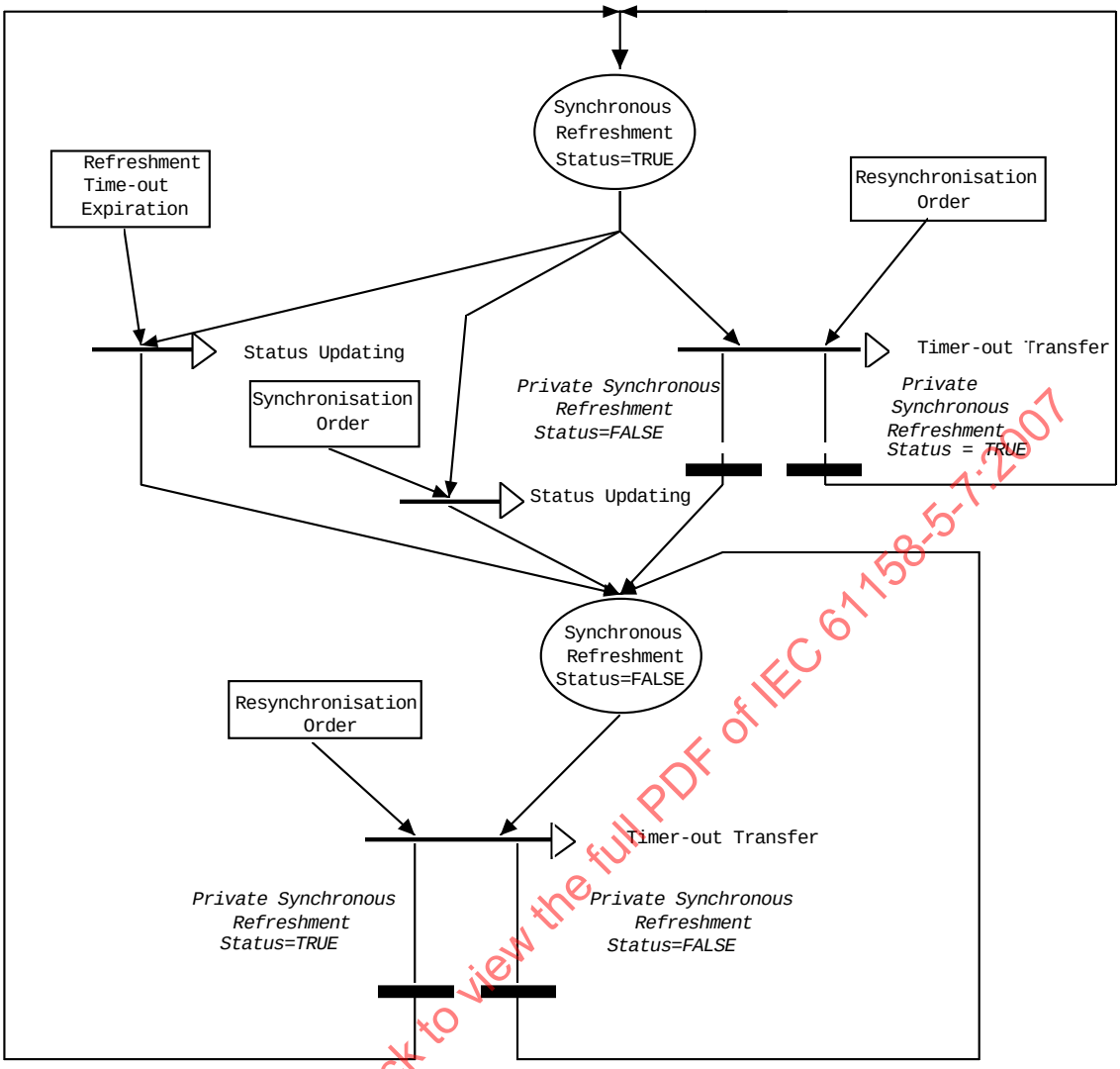


Figure 27 – Synchronous refreshment public mechanism evaluation net

Table 22 – Synchronous refreshment public mechanism events and actions

Initial state: **Synchronous refreshment status = FALSE**

State of the refreshment time-out = IDLE

Requests: **Refreshment Time-out Expiration:**

Switch of the public time-out associated with the synchronous refreshment status from the RUNNING state to the IDLE state.

Resynchronization Order:

Occurrence of the synchronization order associated with the public resynchronisation mechanism.

Synchronization Order:

Occurrence of the synchronisation order associated with the refreshment status and to the private synchronous refreshment state.

Action: **Time-outs Transfer:**

Transfer in the *State* attribute value and *Current* attribute value of the private time-out of the private asynchronous refreshment status, of the attributes State and Preset belonging to the public time-out associated with the refreshment status.

6.2.1.3.6.2.2.6 Mechanism associated with the punctual refreshment

The mechanism which elaborates the punctual refreshment for a resynchronized variable may be described with two evaluation nets one of which describes the private punctual refreshment status associated with the private variable value, and the other the punctual refreshment status associated with the public variable value.

Private mechanism:

The private time-out associated with the punctual refreshment is preset with the production time slot of the variable and set at the occurrence of the synchronization order associated with this status, as shown in Figure 28 and Table 23.

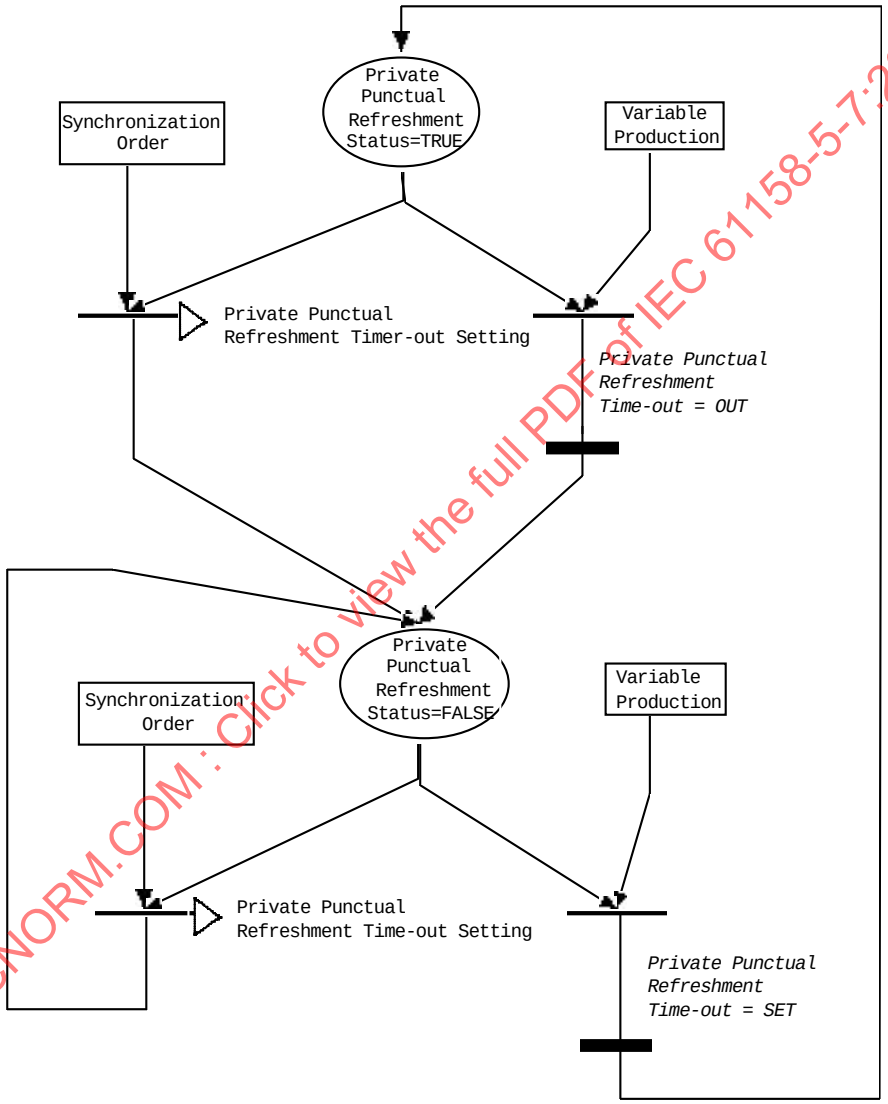


Figure 28 – Punctual refreshment private mechanism evaluation net

Table 23 – Punctual refreshment private mechanism events and actions

Initial state:	**Private punctual refreshment status = FALSE**
	State of the private punctual refreshment time-out = IDLE
Requests	**Variable Production:**
	Invocation of services A_WRITELOC, A_WRITEFAR or A_WRITE by the user
	Synchronization Order:
	Occurrence of the synchronization order associated with the punctual refreshment status and to the private punctual refreshment.
Action:	**Private Punctual Refreshment Time-out Setting**
	Switch of the private time-out associated with the private punctual refreshment status from the IDLE state to the RUNNING state, with the preset value equal to the *Production time slot* as attribue value.

Public mechanism:

The public mechanism which elaborates the punctual refreshment status of a resynchronized variable is represented by the following evaluation net, shown in Figure 29 and Table 24.

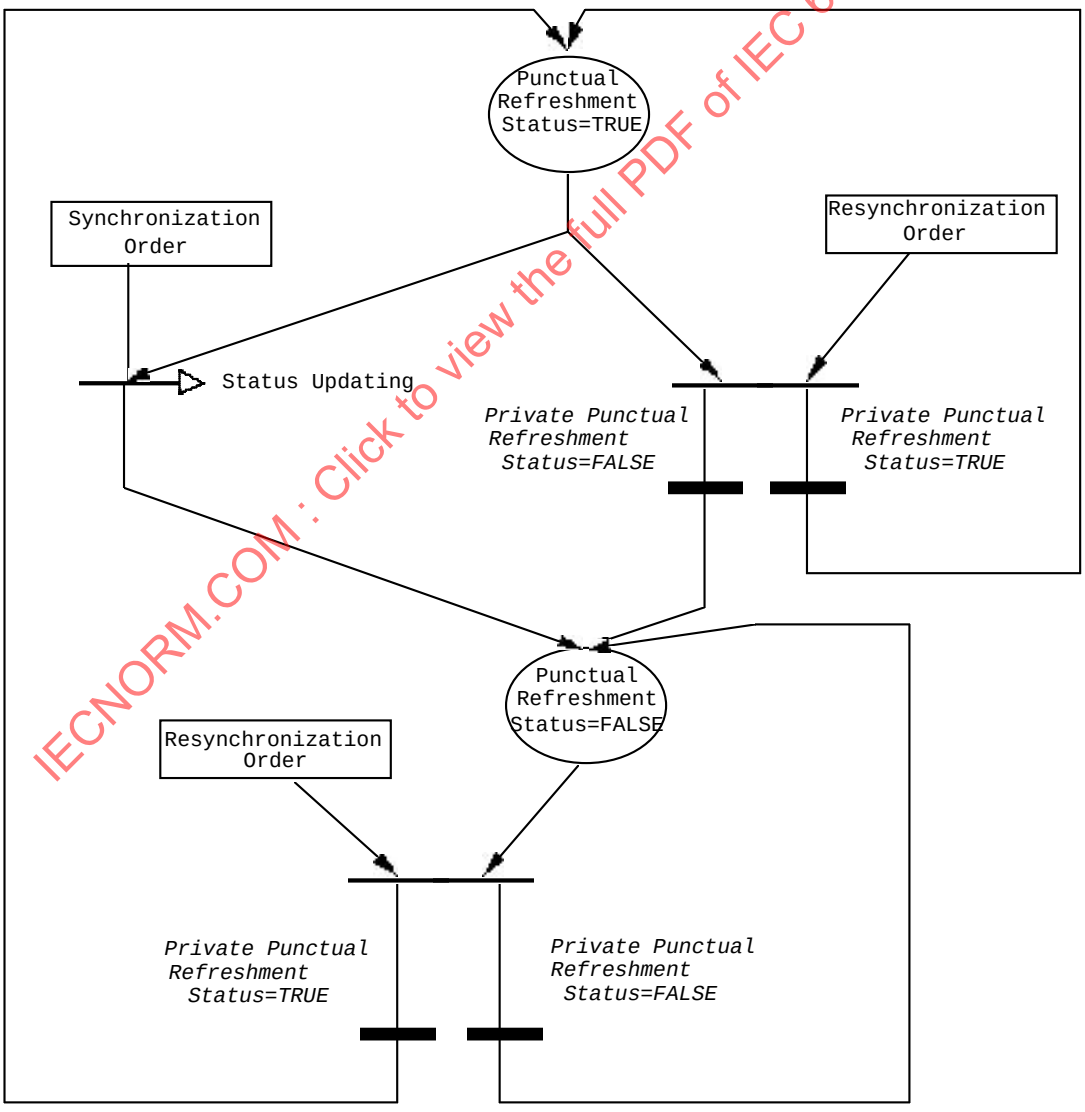


Figure 29 – Punctual refreshment public mechanism evaluation net

Table 24 – Punctual refreshment public mechanism events and actions

Initial state	**Punctual refreshment status = FALSE**
Requests:	**Resynchronization Order**
	Occurrence of the synchronization order associated with the public resynchronization mechanism.
	Synchronization Order:
	Occurrence of synchronization order associated with the punctual refreshment status and to the private punctual refreshment status.

The elaboration of synchronous refreshment status and punctual refreshment status requires specific precautions with regard to the respective choice of the synchronization variables associated with synchronous status and those associated with the resynchronization.

NOTE The specific choice consisting in using the same synchronization variable results in status that will be FALSE whenever the user performs a write.

The choice of two distinct synchronization variables, which transmission on the bus is separated by a time interval smaller than the production periods or the production time slots, may lead to status FALSE if the resynchronization order arrival time precedes the user writing time.

6.2.1.3.6.2.3 Resynchronization of a consumed variable

6.2.1.3.6.2.3.1 Principle

If the network provides a variable value in the public buffer (TPU) with a Tr transmission period, and at the same time an AP consumes this value in the private buffer (TPP), then the transfer of the variable value from the public buffer to the private buffer is performed on the reception of a resynchronization order with a Tc period.

The consumer resynchronization mechanism holds in the private buffer the same value of the consumed variable during the time interval between two occurrences of the same resynchronization order. Figure 30 shows these principles.

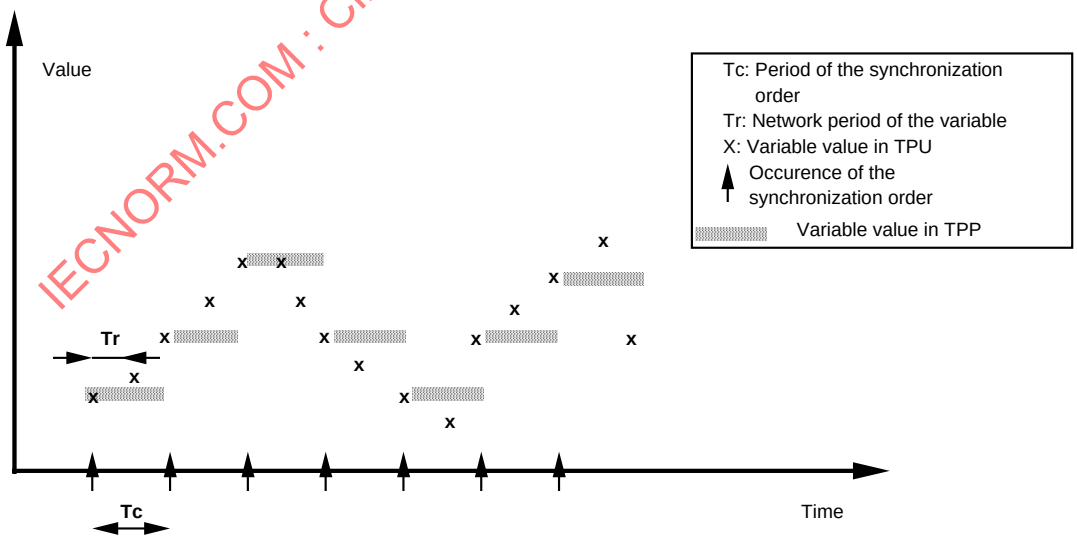


Figure 30 – Principles for the resynchronisation of a consumed variable

NOTE The resynchronization mechanism of a consumed variable is useful only when $T_r << T_c$

The resynchronization service implies the use of extra resources like additional buffers for private values associated with the public values of the resynchronized variables.

With each public buffer of a resynchronized variable a time-out of promptness status elaboration time-out, and punctual promptness elaboration time-out, are optionally associated.

The resynchronization service requires the use of a private time-out associated with the private buffer in order to allow the transmission status to preserve their semantics when the variables are resynchronized.

The elaboration of the transmission validity status for a resynchronized variable at the consumer entity level involves the use of two mechanisms associated respectively with the public and private values of the resynchronized variable.

6.2.1.3.6.2.3.2 Specification of the resynchronization in consumption class

This class describes additional resources used to resynchronize a consumed variable at an application entity level.

The class used to describe a resynchronized variable within consuming entity inherits from attributes of the *Resynchronization in consumption* class.

Class:	RESYNCHRONIZATION IN CONSUMPTION	
Attribute	:	Private value
Constraint	:	Promptness elaborated = TRUE
Attribute	:	Private Promptness status
Attribute	:	*Inherit from* Time-out
Constraint	:	Punctual Promptness elaborated = TRUE
Attribute	:	Private Punctual Promptness status
Attribute	:	*Reference* Synchronization variable

Private value:

The resynchronization principle is based on data which is double buffered. It is necessary to add a private value to the public value of the variable. The private value may now be read asynchronously.

Promptness elaborated:

This attribute, both in the private context and in the public context, conditions the existence of a promptness status associated with a variable.

When this attribute is TRUE a resynchronized variable has the use of a promptness status and a private promptness status.

This conditional attribute is declared in the *Consumed variable* class.

Private promptness status

This private status corresponds to internal information from the consuming entity of the resynchronized variable, from which is built the value of the promptness status.

The characteristic of this status is SYNCHRONOUS or ASYNCHRONOUS according to the global declaration associated with resynchronized variable in consumption which is made in the *Promptness* class.

In the case of a private synchronous status the synchronization variable used by the private mechanism associated with this status is declared globally in the *Promptness* class.

Inherit from Time-out:

This set of resources allows the user to maintain the semantics of the promptness status associated with a resynchronized variable.

In fact, the use of a synchronous or asynchronous promptness status requires the use of an extra time-out to preserve the whole semantics of this status.

Punctual Promptness elaborated

This attribute conditions globally in the private context and in the public context the existence of a punctual promptness status associated with a variable.

When this attribute is TRUE, a resynchronized variable has the use of a punctual promptness status and a private punctual promptness status.

This conditional attribute is declared in the *Consumed variable* class.

Private Punctual Promptness status

This private status corresponds to internal information from the consuming entity of the resynchronized variable, from which is built the value of the punctual promptness status.

In the case of a private punctual status the synchronization variable used by the private mechanism associated with this status is declared globally in *Punctual Promptness* class

***Reference* (Synchronization) variable:**

The reference to synchronization variable indicates the resynchronization order associated with a resynchronized variable.

This resynchronization order conditions the transfer of variable values, transmission statuses and current values of time-outs from the private domain to the public domain. This reference indicates the *A_Name* of the *Consumed Variable Local Image* object.

6.2.1.3.6.2.3.3 Mechanism associated with the consumed value

The resynchronization mechanism in consumption of the local variable value is the transfer from the private buffer to the public buffer, on the resynchronization order.

The evaluation net of this service element which manages the resynchronization actions within a consuming application entity for a resynchronized variable is shown in Figure 31.

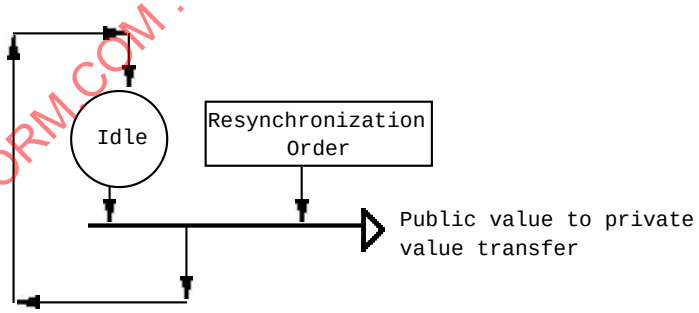


Figure 31 – Resynchronisation mechanism state machine for consumed variable

6.2.1.3.6.2.3.4 Mechanism associated with asynchronous promptness

The mechanism which elaborates the asynchronous promptness status for a resynchronized variable may be described with two evaluation nets one of which describes the private asynchronous promptness and the other the asynchronous promptness status associated with the public value of the variable.

Public mechanism:

The public time-out associated with the promptness is preset with the consumption period of the variable and set at the receiving time of a new public variable value, as shown in Figure 32 and Table 25.

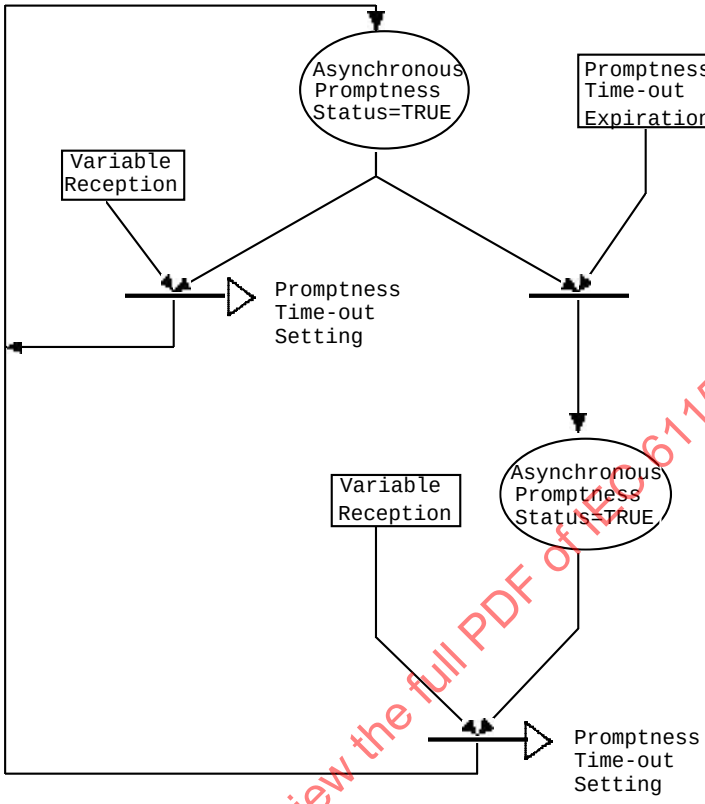


Figure 32 – Asynchronous promptness public mechanism evaluation net

Table 25 – Asynchronous promptness public mechanism events and actions

Initial state:	**Asynchronous promptness status = IDLE**
	State of the promptness time-out = OUT
Requests:	**Variable Reception:**
	Updating of the variable value by the network
	Promptness Time-out Expiration:
	Switch of the public time-out associated with the public asynchronous promptness status from the RUNNING to the IDLE state.
Actions:	**Promptness Time-out Setting:**
	Switch of the public time-out associated with the public asynchronous status from IDLE state to RUNNING state, with the preset value equal to the *Consumption period* attribute value.

Private mechanism:

The private time-out is preset with the current value of the public time-out and set on the resynchronization order associated with this status.

The mechanism to elaborate the asynchronous private promptness status of a resynchronized variable is represented by the following evaluation net, shown in Figure 33 and Table 26.

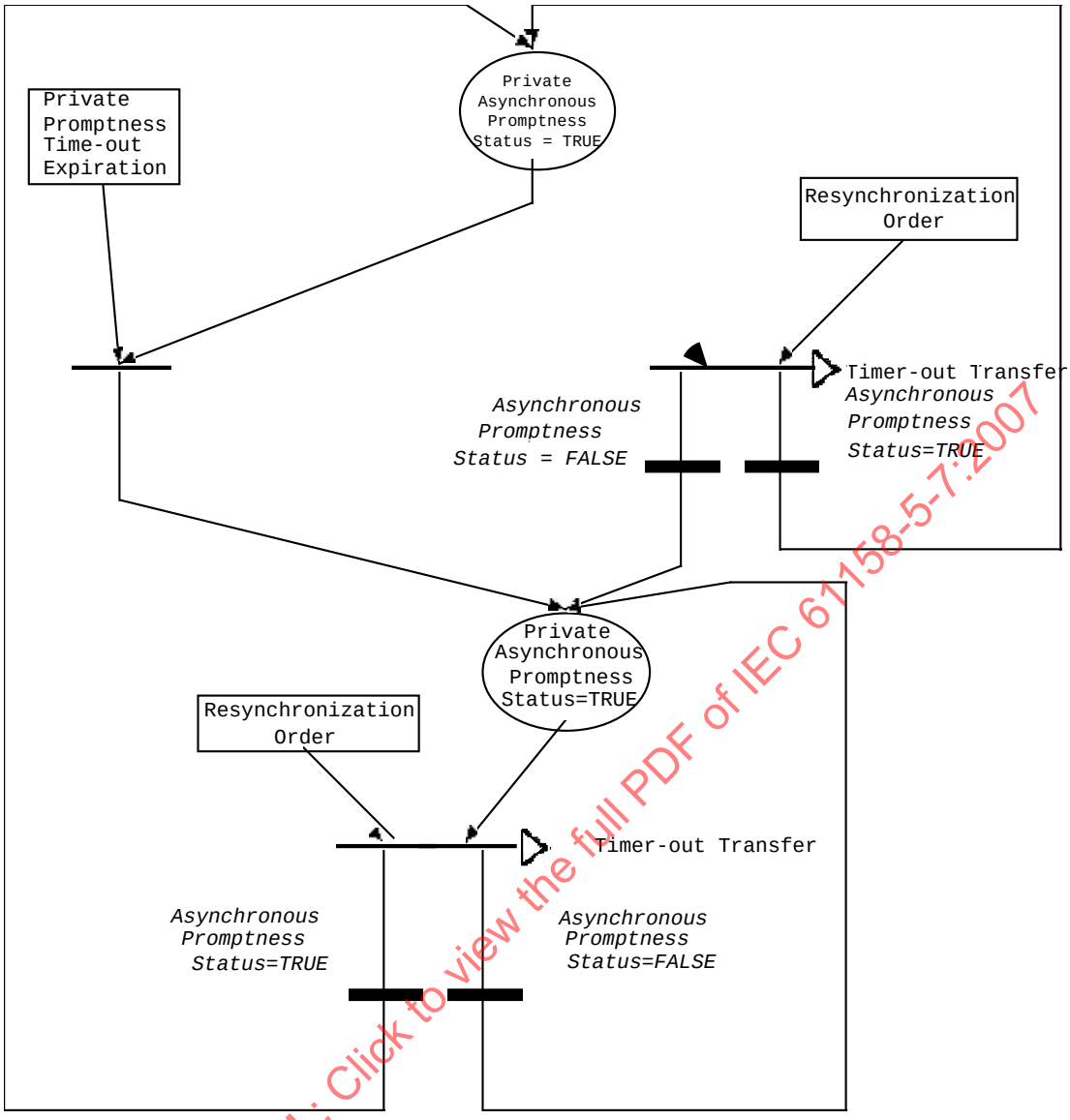


Figure 33 – Asynchronous promptness private mechanism evaluation net

Table 26 – Asynchronous promptness private mechanism events and actions

Initial state:	**Private asynchronous promptness status = FALSE**
	State of the private promptness time-out = IDLE
Requests:	**Private Promptness Time-out Expiration**
	Switch of the private time-out associated with the private asynchronous promptness status from the RUNNING state to the IDLE state.
	Resynchronization Order
	Occurrence of a synchronization order associated with the resynchronization mechanism.
Action:	**Time-outs Transfer**
	Transfer in the State attribute value and Current attribute value of the public time-out of the public asynchronous promptness status, of the attributes State and Preset belonging the private time-out associated with the asynchronous promptness status.

6.2.1.3.6.2.3.5 Mechanism associated with synchronous promptness

The mechanism which elaborates the synchronous promptness status for a resynchronized variable may be described with two evaluation nets one of which describes the private

synchronous promptness status and the other the synchronous promptness status associated with the public value of the variable.

Public mechanism:

The public time-out associated with the promptness is preset with the consumption period of the variable and set at the receiving time of a new synchronization order, as shown in Figure 34 and Table 27.

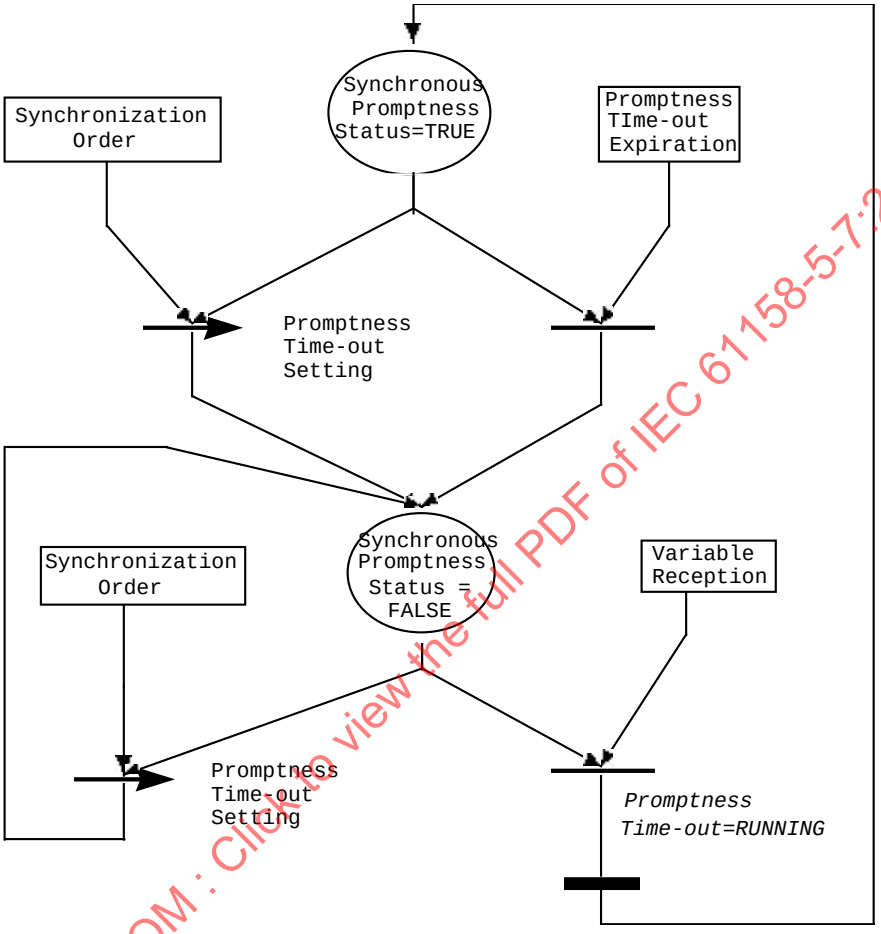


Figure 34 – Synchronous promptness public mechanism evaluation net

Table 27 – Synchronous promptness public mechanism events and actions

Initial state:	**Synchronous promptness status = FALSE**
	State of the promptness time-out = IDLE
Requests:	**Variable Reception**:
	Updating of the variable value by the network
	Synchronization Order:
	Occurrence of a synchronization order associated with the promptness status and to the private synchronous promptness status.
	Promptness Time-out Expiration:
	Switch of the public time-out associated with the synchronous promptness status from the RUNNING state to the IDLE state.
Action:	**Promptness Time-out Setting**:
	Switch of the public time-out associated with the public synchronous status from IDLE state to the RUNNING state with the preset value equal to the Consumption period attribute value.

Private mechanism:

The private time-out is preset with the current value of the public time-out and set on the Occurrence of the resynchronization order.

The private mechanism to elaborate the synchronous promptness status of a resynchronized variable is represented by the following evaluation net, shown in Figure 35 and Table 28.

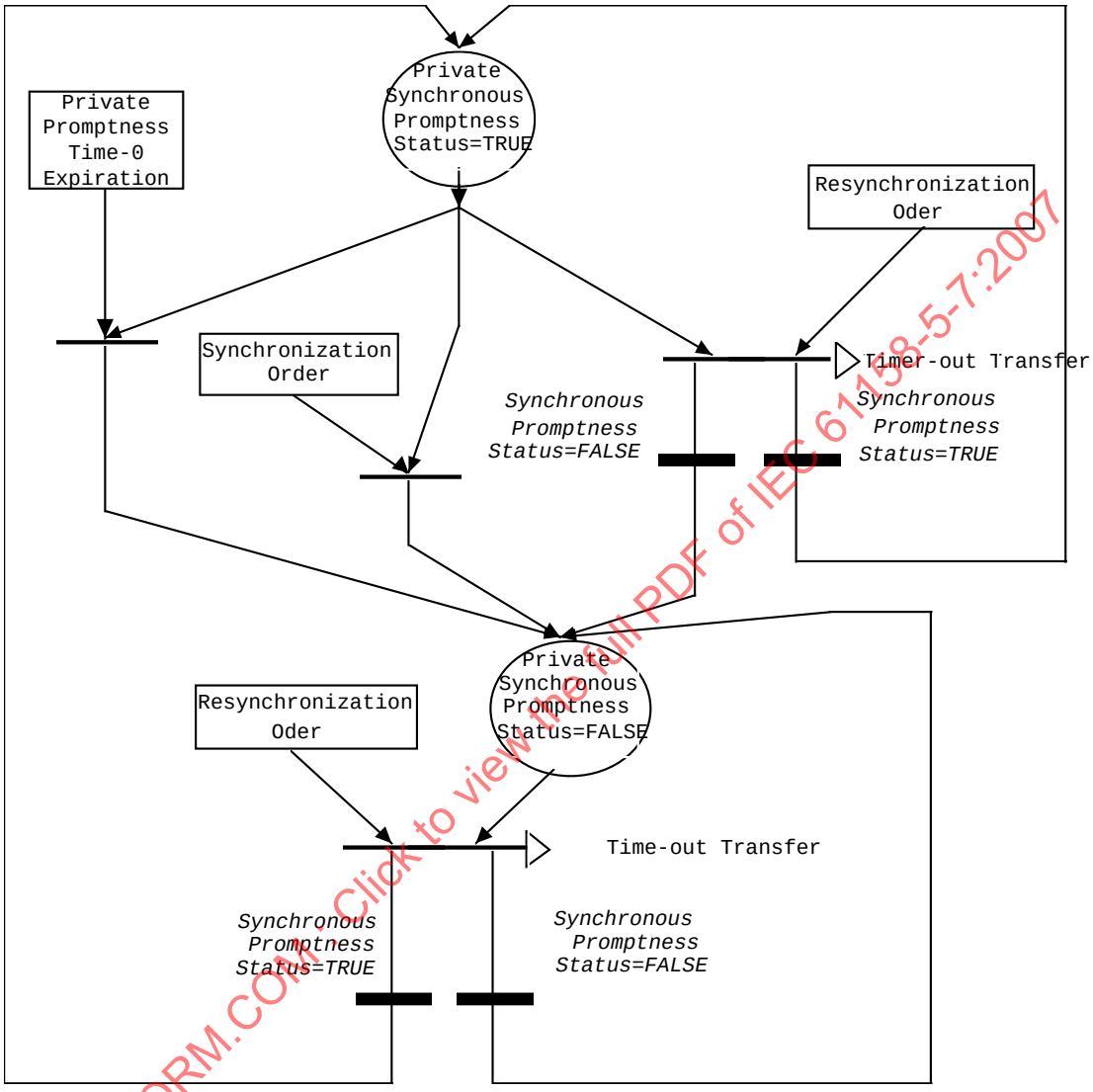


Figure 35 – Synchronous promptness private mechanism evaluation net

Table 28 – Synchronous promptness privatemechanism events and actions

Initial state:	**Private synchronous promptness status = FALSE**
	State of the private promptness time-out = IDLE.
Requests:	**Private Promptness Time-out Expiration**
	Switch of the private time-out associated with the synchronous promptness from the RUNNING state to the IDLE state.
	Resynchronization Order
	Occurrence of a synchronization order associated with the resynchronization mechanism
	Synchronization Order
	Occurrence of a synchronization order associated with the promptness status and with the private synchronous promptness
Action:	**Time-outs Transfer:**
	Assignation of the attributes value State and Current value of the public synchronous promptness status, to the attributes State and Preset of the private timer associated with the synchronous promptness status.

6.2.1.3.6.2.3.6 Mechanism associated with the punctual promptness

The mechanism which elaborates the punctual promptness for a resynchronized variable may be described with two evaluation nets which one described the private punctual promptness status associated with the private value, and the other the punctual promptness status associated with the public variable value.

Public mechanism:

The public time-out associated with the punctual promptness is preset with the consumption time slot of the variable and set at the occurrence of the synchronization order associated with this status, as shown in Figure 36 and Table 29.

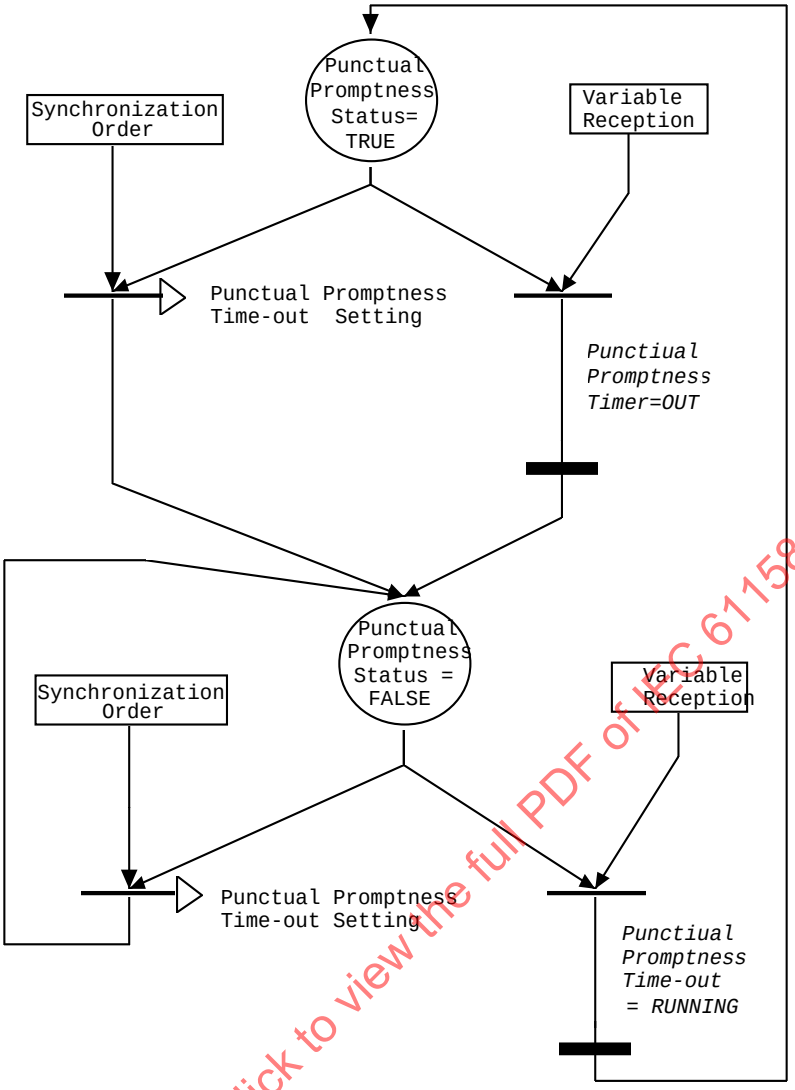


Figure 36 – Punctual promptness public mechanism evaluation net

Table 29 – Punctual promptness public mechanism events and actions

Initial state **Punctual promptness status = FALSE**

State of the promptness time-out = OUT

Requests: **Variable Reception:**

Updating of the variable value by the network.

Synchronization Order:

Occurrence of a synchronization order associated with the punctual promptness status and to the private punctual promptness status.

Action: **Punctual Promptness Time-out Setting**

Switch of the public time-out associated with the public punctual promptness status from the IDLE state to the RUNNING state, with the preset value equal to the *Consumption time slot* attribute value.

Private mechanism:

The private mechanism which elaborates the punctual promptness status of resynchronized variable is represented by the following evaluation net, shown in Figure 37 and Table 30.

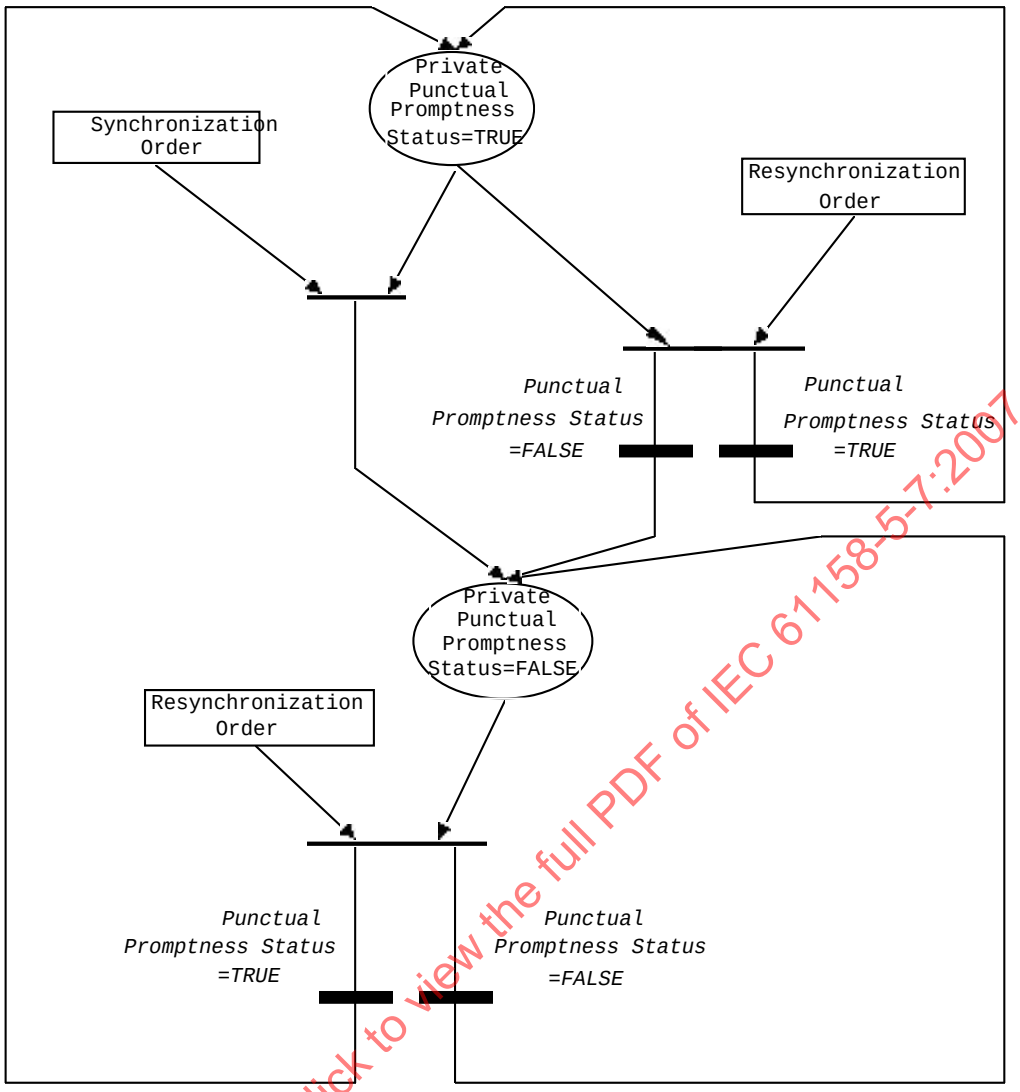


Figure 37 – Punctual promptness private mechanism evaluation net

Table 30 – Punctual promptness privatemechanism events and actions

Initial state	**Private punctual promptness status = FALSE**
Requests	**Resynchronization Order**:
	Occurrence of asynchronization order associated with the resynchronization mechanism.
	Synchronization Order:
	Occurrence of the synchronization order associated with the punctual promptness status and to the private punctual promptness status.

The elaboration of a synchronous promptness and a punctual promptness status requires specific precautions for respective choices of the synchronization variables associated with synchronous status and those associated with the resynchronization.

NOTE The specific choice consisting in using the same synchronization variable results in status that will be FALSE whatever the time of the network updating.

The choice of two distinct synchronization variables, for which transmission on the bus is separated by a time interval smaller than the consumption period or the consumption time slot, may lead to status FALSE if the time of the arrival of the resynchronization order precedes the updating time by the network.

6.2.1.4 Variable lists access sub-ASE

6.2.1.4.1 Concepts

6.2.1.4.1.1 List definition

The application layer supports definition and handling of variable lists.

A list is an ordered set of variables of NORMAL class. A variable list cannot include variables of SYNCHRONIZATION or DESCRIPTION classes.

A variable list is globally defined and instantiated at the application entity level. It is composed exclusively of consumed variables.

Several lists can be defined within a single system, but they should all be separated two by two:

$$\forall_{i,j} \ \mathbf{list}_i \cap \mathbf{list}_j = \varnothing$$

However ; several instances of a same list can be defined, within a system, because several APs can be consumers of the same list.

A variable list defined in this way can only be handled by an appropriate read service permitting global access to the whole list. However, all the variables belonging to a list can also be handled individually by the variable read services.

NOTE There is no support for the concept of a recursive list at the level of the application services provided to the user. (Therefore, there is no list of list.)

The variables declared by the user as belonging to a variable list cannot be grouped together into an structure, because it is necessary for them to be optimisation provided with individual validity information.

Most of the variable lists implemented in the MPS environment are composed of periodic variables having the same period, but lists of non periodic variables are also envisaged.

6.2.1.4.1.2 Status associated with a list

At the time of a list read, the user can get additional information relative to the variable consistency of this list. These status are elaborated or not according to the configuration associated to the list. They inform the user about the integrity of all the variables composing the list.

These status are returned by the list access service.

In the MPS environment the various status associated with the variable lists inform the user regarding:

- the TRANSMISSION CONSISTENCY of the list
- the PRODUCTION CONSISTENCY of the list
- the PUNCTUAL TRANSMISSION CONSISTENCY of the list
- the PUNCTUAL PRODUCTION CONSISTENCY of the list
- the SPATIAL CONSISTENCY of the list

All these status associated with a list by a user are elaborated using:

- information local to a consuming AE, and relative to the transmission validity of the variables of the list,

- information elaborated by the remote producers of the variables of the list, and relative to their production validity.

The information relative to the transmission validity, involved in the elaboration of the consistency status for each of the list variables is

- the asynchronous or synchronous promptness,
- the punctual promptness

The information relative to the production validity involved in the elaboration of the consistency status for each of the list variables is

- the asynchronous or synchronous refreshment,
- the punctual refreshment.

NOTE the production validity information elaborated within the producing entities of the list variables is carried on the network with the values of the variables.

6.2.1.4.1.3 Control elements and reliability

The reliability of transmission of the variables of a list can be enhanced by using a recovery mechanism ; This mechanism performs a retransmission request for a number of times limited to a maximum defined for each of the instances of this list.

The evaluation of the spatial consistency status is locally derived from a purely logical function based on the consistency variable values.

The concept of spatial consistency is based on:

- The existence of a consistency variable produced by every subscriber consuming a local image of the list, as well as on the existence of an interchange mechanism of these consistency variables between the different subscribers.
- The existence of a local mechanism of value management of the produced consistency variable.
- The definition of a certain sequential ordering action for the interchange of the variables of the list, and the consistency variables.

Sequential ordering of the variable interchange on network

In order to elaborate a spatial consistency status on a variable list, it is necessary to define a cycle corresponding to a transmission of all the list variables in order to be able to set up a temporal reference for the consistency variable management.

This cycle is delimited by a Vs synchronization order whose occurrence defines the beginning of a new variable interchange cycle. All the list variables of the list need to be received and all the consistency variables need to be interchanged between two consecutive occurrences of the Vs synchronization order. This is shown in Figure 38.

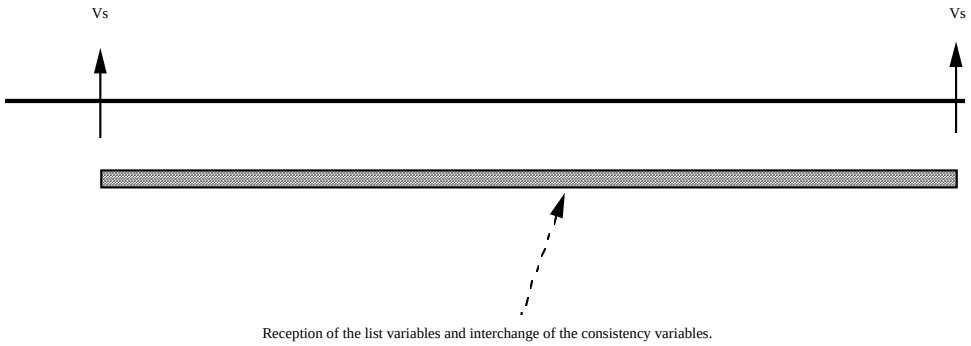


Figure 38 – Spatial consistency list variables interchange mechanism

Moreover, a condition of using the lists requires that the consistency variable interchange be performed after interchange of all the list variables, and within the current cycle, as shown in Figure 39.

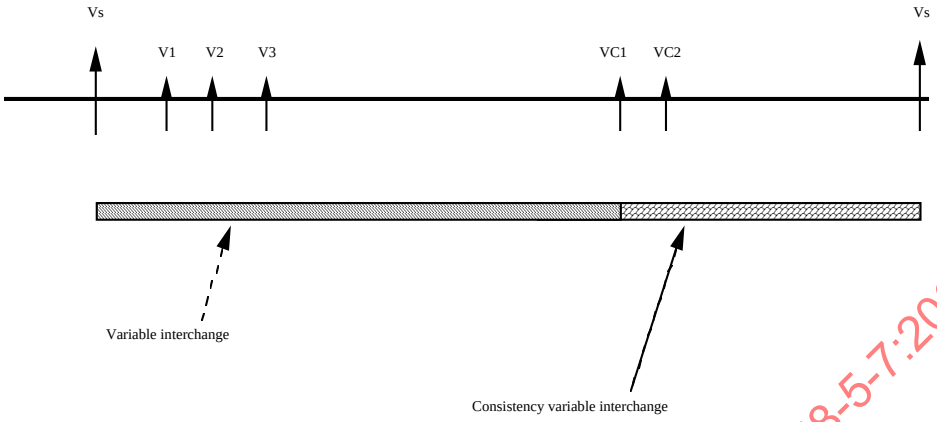


Figure 39 – Spatial consistency – consistency variable interchange mechanism

When the spatial consistency status elaboration mechanism is involved with a recovery mechanism, as shown in Figure 40, the eventual recovery on variables concerned by transmission errors need to be performed after the theoretical reception of all the list variables, and before the consistency variable interchange. This makes the recovery mechanism and the spatial consistency mechanism meaningful.

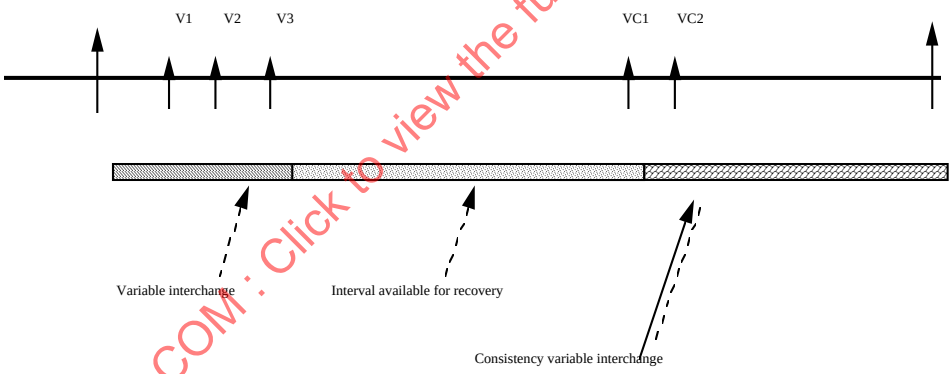


Figure 40 – Spatial consistency – list recovery mechanism

Finally, the spatial consistency status locally elaborated inside an AE and made available to the user is significant only when all the consistency variables have been interchanged. This leads to defining a validity interval of the spatial consistency status, as shown in Figure 41.

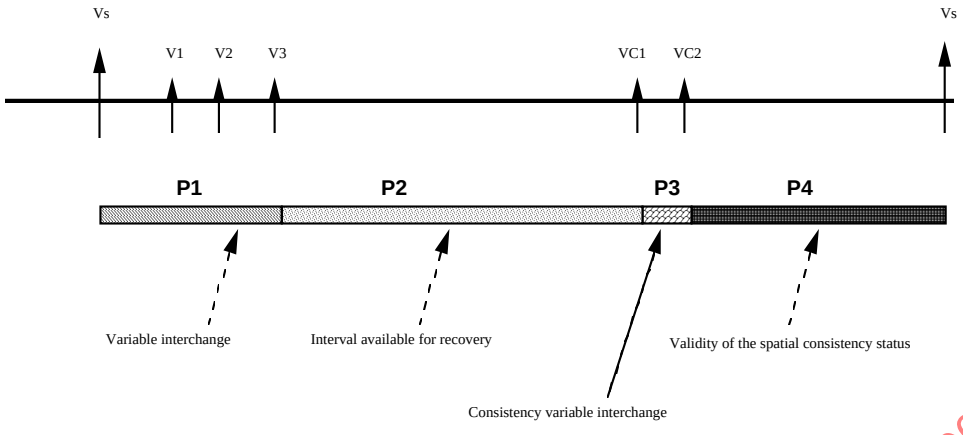


Figure 41 – Spatial consistency – validity of the spatial consistency status

6.2.1.4.1.4 Object model of a variable list

A variable list in the MPS environment is modeled by using the *Variable List Local Image* object which contains all the attributes required by an application entity consuming this variable list.

This object comes from the instantiation of the *Identified Variable List* class which contains all the attributes characterizing a variable list.

The common attributes of a variable list in the different entities, described in the *Identified variable List* class come from the instantiation of a *Variable List* generic class which describes the whole of the generic attributes of a variable list.

Moreover, it is pointed out that the *Variable List* generic class refers to the *Variable* generic class in order to describe the variables composing this variable list. This is all shown in Figure 42.

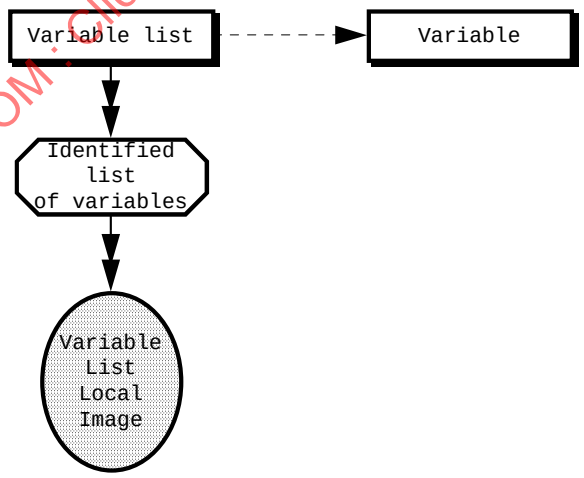


Figure 42 – Object model of a variable list

6.2.1.4.2 Variable list class specification

6.2.1.4.2.1 *Variable List* generic class specification

6.2.1.4.2.1.1 Variable list generic class model

FAL ASE: **MPS ASE**

CLASS: Variable list
CLASS ID: not used
PARENT CLASS: not used
Generic Class: VARIABLE LIST
CLASS ATTRIBUTES:
Key-Attribute : A_Name
Attribute : List of Reference Consumed Variable
Attribute : Spatial consistency required [TRUE,FALSE]
Attribute : Recovery required [TRUE,FALSE]
Constraint : Spatial consistency required = TRUE ***OR*** Recovery required = TRUE
Attribute : *Reference* (Synchronization) Variable
Constraint : Spatial consistency required = TRUE
Attribute : Inconsistency detection [TRANSITORY,PERMANENT,UNDEFINED]
Attribute : *List of*
Attribute: *Reference* (Consistency) Variable
Constraint : Recovery required = TRUE
Attribute : Recovery period
Instance Attributes:
Attribute : Transmission consistency required [TRUE,FALSE]
Constraint : Transmission consistency required = TRUE
Attribute : Transmission consistency status [TRUE,FALSE]
Attribute : Production consistency required [TRUE,FALSE]
Constraint : Production consistency required = TRUE
Attribute : Production consistency status [TRUE,FALSE]
Attribute : Punctual transmission consistency required [TRUE,FALSE]
Constraint : Punctual transmission consistency required = TRUE
Attribute : Punctual transmission consistency status [TRUE,FALSE]
Attribute : Punctual production consistency required [TRUE,FALSE]
Constraint : Punctual production consistency required = TRUE
Attribute : Punctual production consistency status [TRUE,FALSE]
Constraint : Spatial consistency required = TRUE
Attribute : *Reference* (Consistency) Produced Variable
Attribute : Spatial consistency status [TRUE,FALSE]
Constraint : Recovery required = TRUE
Attribute : Recovery list size
Attribute : Recovery list contents
Attribute : Recovery Nature [FRAGMENTED,INDIVISIBLE]
Attribute : Recovery List Identifier
Attribute : *Inherit from* Time-out

6.2.1.4.2.1.2 Attributes

A_Name:

This attribute supports unique identification at the interface between the application layer and the user. The A_Name of a variable list belongs to the MPS addressing space

Moreover, the maximum length of an A_Name of a variable list should not exceed 14 characters.

List Of Reference Consumed Variable:

This attribute allows a variable list to refer to the constituent variablesThis set of references between an *Identified Variable List* class and *Consumed variable* classes is identical for each local image

Spatial consistency required:

The spatial consistency status associateda variable list is optional.

A "TRUE" value for this boolean attribute indicates that a spatial consistency status should be elaborated at the list level.

Recovery required:

The implementation of the recovery mechanism is optional.

A "TRUE" value for this boolean attribute indicates that a recovery mechanism should be implemented for the variable list.

***Reference* (synchronization) Variable:**

The variable lists refer to a synchronization order when a spatial consistency status is elaborated or when the recovery mechanism is implemented. This synchronization order is used on one hand by the service elements of various subscribers which elaborate the values of the consistency variables which are involved in the spatial consistency status elaboration, and on the other hand by the service elements that manage locally the recovery mechanism at the level of every application entity.

Inconsistency detection:

This attribute specifies the type of the consistency variables that are involved in the elaboration of the spatial consistency status.

The inconsistency detection is TRANSITORY when the consistency variable value type takes into account the transmission faults during the last interchange.

The inconsistency detection is PERMANENT when the consistency variable value type takes into account the transmission faults of the last 2^{16} previous interchanges.

The inconsistency detection is UNDEFINED when the consistency variable value type is freely defined by the user for an elaboration of a spatial consistency status in his user application.

***Reference* (consistency) Variable:**

This set of references to consistency Variables locally supports the transmission status of the variables of the list within the other application entities consuming this list.

A consistency variable characterizes a variable list at the level of a particular application entity. It supplies information regarding the transmission validity of each variable of this list from the transmission of the last synchronization variable associated to the list.

Recovery period:

This attribute represents the maximum duration allowed between two reinitialisations of the contents of the recovery list.

This period is expressed in milliseconds.

Transmission consistency required:

Transmission consistency associated with a variable list is optional.

When it has the value 'TRUE" this attribute indicates that a transmission consistency status should be elaborated for this list.

Transmission consistency status:

The transmission consistency status provided to the user is of type boolean.

This status is elaborated at the level of every application entity receiving variables of a list defined with a transmission consistency status.

Production consistency required:

Production consistency associated with a variable list is optional.

When it has the value TRUE this attribute indicates that a production consistency status should be elaborated for this list.

Production consistency status:

Production consistency is a status provided to the user.

This status of type Boolean is elaborated at the level of every application entity receiving the variables of a list with a production consistency status.

Punctual transmission consistency required:

Punctual transmission consistency associated with a variable list is optional.

When it has the value TRUE this attribute indicates that a punctual transmission consistency status should be elaborated for this list.

Punctual transmission consistency status:

Punctual transmission consistency is a status provided to the user.

This status of type Boolean is elaborated at the level of every application entity receiving the variables of a list defined with a punctual transmission consistency status.

Punctual production consistency required:

The punctual production consistency associated with a variable list is optional.

This attribute of boolean type indicates that a punctual production consistency status should be elaborated for this list when it has the value TRUE.

Punctual production consistency status:

Punctual production consistency is a status provided to the user.

This status of type boolean is elaborated at the level of every application entity receiving the variables of a list defined with a punctual production consistency status.

Spatial consistency status:

Spatial consistency is a status provided to the user.

This status of type boolean is elaborated at the level of every application entity receiving variables of a list defined with a spatial consistency status.

***Reference* (consistency) Produced Variable:**

This reference designates the consistency variable locally produced and transmitted to the other application entities consuming the list in order to inform them about the transmission operation of the variables locally within the entity producing the consistency variable.

Recovery list size:

This attribute characterizes the maximum number of elements of the variable list for which a retransmission will be requested during the same cycle. The size of the recovery list is always smaller or equal to the number of elements which compose the variable list.

Recovery list contents:

This attribute characterizes a set of identifiers associated with elements of a variable list that were not successfully received after the last reception on the network of the synchronization variable associated to this list.

These contents are not accessible to the user, but they are directly interpretable by a data link service element to recover the failed transmissions during the last interchange.

The contents of a recovery list is used to perform a transmission recovery request of the maximum number of elements defined by the recovery list size.

The encoded value of the recovery list is empty in case of non recovery, i. e. when all the variables of the variable list have been correctly transmitted during the last interchange.

Recovery list identifier:

This attribute specifies the identifier with which the contents of the recovery list are associated. This identifier references in a unique way this recovery list associated to a local image of a variable list within a MPS application entity.

Inherit from time-out:

The management of the recovery list contents takes into account the event associated with the time-out expiration which indicates a time which is too long without initialization of the recovery list, as well as an event associated with the occurrence of the synchronization order which initializes the recovery list contents. This time-out is set upon occurrence of the recovery list synchronization and reset automatically when it is over.

Recovery nature:

This attribute defines the quality of service associated with the recovery_mechanism on a variable list. When this attribute has value INDIVISIBLE, the recovery is performed immediately after transmission of the recovery list contents on the network. When this attribute has the value FRAGMENTED the recovery is delayed in a time slot reserved for aperiodic traffic ; moreover, it should be performed only according to the network load.

6.2.1.4.2.2 *Identified Variable List* class specification

The *Identified Variable List* class comes from the instantiation of the *Variable List* generic class

6.2.1.4.2.3 *Variable List Local Image* object specification

The *Variable List Local Image* object comes from the instantiation of the *Identified Variable List* class

6.2.1.4.3 Services

6.2.1.4.3.1 List read service

6.2.1.4.3.1.1 Function

This function allows the user to perform a local read of all the values of the variables composing a list locally declared as consumed.

In addition to the list variable values, this service optionally provides various consistency status associated with this variable list.

6.2.1.4.3.1.2 Service primitives

The read of a variable list is performed using the A_READLIST service primitives, shown in Table 31:

A_READLIST.Rq (Arguments): List read request

A_READLIST.Cnf (Results): Return of the read value.

Table 31 – A_Readlist service parameters

Parameter name	Req	Cnf
Argument		
List specification	M	M (=)
Result (+)		S
Value		M
Transmission consistency status		C
Production consistency status		C
Punctual consistency status		C
Punctual production consistency status		C
Spatial consistency status		S
Result (-)		S
Type of error		M

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. See 1.2.

Argument

List specification:

This argument corresponds to the A_Name attribute of the *Variable List Local Image* object which is to be accessed.

Result (+)

Value

List of the variable values belonging to the list which is read.

Transmission consistency status

Optional status describing the transmission consistency of all the variables of the list.

Production consistency status

Optional status describing the production consistency of all the variables of the list.

Punctual transmission consistency status

Optional status describing the punctual transmission consistency of all the variables of the list.

Punctual production consistency status

Optional status describing the punctual production consistency of all the variables of the list.

Spatial consistency status

Optional status describing the spatial consistency of all the variables of the list.

Result (-)

Error type

Description of the various error types returned after a non successful read request:

Error_1: One of the buffers, where is located the value of a variable of the list is not accessible.

Error_2: The list specification does not correspond, within the addressing space, with a list identification declared as consumed in the AP in consideration.

Error_3: The list is declared locally as invalid.

6.2.1.4.3.1.3 Service procedure

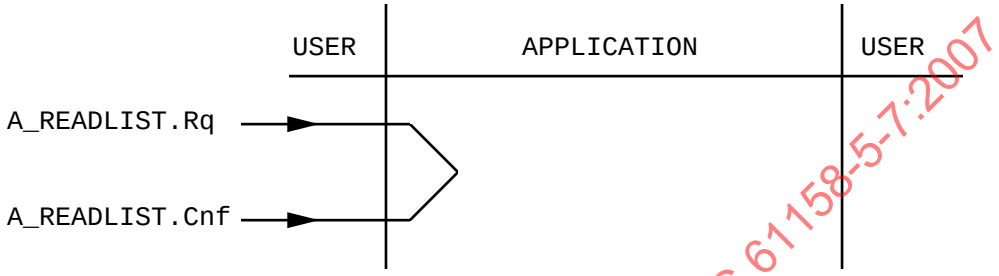


Figure 43 – A_Readlist service procedure

This service, shown in Figure 43, provides all the values locally available of the variables of a list at a given time.

The various elaborated status describe the various consistency levels applied to the variables of the list.

Read requests of a list are sequentially processed ; a new request can be initiated only after confirmation of the previous one.

6.2.1.4.4 Variable list consistency and recovery mechanisms

6.2.1.4.4.1 Transmission consistency

6.2.1.4.4.1.1 Description

Information relative to the transmission consistency of a variable list can be elaborated within a consuming entity, and provided to the user.

This consistency information is provided with two possible semantics:

- the asynchronous transmission consistency informs the consuming user about the respect, for all the variables of this list, of their respective consumption period by the network,
- the synchronous transmission consistency informs the consuming user about the respect, for all the variables of this list, of an availability within their respective consumption period initialized upon occurrence of a single synchronization order associated with the list.

Moreover, these synchronous or asynchronous promptnesses are elaborated with a mandatory value of the *Consumption Period* attribute which is identical for all the variables of this list.

In the case of an asynchronous transmission consistency, all the variables composing this list should elaborate an asynchronous promptness status.

In the case of a synchronous transmission consistency, all the variables composing this list should elaborate a synchronous promptness status.

6.2.1.4.4.1.2 Composition rule

The transmission consistency status is the logical product of the promptness status of the different variables of the list. According to the synchronous or asynchronous characteristics of the promptness status associated with the variables of the list, the resulting transmission consistency status should represents the transmission consistency which is respectively synchronous or asynchronous.

6.2.1.4.4.2 Production consistency

6.2.1.4.4.2.1 Description

Information relative to the production consistency of a variable list can be elaborated at the level of a consuming entity and made available to the user.

These consistency information are provided with two possible semantics:

- the asynchronous production consistency informs the consuming user about the respect, by all the users producing the variables of this list, of their respective production period,
- the synchronous production consistency informs the consuming user about the respect, by all the users producing the variables of this list, of a refreshment within their respective production period initialized by the producing entity upon occurrence of a synchronization order associated with the variable list.

In the case of an asynchronous production consistency, all the variables composing the list should elaborate an asynchronous refreshment status, whereas in the case of a synchronous production consistency, all the variables composing the list should elaborate a synchronous refreshment status. Moreover, these asynchronous and synchronous refreshment status are elaborated with a mandatory value of the *Production Period* attribute which is identical for all the variables of the list.

6.2.1.4.4.2.2 Composition rule

The production consistency status is the logical product of the refreshment status of the various variables of the list.

According to the synchronous or asynchronous characteristics of the refreshment status elaborated by the entities producing the variables of the list, the status should represent the production consistency which is respectively synchronous or asynchronous.

6.2.1.4.4.3 Punctual transmission consistency

6.2.1.4.4.3.1 Description

Information relative to the punctual transmission consistency of a variable list can be elaborated within a consuming entity and provided to the user.

The punctual transmission consistency status informs the consuming user about the respect of an availability by the network of all the variables of this list within their respective consumption time slot.

These respective consumption time slots are initialized by the consuming entity upon occurrence of the synchronization order associated to the list.

All the variables composing the variable list elaborating a punctual transmission consistency status should elaborate a punctual promptness status with the same value of the *consumption time slot* attribute.

6.2.1.4.4.3.2 Composition rule

The punctual transmission consistency status is the logical product of the punctual promptnesss status of all the variables composing this list.

6.2.1.4.4 Punctual production consistency

6.2.1.4.4.1 Description

Information relative to the punctual production consistency of a variable list can be elaborated within a consuming entity and provided to the user.

The punctual production consistency status informs the user consuming the variable list about the respect, by the all the users producing the variables of this list, of a production within their respective time slot.

All the production time slots of the variables of the list are initialized by the producing entities upon occurrence of the synchronization order associated with the list. The variables composing the variable list elaborating a punctual refreshment consistency status should elaborate a punctual promptness status with the same value of the *production time slot* attribute.

6.2.1.4.4.2 Composition rule

The punctual production consistency status is the logical product of the punctual refreshment status of all the variables composing the list.

6.2.1.4.5 Spatial consistency

6.2.1.4.5.1 Description

A status relative to the spatial consistency of a variable list can be elaborated at the level of a consuming entity and made available to the user.

This spatial consistency status is a piece of information from which the user will be able to build a consistency status whose semantics are adapted to the needs of his application ; (For example, the semantics of the user consistency status can be the equality of the values of all the multiple copies of the list variables).

The elaboration mechanism of a spatial consistency status relies on the broadcast by all the entities consuming the same list declared with a spatial consistency characteristics of a consistency variable. This consistency variable reflects locally the transmission validity of each of the variables of the list.

A consistency variable is elaborated by each of the entities consuming a single variable list and provided to the other entities consuming this list.

The elaboration of this spatial consistency status requires that each of the AE consuming a single variable list is provided with a reception report of the list variables of the other consuming AEs. To provide reception report, additional interchanges allow every AE to broadcast to the others its consistency variable relative to this same list.

6.2.1.4.5.2 Composition rule

At the level of a entity consuming a variable list, the processing associated with the spatial consistency relies on the values of consistency variables associated with this list. the spatial consistency status is elaborated from the comparison of the values of the consistency variables associated with a list.

The spatial consistency status value is

TRUE: when there is equality between all the values of the consistency variables associated with this list.

FALSE: in all the other cases.

6.2.1.4.4.5.3 Consistency variable specification

When a spatial consistency status is associated with a variable list, it necessary to elaborate a consistency variable at the level of every application entity consuming this variable list.

This consistency variable is provided to the other entities consuming this variable list.

A consistency variable in the MPS environment is modelled in the same way as for the variables by using the following objects:

– a (consistency) *Produced Variable Local Image* object which contains all the attributes required for a producing application entity,
– one or several (consistency) *Consumed Variable Local Image* objects which contain all the attributes required for consuming application entities.

These objects correspond to particular instantiation of the classes describing the variables for which the *Class* attribute of the object has the value 'SYNCHRONIZATION'.

A consistency variable is different from the other variable classes by a reference to the associated variable list, in order that the service element producing this variable can elaborate the corresponding value.

6.2.1.4.4.6 Predetermined values used by the list consistency mechanisms

6.2.1.4.4.6.1 Predetermined attribute value of the *Identified Variable* class

An *Identified Variable* class corresponding to a consistency variable is characterized by predetermined values for the following attributes:

Class:

This attribute has the value: SYNCHRONIZATION

Consistency Variable:

This attribute has the value: TRUE

6.2.1.4.4.6.2 Predetermined attribute value of the *Produced Variable Local Image* class

A *Produced Variable Local Image* object corresponding to a consistency variable is characterized by predetermined values for the following attributes:

Resynchronized Variable:

This attribute has the value: FALSE.

All the other non quoted attributes are not given predetermined values.

6.2.1.4.4.6.3 Predetermined attribute value of the *Consumed Variable Local Image* object

A *Consumed Variable Local Image* object corresponding to a consistency variable is characterized by predetermined values for the following attributes:

Resynchronized Variable:

This attribute has the value: FALSE.

All the other non quoted attributes are not given predetermined values.

6.2.1.4.4.6.4 Consistency variable type specification

6.2.1.4.4.6.4.1 Consistency variable functionality

The consistency variable values are assigned with a type issued from the instantiation of the *Type Constructor* class.

When the *Inconsistency Detection* attribute has the value 'TRANSITORY', the consistency variable type should be 'K_SCONS'.

When the *Inconsistency Detection* attribute has the value 'FRAGMENTED', the consistency variable type should be 'K_LCONS'.

When the *Inconsistency Detection* attribute has the value 'UNDEFINED', the consistency variable type can be described by any instance of the *Type Constructor* class.

6.2.1.4.4.6.4.2 Predetermined attribute value of the *Type Constructor* class

Instances of the *Type Constructor* class associated with a consistency variable to detect a TRANSITORY spatial inconsistency:

A_Name: K_SCONS
Construction : STRUCTURE
Named field : FALSE
Reference : K_UNS_16 -- Interchange Nr.
Reference : K_BOOLAC64 -- Transmission status
A_Name : K_UNS_16
Construction : SIMPLE
Class of type UNSIGNED
Size : 16
A_Name : K_BOOLAC*nn*
Construction : ARRAY
Compacted : TRUE
Dimension : *nn* (list variable number)
Reference : K_BOOL
A_Name : K_BOOL
Construction : SIMPLE
Class of type BOOLEAN

Instances of the *Type Constructor* class associated with a consistency variable to detect a PERMANENT spatial inconsistency:

A_Name: K_LCONS
Construction : STRUCTURE
Named field : FALSE
Reference : K_UNS_16 -- Interchange Nr.
Reference : K_UNS_16AC*nn* -- Transmission_date
A_Name : K_UNS_16AC*nn*
Construction : ARRAY
Compacted : TRUE
Dimension : *nn* (list variable number)

Reference : K_UNS_16

6.2.1.4.4.6.4.3 Field semantics

The semantics of the consistency variable value fields when the *Inconsistency Detection* attribute does not have the value UNDEFINED is as follows:

Structure fields associated with the 'K_SCONS' type:
Interchange number:

The value of the variables of the list is characterized by the interchange number during which they have been updated by the network. This interchange number is a global data of the MPS environment produced by a specific application entity and broadcast to the entities elaborating spatial consistency information by using the synchronization variable associated with the list.

This interchange number is used to check that the processed consistency variable values dates are identical during the spatial consistency processing.

Transmission_status:

The transmission status of the variable list is an array whose elements represent respectively the transmission status for each of the list variables.

The variable transmission status at the level of a consuming application entity indicates whether this variable has been received since the last occurrence of the synchronization variable associated to the list.

Structure fields associated with the 'K_LCONS' type:

Interchange number:

This field is provided with the same semantics as that with the same name in the 'K_SCONS' type.

Transmission_date:

The transmission date of the variable list is an array whose elements represent respectively the number of the last interchange of each of the list variables.

The number of the last interchange of a variable at the level of a consuming application entity corresponds to that taken into account during the last reception of this variable.

This interchange number, taken into account for a variable during its reception, corresponds to the synchronization variable value of the list which this variable belongs to.

6.2.1.4.4.7 Elaboration mechanism specification of the consistency variable values

This mechanism is involved only when the *Inconsistency detection* attribute associated with the variable list has a value other than 'UNDEFINED'.

This service element elaborates the predetermined type value involved in the spatial consistency status elaboration of a variable list, as shown in Figure 44.

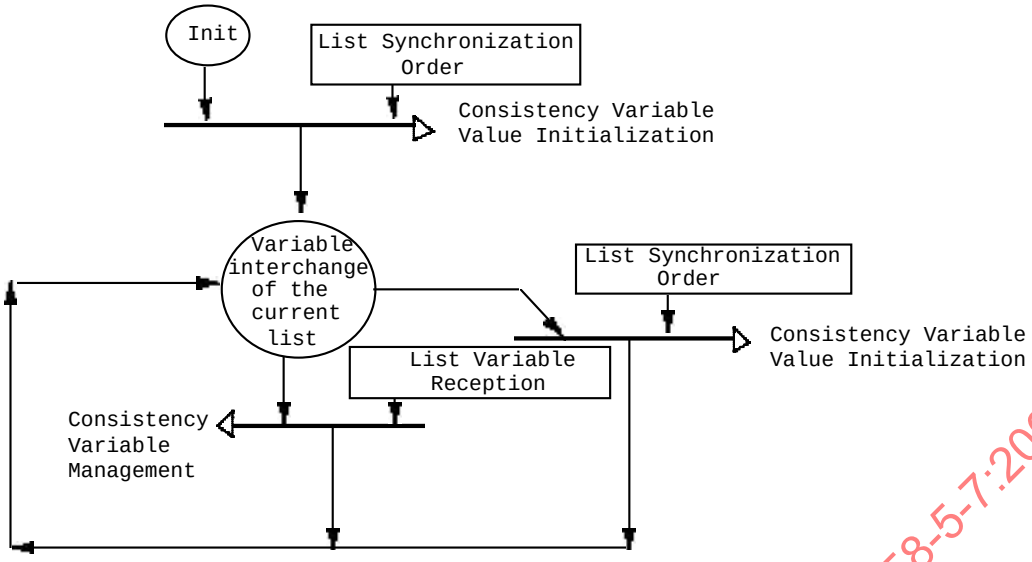


Figure 44 – Consistency variable value evaluation net

The initialization of the consistency variable value upon occurrence of the synchronization order associated with a variable list consists in giving its different fields a specific value that characterizes the implementation of the consistency processing.

The value of the consistency variable at the end of the initialization is as follows:

– the ' Interchange number ' field specifies the interchange number corresponding to the synchronization variable value which triggered this initialization,

– the ' Transmission status ' field specifies, for each variable of the list, a transmission status 'FALSE',

– the ' Transmission date ' field specifies for each the variable composing the variable list with which the consistency variable is associated, an interchange number corresponding to that which triggered the initialization.

The initialization of a consistency cycle is performed upon occurrence of the synchronization order associated with the variable list. This initialization corresponds to an assignation of the consistency variable field ' *Interchange number* ' of the synchronization variable value which corresponds to the synchronization order of the list. The consistency variable value management upon reception of a variable consists in modifying the value of the Transmission_Status and Transmission_Date according to the selected predetermined type.

In the case of the Transmission_Status field, the value management consists in making a change of the transmission status of each of the variables of the received list from FALSE to TRUE.

In the case of the Transmission_date field, the management value consists in memorizing the current interchange number in the transmission date associated with the variable.

Example

Given the list {V1, V2, V3, V4, V5} and the associated synchronization variable: S_LIST

Interchange timing diagram:

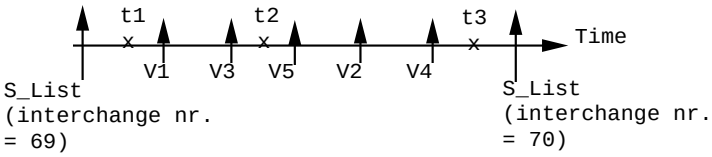


Figure 45 – Consistency interchange timing diagram

If we consider that all the variables were received during the previous interchange, the consistency variable associated with the various instants of the interchange has the following values:

t1 : Consistency variable value
Inconsistency Detection:= TRANSITORY {69; 0,0,0,0,0}
Inconsistency Detection:= PERMANENT {69; 68,68,68,68,68}
t2 : Consistency variable value
Inconsistency Detection:= TRANSITORY {69; 1,0,1,0,0}
Inconsistency Detection:= PERMANENT {69; 69,68,69,68,68}
t3 : Consistency variable value
Inconsistency Detection:= TRANSITORY {69; 1,1,1,1,1}
Inconsistency Detection:= PERMANENT {69; 69,69,69,69,69}

6.2.1.4.4.8 Access to the consistency variables

The consistency variables are handled by a subset of the services provided to the user for normal variables.

The services permitting access to the consistency variables are those existing for the synchronization variables, with the following restriction:

A_WRITELOC: Only when the *Inconsistency detection* attribute has a value other than UNDEFINED.

6.2.1.4.4.9 Specification of the recovery mechanism

A variable list can be provided with a recovery mechanism whose implementation depends on this list according to the user needs concerning the reliability.

This recovery mechanism elaborates recovery list contents corresponding to an ordered set of references to elements of the variable list which were not successfully received since the last reception of the list synchronization variable, under the condition that this synchronization variable has arrived within a time interval smaller than the recovery period.

The sequence operation of the variable references in the recovery list is performed in the declaration order of the list variables.

The references to the list variables, stored in the recovery contents by this mechanism, corresponds to the variable identifiers of this list. These recovery contents which conditions the initiation of new interchanges are limited to a maximum number of references defined at the time of configuration of the variable list.

NOTE The use of the recovery mechanism, with a recovery period equivalent to the network period of the synchronization variable of the list, releases all the losses which appeared during the list synchronization variable transmission.

Figure 46 shows the evaluation net of the service element which elaborates the contents of the recovery list associated with a variable list at the application level.

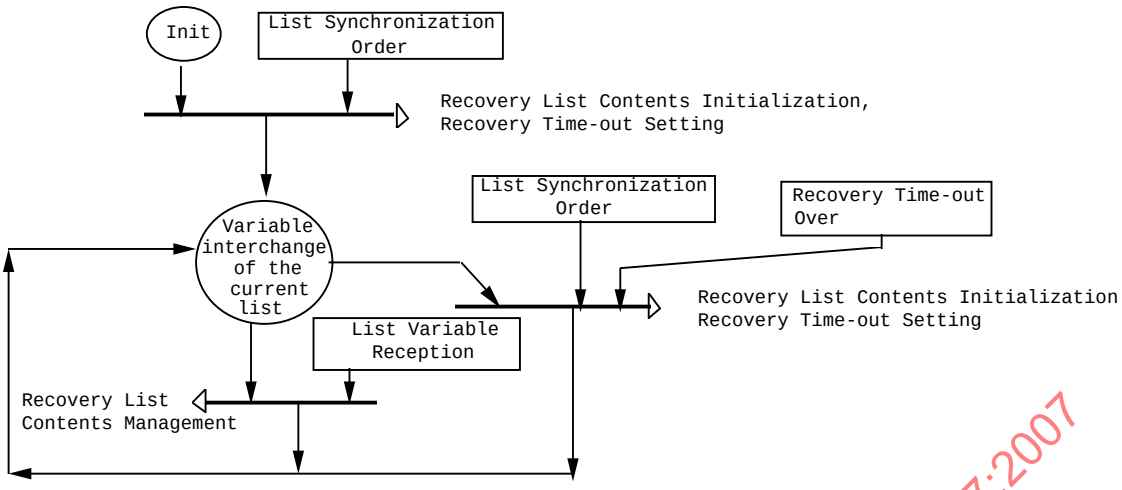


Figure 46 – Recovery mechanism evaluation net

• The initialisation of the recovery list contents, upon occurrence of the synchronization order associated with the variable list, consists in giving it the list of all the variable list identifiers as contents value.

As the synchronization variable associated with the list precedes all the list variables during an interchange, it should be considered that all the list variables are not received at the instant of the synchronization order occurrence.

- The recovery list management upon reception of a variable, consists in removing the identifier associated with this variable of this recovery list.
- The setting of the recovery time-out is performed upon occurrence of the list synchronization order and also automatically when it is expired. When it is set, the value of the *Preselection* attribute of this time-out is that of the *Recovery period* attribute of the *Variable List Local Image* object.

Example:

Given the list {V1, V2, V3, V4, V5} and the associated synchronization variable: S_LIST. The maximum recovery list size is limited to 4 elements.

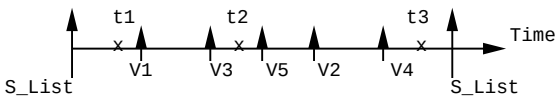


Figure 47 – Recovery interchange timing diagram

The recovery list, shown in Figure 47, has the following values at the different interchange instants:

t1 : Recovery list contents = { V1, V2, V3, V4}

t2 : Recovery list contents = { V2, V4, V5}

t3 : Recovery list contents = { ∅ }

6.2.1.5 Description of the configuration

The description specification of the application layer objects refers to static attributes of the objects:

– Variable of class normal, synchronization (by limiting to those attributes shared by the producer and the consumer plus those specific to the producer).
– Type of a variable.
– Access to a variable.
– Variable list (by limiting to those attributes shared by the consumers of this list).

6.2.1.5.1 Description variables

The configuration description accessible to the user anywhere on the network should be addressed at the interface between the application layer and the user.

For this purpose, the configuration description associated with the various objects of the application layer is modelled by description variables.

These description variables are themselves subjected to particular constraints which endows them with the same configuration on the network, in order not to define recursive description variables describing the description variable configuration.

These particular constraints are translated at the application layer level, into predefined values of the *Consumed Variable Local Image* and *Produced Variable Local Image* objects associated to the description variables.

6.2.1.5.2 Specification of a description variable

6.2.1.5.2.1 Description variable overview

A description variable in the AL environment is modeled in the same way as the other variables using the following objects.

- A *Produced Variable Local Image* object, which includes all the attributes required for the definition of a description variable in a producing application entity.
– A *Consumed Variable Local Image* object, which includes all the attributes required for the definition of a description variable in a consuming application entity.

Both these object categories correspond to particular instantiation of the *Identified Variable* class, for which the *Class* attribute of the object has the value DESCRIPTION.

A *Produced Variable Local Image* or *Consumed Variable Local Image* object corresponding to a variable of class DESCRIPTION are different from those corresponding to ' normal ' variables by predetermined values of some of its attributes.

6.2.1.5.2.2 Predetermined values of the *Identified Variable* class attributes.

An *Identified Variable* class corresponding to a description variable is characterized by values predetermined for the following attributes:

Transmitted Status:

This attribute has the value ' FALSE '.

Class:

This attribute has the value ' DESCRIPTION '.

All the non quoted attributes are not given predetermined values.

6.2.1.5.2.3 Predetermined values of the attributes of the *Produced Variable Local Image* object

A *Produced Variable Local Image* object corresponding to a description variable is characterized by values predetermined for the following attributes:

Resynchronized Variable:

This attribute has the value ' FALSE '.

Elaborated Refreshment:

This attribute has the value ' FALSE '.

Elaborated Punctual Refreshment:

This attribute has the value ' FALSE '.

Required Universal Services:

This attribute has the value ' FALSE '.

All the non quoted attributes are not given predetermined values.

6.2.1.5.2.4 Predetermined values of the attributes of the *Consumed Variable Local Image* object

A *Consumed Variable Local Image* object corresponding to a description variable is characterized by values predetermined for the following attributes:

Resynchronized Variable:

This attribute has the value ' FALSE '.

Required Universal Services:

This attribute has the value ' FALSE '.

All the non quoted attributes are not given predetermined values.

6.2.1.5.2.5 Description variable names

The description variable names handled at the interface between the application layer and the user, belong to the unique addressing space referencing the application objects. In this MPS addressing space, the standard defines four (reserved name sets) to address the description variables:

A_Name of the variable starting by:

"**V**_" Description of a variable

"**A**_" Description of a variable access

"**T**_" Description of a variable type

"**L**_" Description of a variable list.

6.2.1.5.2.6 Description variable types

6.2.1.5.2.6.1 Description variable type definition

The values of description variables are given a type which depends on the nature of the description, corresponding to predefined instances of the *Type Constructor* class.

6.2.1.5.2.6.2 Predefined instances of the *Type Constructor* class

Four predefined instances provide semantics to the types of the description variables.

- variables of NORMAL, SYNCHRONIZATION class.
- renamed accesses to variables of NORMAL, SYNCHRONIZATION class.
- types of variables.
- lists of variables.

Instance of the *Type Constructor* class associated with the type of a description variable of a variable of NORMAL, SYNCHRONIZATION class:

A_Name : K_DVAR
Construction : PREDEFINED

Instance of the *Type Constructor* class associated with the type of a description variable of a renamed access to a variable:

A_Name : K_DACCESS
Construction : PREDEFINED

Instance of the *Type Constructor* class associated with the type of a description variable of a type:

A_Name : K_DTYPE
Construction : PREDEFINED

Instance of the *Type Constructor* class associated with the type of a description variable of a variable list:

A_Name : K_DLIST
Construction : PREDEFINED

NOTE It has to be stated that the predefined type description of the environment means only that they are predefined. The precise knowledge of the semantics of the predefined types in the AL environment is part of the conformance to the MPS standard.

6.2.1.5.2.6.3 Semantics of the predefined types of the description variables

The types of the description variables provide a semantics to these variables. These semantics reflect the values of the static attributes shared by all the instances of the described objects as well as some attributes specific to the producer of the object.

The semantics of description variable type of a variable of NORMAL or SYNCHRONIZATION class describe the values of the following attributes, if they exist:

A_Name

Type Constructor Reference

Transmitted Status

Significant Status

Identifier

Transmission Mode

Network Period

Class

Consistency Variable

Variable List Reference

Resynchronized Variable

Synchronization Variable Reference

Elaborated Refreshment

Refreshment Characteristics

Production Period

Synchronization Variable Reference

Elaborated Punctual Refreshment

Time slot

Synchronization Variable Reference

The semantics of the description variable type of a renamed access to a variable describe the values of the following attributes, if they exist:

A_Name

Variable Reference

Access Mode

Access Path

The semantics of the description variable type of a variable type describe the values of the following attributes, if they exist:

A_Name

Construction

Primitive Type

Size

Compacted

Dimension

Reference Type Constructor

Named Field

Field Name

Reference Type Constructor

The semantics of the description variable type of a list variable describe the values of the following attributes, if they exist:

A_Name

List of Reference Variable

Spatial Consistency Required

Recovery Required

Reference Synchronization Variable

Inconsistency Detection

List of Reference Consistency Variable

Recovery Period.

6.2.1.5.2.6.4 Identifier of the description variables

The identifiers accepted for the description variables belong to a reserved subset defined by the network management standard.

Moreover, there exists an addressing relationship between identifiers of the variables and those of the description variables which is defined by the network management standard.

6.2.1.5.2.6.5 Access to the description variables

The description variables can be handled by a subset of services provided to the user for normal variables.

The services providing access to the description variables are

- A_WRITELOC
- A_WRITEFAR
- A_READLOC
- A_READFAR
- A_SENT
- A_RECEIVED
- A_UPDATE

All other services, provided to the user for a normal variable, are not permitted for a description variable.

6.2.2 Virtual model of a device (VMD) ASE

6.2.2.1 Overview

The modelling of a Sub-MMS VMD within an application is similar to its equivalent MMS.

6.2.2.2 Concepts

6.2.2.2.1 Scope of objects

In order to simplify object identification, the listed classes of objects do not have the scope (in the MMS sense) explicitly specified.

In other words appurtenance of an object of a given class to another object of another class is implicitly known (variable belonging to a particular domain for example) and should not appear in the object identification.

6.2.2.2.2 Object identification

The identification of objects "variable", "variable-list", "pi", "domain" and "event" is done by an identifier capable of taking either the form of an integer value (Index), or the form of a character string (Name). The number of characters in the string is either undetermined or determined by the companion standards.

Taking into consideration the elimination of the concept of scope, the identification of an object should be unique, either within its class or within a space common to all the classes. An object can be identified by an index and/or a name.

6.2.2.2.3 Environment management

6.2.2.2.3.1 Environment management services

The Sub-MMS environment management services are

1- Initiate, allows negotiating the establishment of a Sub-MMS environment.

2- Conclude, allows proposing the closing of a negotiated Sub-MMS environment.

3- Abort, allows suddenly breaking a negotiated Sub-MMS environment.

The initiate and Abort services are compulsory for the devices supporting the management of the negotiated Sub-MMS environment. The Conclude service is negotiable.

The Reject service allows the notification of protocol errors. It is compulsory for all the equipment.

6.2.2.2.3.2 Type of communication

6.2.2.2.3.2.1 Supported communcation types

A device can support the following types of communication:

a) communication in a negotiated Sub-MMS environment,

b) communication in a predefined Sub-MMS environment,

c) communication without Sub-MMS environment.

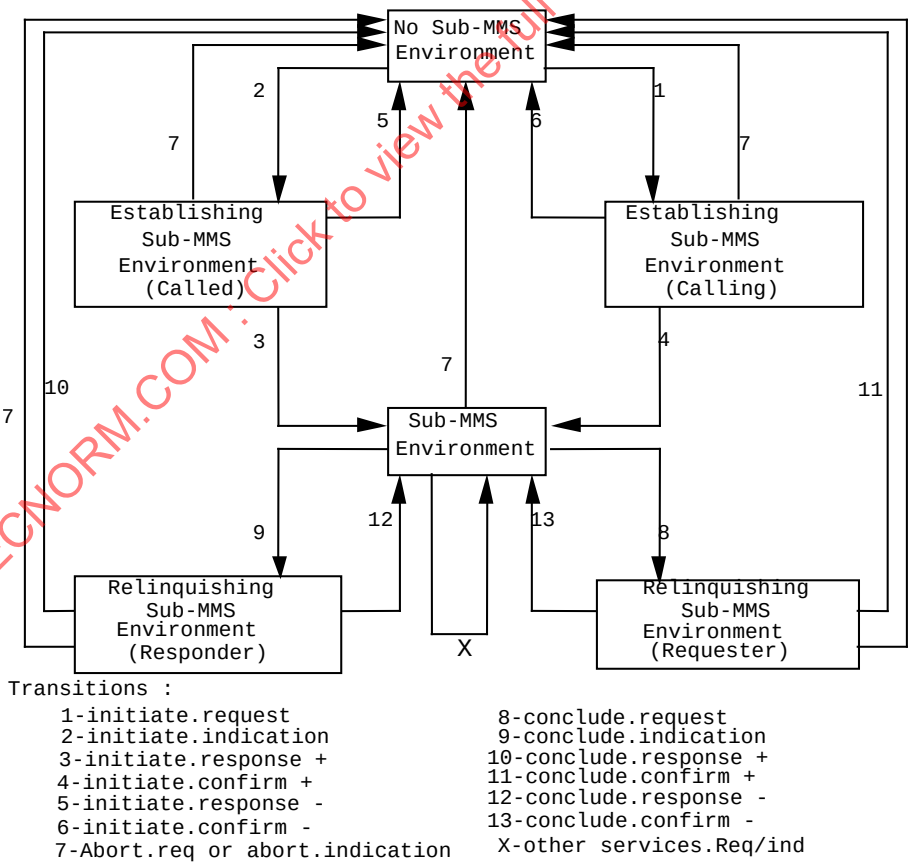


Figure 48 – Flowchart of the sub-MMS environment management state

6.2.2.2.3.2.2 Communication in a negotiated sub-MMS environment

The Sub-MMS environment is negotiated between a user of the calling Sub-MMS and a user of the called Sub-MMS.

The resources allocated to this environment are reserved for these two users.

A Sub-MMS environment known as association has several states, as shown in Figure 48. The initiate, conclude and abort services can act on the association.

The initial state for the calling Sub-MMS as well as the called should be the state "No Sub-MMS Environment".

In the state "No Sub-MMS Environment", a user of the sub-MMS should only invoke the request of the initiate service primitive.

In the "Establishing Sub-MMS Environment (Calling)" state, a user of the sub-MMS should only invoke the request of the Abort service primitive.

In the "Establishing Sub-MMS Environment (Called)" state, a user of the sub-MMS can only invoke the response to the initiate service primitive or the request of the Abort service primitive.

In the "Relinquishing Sub-MMS Environment (Requester)" state, the only request for a primitive which a user of the sub-MMS can invoke is the request of the Abort service primitive. It is noted that the responses (+) or (-) can continue to be invoked.

In the "Relinquishing Sub-MMS Environment (Responder)" state, a user of the Sub-MMS can only invoke the request of the Abort service primitive and the response to the Conclude service primitive.

The only events which provoke the output of the "Sub-MMS environment" state are the invocations of the Abort service primitive request, the Conclude service primitive request as well as the reception of the indications of the Abort service primitive and the Conclude service primitive.

In the "Sub-MMS Environment" state, a user of the sub-MMS can invoke any of the requests or responses of the previously negotiated service primitives, on condition that the following points are observed:

a) other sections of this document restraining the use of the services through sequencing constraints;

b) a response to a primitive should not be invoked unless a request of the primitive corresponding to a service indication has been received;

c) the request of the initiate service primitive should not be invoked.

6.2.2.2.3.2.3 Communication in a predefined sub-MMS environment

The SUB-MMS environment is established implicitly between two or more users. The resources allocated to this environment are reserved to its users.

The Sub-MMS environment management services (Initiate, Conclude, Abort) are not signficant.

If an Initiate or Conclude Request PDU is received, a Reject PDU is issued with the parameters Reject PDU type equal to PDU ERROR and Reject Code equal to ILLEGAL-MAPPING.

If a Request PDU of a confirmed service not supported in this environment is received, a Reject PDU message is issued with the parameters Reject PDU Type equal to CONFIRMED. REQUEST PDU and Reject Code equal to UNRECOGNIZED SERVICE.

NOTE 1 A Sub-MMS environment established between more than two users only supports unconfirmed services.

NOTE 2 The reception of an Abort.Indication is ignored.

6.2.2.2.3.2.4 Communication without sub-MMS environment

No Sub-MMS environment is established between two or more users.

The resources allocated to this type of communication are available to all the potential remote users.

The Sub-MMS environment management services (Initiate, Conclude, Abort) are not supported.

If an Initiate or Conclude Request PDU is received, a Reject PDU is issued with the parameters Reject PDU Type equal to PDU ERROR and Reject code equal to ILLEGAL MAPPING.

If a Request PDU of a confirmed service not supported by the equipment is received, a Reject PDU is issued with the Reject PDU Type parameters equal to CONFIRMED.REQUEST PDU and Reject Code equal to UNRECOGNIZED-SERVICE.

NOTE The reception of an Abort.Indication is ignored.

6.2.2.2.3.3 Error types

6.2.2.3 VMD class specification

6.2.2.3.1 VMD class model

FAL ASE:	**VMD ASE**
CLASS: VMD	
CLASS ID:	not used
PARENT CLASS:	not used
Object: VMD	
(m) Key Attribute:	Executive Function
(m) Attribute:	Vendor Name
(m) Attribute:	Model Name
(m) Attribute:	Revision
(m) Attribute:	Logical Status (STATE-CHANGES-ALLOWED,NO-STATE-CHANGES-ALLOWED, OTHER)
(m) Attribute:	Physical Status (OPERATIONAL, PARTIALLY-OPERATIONAL, INOPERABLE, NEEDS-COMMISSIONING)
(m) Attribute:	List of Capabilities
(m) Attribute:	List of Program Invocations
(m) Attribute:	List of Domains
(m) Attribute:	List of Variables
(m) Attribute:	List of Variable Lists
(m) Attribute:	List of Events
(o) Attribute:	Additional Detail

6.2.2.3.2 Attributes

Executive function:

This attribute represents the set of software ensuring the service procedures.

Vendor name:

This attribute identifies the constructor of the device.

Model name:

This attribute identifies the device model which supports the VMD.

Revision:

This attribute identifies the version index of the device system supporting the VMD. The value of this attribute is assigned by the vendor.

Logical status:

Sub-MMS distinguishes two levels of functions which are described in this attribute. Other levels of functions could be defined eventually without altering the interoperability of the device.

1-STATE-CHANGES-ALLOWED:

In this status all the VMD supported services can be carried out,

2-NO-STATE-CHANGES-ALLOWED:

In this status only the following services can be carried out, if they are VMD supported:

- Initiate
- Conclude
- Abort
- Identify
- Status
- Read
- Get Alarm summary
- Get Name List
- Get Domain Attributes
- Get Program Invocation Attributes
- Get Variable Access Attributes
- Get Variable List Attributes
- Get Event Condition Attributes

3-OTHER:

Other statuses can be defined. These statuses should be described in the PICS.

Physical Status:

This attribute gives the hardware status of the device

List of capabilities:

This attribute enables defining a set of characteristics specific to the VMD.

List of program invocations:

This attribute gives the set of invocation programs belonging to the VMD. The program invocations can be predefined in the VMD or created dynamically by a specific Sub-MMS service or following a domain loading.

List of domains:

This attribute gives the set of Domains belonging to the VMD. The domains can be predefined in the VMD or created dynamically by the loading services of the SUB-MMS domains.

List of variables:

This attribute gives the set of objects of the variable class belonging to the VMD. The variables can be predefined in the VMD or created dynamically following a domain loading.

List of variable lists

This attribute gives the set of object variable lists belonging to the VMD. The variable lists can be predefined in the VMD or created dynamically by a specific Sub-MMS service or following a domain loading.

List of events:

This attribute gives the set of events belonging to the VMD. The events could be predefined in the VMD or created dynamically following a domain loading.

Additional detail:

This attribute can contain additional information

Besides the VMD object defined above, the Sub-MMS application layer also supports the objects mentioned below:

- Domain objects,
- Program Invocation objects,
- Variable objects,
- Variable list objects,
- Event objects…

The C.S can introduce other classes of objects.

6.2.2.4 Services

6.2.2.4.1 Confirmed initiate service

6.2.2.4.1.1 Functionality

The initiate service should be used to establish the environment of the Sub-MMS (association) in order to authorize the exchange of information between two users of the Sub-MMS according to their resources and requirements.

6.2.2.4.1.2 Service primitives

The structure of the service is shown in Table 32.

Table 32 – Confirmed initiate service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument				
Quality of service calling	M	M (=)		
Extension Calling	C	C (=)		
Proposed Max Serv Outstanding Calling	M	M		
Proposed Max Serv Outstanding Called	M	M		
Proposed Data Structure Nesting	M	M		
Init Request Detail	M	M		
Result(+)			S	S (=)
Quality of service called			M	M (=)
Extension Called			C	C (=)
Negotiated Max Serv Outstanding Calling			M	M (=)
Negotiated Max Serv Outstanding Called			M	M (=)
Negotiated Data Structure Nesting			M	M (=)
Init Response Detail			M	M
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

This parameter should transport the service parameters specific to the initiate service request.

For the parameters of the request primitive, the Sub-MMS supplier in the calling system can reduce the values supplied by the calling Sub-MMS user. For the parameters of the indication primitive, the supplier of the Sub-MMS in the called system can reduce the values supplied by the InitiateRequestPDU except for the Indication Services Support Calling parameter. The reduction of the values should follow the reduction rules as defined in the parameter descriptions. No other modification of these values is authorized.

NOTE The fact of authorizing the service suppliers to reduce the values proposed by the users, gives a mechanism which can be used to check the adequation of the values given by the user to the resources of the service supplier. Its implementation is a local problem.

Quality of the calling service:

This parameter should indicate the choice of the service quality. This standard distinguishes between two service qualities. The level 1 service quality is reserved for communications without object access protection. Whereas the level 2 quality is that using object access protection. In this case, an access right verification mechanism is installed and checks the compatibility of the Password and Access Groups parameters provided during invocation of the Initiate service request, with the parameters Password, Access Groups and Access Rights of the objects which are accessed by the calling user.

The information necessary for the service 2 quality are those supplied in the Extension Calling parameter.

New service qualities can be specified. We recommend that the information necessary for these new service qualities are provided in the Extension calling parameter.

Extension calling:

The structure of this parameter is shown in Table 33.

Table 33 – Detailed structure of the extension calling parameter

Parameter name: Extension Calling	Req	Ind
Quality of service 1	C	C (=)
Null	M	M (=)
Quality of service 2	C	C (=)
Password	M	M (=)
Access Groups	M	M (=)
Others Qualities of service	C	C (=)
Local Detail Calling	U	U (=)

Password:

This parameter, present for service 2 quality, defines the password, unique for any association. It will be used for access to all the objects of the called user on this association. If access by a password is not requested, this parameter should take value 0.

Access groups:

This parameter, present for a ,service 2 quality, defines the access groups to which the calling user belongs. This belonging is applied for the access to all the objects of the Server on this association.

Local detail calling:

This parameter can specify information concerning the service quality extensions.

Proposed max serv outstanding calling:

This parameter, proposed by the calling user, should identify the maximum number of indications of confirmed services waiting for a response on the calling user side on this association.

The value of this parameter can be reduced by the supplier of the Sub-MMS service (this will then be different in the service indication).

Proposed max serv outstanding called:

This parameter, proposed by the calling user, should identify the maximum number of indications of confirmed services awaiting response from the called user on this association.

The value of this parameter can be reduced by the supplier of the Sub-MMS service (this will then be different in the service indication).

Proposed data structure nesting:

This parameter should indicate the maximum number of overlapping levels in the data structures used on this association between the two users of the Sub-MMS. The value of this parameter can be reduced by the supplier of the Sub-MMS service. The value taken by this parameter should be higher than or equal to 1.

Init Request detail:

The structure of this parameter is shown in Table 34.

Table 34 – Detailed structure of the init request detail parameter

Parameter name: Init Request Detail	Req	Ind
Proposed Version Number	M	M
Proposed Parameter CBB	M	M
Indication Services Support Calling	M	M
Extension	U	U (=)
Request Services Support Calling	M	M (=)
Access by Name Support Calling	M	M (=)

Proposed version number:

This parameter specifies the minor revision number of the Sub-MMS protocol services proposed by the calling user.

The value of this parameter can be reduced by the supplier of the Sub-MMS service (this will then be different in the service indication) but should always be higher than 1.

Proposed parameter CBB:

This parameter should specify the negotiated set of conformity parameters relative to the variables (CBBs) which will be supported on this association. The value of this parameter in the service indication primitive should represent the intersection between the set of CBBs supported by the calling user and the set of CBBs supported by the supplier of the Sub-MMS service.

Indication services support calling:

This parameter defines the indications of services supported by the calling user. The value of this parameter can be reduced by the supplier of the Sub-MMS service supplier of the calling user.

Extension:

This parameter, if present, includes two items of information: Request Services Support Calling and Access by Name Support Calling.

Request services support calling:

This parameter defines the requests of services supported by the calling user.

Access by name support calling:

This parameter defines if the access to the objects by their name is supported by the calling user as requester or responder of the service.

Result(+)

This selection indicates that the requested service has been executed with success. In this case the following parameters should be entered:

Quality of called service:

This parameter should indicate the choice of the called user in terms of service quality. This standard distinguishes two qualities of service. The level 1 quality service is reserved for communications without access protection for the objects whereas the level 2 quality service is that using access protection for the objects. In this case, a mechanism to check the access rights is installed and checks the compatibility of the Password and Access Group parameters provided during the invocation of the initiate service request with the Password, Access Groups and Access Rights parameters of the objects which are accessed by the called user.

The information necessary for the quality of service 2 are supplied in the Extension Called parameter.

New service qualities can be specified. We recommend that the information necessary for these new service qualities are supplied in the Extension called parameter.

Extension called:

The structure of this parameter is shown in Table 35.

Table 35 – Detailed structure of the extension called parameter

Parameter name: Extension Called	Rsp	Cnf
Quality of service 1	C	C (=)
Null	M	M (=)
Quality of service 2	C	C (=)
Password	M	M (=)
Access Groups	M	M (=)
Others Qualities of service	C	C (=)
Local Detail Called	U	U (=)

Password:

This parameter, present for a service 2 quality, defines the password, unique for all associations. It will be used for access to all the objects of the calling user on this association. If access with the help of the password is not requested, this parameter should take value 0.

Access groups:

This parameter, present for a service 2 quality, defines the access groups to which the calling user belongs. This belonging applies for the access to all the objects of the calling user on this association.

Local detail called:

This parameter can specify the information concerning the service quality extensions.

Negotiated max serv outstanding calling:

This parameter should identify the maximum number of indications of confirmed services waiting for a response on the calling side of this association.

The value of this parameter can be reduced in the response primitive with respect to the value contained in the indication primitive.

Negotiated max serv outstanding called:

This parameter should indicate the maximum number of indications of confirmed services waiting for a response on the called side of this association.

The value of this parameter should be reduced in the response primitive with respect to the value contained in the indication primitive.

Negotiated data structure nesting:

This parameter should indicate the maximum number of confirmed services waiting for a response on the side of the called party in this association.

The value of this parameter should be reduced in the response primitive with respect to the value contained in the indication primitive.

Negotiated data structure nesting:

This parameter should indicate maximum number of nesting levels in the data structures used on this association between the two users of the Sub-MMS.

The value of this parameter can be reduced in the response primitive with respect to the value contained in the indication primitive. The value taken by this parameter should be greater than or equal to 1.

Init response detail:

The structure of this parameter is shown in Table 36.

Table 36 – Detailed structure of the init request detail parameter

Parameter name: Init Response Detail	Req	Cnf
Negotiated Version Number	M	M (=)
Negotiated Parameter CBB	M	M (=)
Indication Services Support Called	M	M
Extension	C	C (=)
Request Services Support Called	M	M (=)
Access by Name Support Called	M	M (=)

Negotiated version number:

This parameter specifies the minor revision number of the Sub-MMS protocol proposed by the called user.

The value of the parameter should be less than or equal to the proposed value; but should always be higher than 1.

Negotiated parameter CBB:

This parameter should specify the negotiated set of CBBs which will be supported on this association. The value of this parameter in the service response primitive should represent the intersection between the set of CBBs supported by the called user and the set of CBBs supplied in the indication primitive.

Indication services support called:

This parameter defines the indications of services supported by the called. The value of this parameter can be reduced by the supplier of the Sub-MMS service of the called user.

Extension:

This parameter should be present if the extension parameter is included in the request. This parameter specifies two items of information: Request Services Support Called and Access by Name Support Called.

Requester services support called:

This parameter defines the services supported by the called user.

Access by name support called:

This parameter defines if the access to the objects by their name is supported by the called user as service requester or responder.

Result(-):

This parameter should indicate that the requested service is unsuccessful. The type of error is indicated by the Error Type parameter.

6.2.2.4.1.3 Service procedure

The called user transmits a positive response if he accepts the proposed values or if he is capable of proposing others in compliance with the rules for reduction of the values. Otherwise he transmits a negative response.

The success of the execution of the initial service should result in the establishment of a Sub-MMS environment. If an Initiate Request PDU is received on an existing association, a Reject PDU should be issued with the parameters Reject PDU Type equal to PDU-ERROR and Reject Code equal to ILLEGAL-MAPPING.

NOTE For reasons of simplicity, this document recommends a single abstract syntax and a single transfer syntax per association. Consequently, an Initiate Request PDU will be accepted or refused according to the proposed context.

6.2.2.4.2 Confirmed conclude service

6.2.2.4.2.1 Functionality

The Conclude service can be used to provoke the ordered termination of the Sub-MMS environment. A Sub-MMS user invokes the Conclude service to indicate that he has accomplished the requests which have been scheduled and that no other request will be invoked.

6.2.2.4.2.2 Service primitives

The structure of the Conclude service is shown in Table 37.

Table 37 – Conclude service parameter

Parameter Name	Req	Ind	Rsp	Cnf
Argument				
Result(+)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

There is no parameter specific to this service in the Conclude service request.

Result(+)

This parameter indicates that the requested service has been successfully executed.

Result(-)

This parameter should indicate that the service request has failed. The Error Type parameter indicates the reason for this failure.

6.2.2.4.2.3 Service procedure

Calling procedure:

1 The invocation of the Conclude.Request primitive is forbidden on an association via which a domain loading is in progress and whose "State" attribute of the object domain has a value different from "READY", "IN-USE".

2 After the invocation of the Conclude.Request primitive:

a) the following actions should be performed:

- pass the Sub-MMS environment in the "Relinquishing Sub-MMS Environment" state (Requester),
- continue to invoke the response primitives in order to serve the requests of his correspondent,
- continue to process the received service confirmations.

b) the following constraint should be observed:

- no service request can be invoked besides Abort.

3 After reception of the Conclude.Cnf(+) primitive:

a) the following action should be performed:

– pass the Sub-MMS environment to the "No Sub-MMS Environment" state.

b) the following constraint should be observed:

– no service invocation should be made on this association.

4 After reception of the Conclude.Cnf(-) primitive:

a) the following actions should be performed:

– pass the Sub-MMS environment to the "Sub-MMS Environment" state,

– if necessary, continue to invoke the service primitives.

Called user procedure

1 Following reception of the Conclude.Indication primitive:

a) the following action should be performed:

- pass the Sub-MMS environment to the "Relinquishing Sub-MMS Environment" (Responder)state,

b) the following constraints should be observed:

- answer in priority to the Conclude.Ind primitive.
- no service request can be invoked except Abort.

2 Invoke the Conclude.Response(+) primitive if the following constraints are observed:

– no request is waiting for a response from its correspondent,

– no request made by its correspondent is waiting for a response,

– no domain loading is in progress.

– no saving of a domain is in progress.

a) The following action should be performed:

- switch the Sub-MMS environment to the "No Sub-MMS Environment" state,
- continue, if necessary, to invoke the primitive services.

b) the following constraint should be observed:

- no service invocation should be made on this association.

3 Invoke the Conclude.Response(-) primitive if one of the following constraints is not respected:

– at least one request is waiting for a response of its correspondent,

– at least one request by its correspondent is waiting for a response,

– at least one domain loading is in progress,

– at least one domain saving is in progress.

a) the following actions should be performed:

- pass the Sub-MMS environment to the "Sub-MMS Environment" state,
- if necessary, continue to invoke the service primitives.

NOTE We recommend reserving for the calling user the role of conclude service initiator.

6.2.2.4.3 Unconfirmed abort service

6.2.2.4.3.1 Functionality

The Abort service should be used to quit the Sub-MMS environment without negotiation. The Sub-MMS user should invoke the Abort request primitive to indicate that it wishes to stop the communications on the association immediately and without negotiation. The Sub-MMS Abort service comes from and uses the MCS A_ABORT service. (See projection of the Sub-MMS PDU on the Type 7 MCS services of IEC 61158-6).

NOTE The Abort indication service primitive can also be generated by the supplier of the Sub-MMS service.

6.2.2.4.3.2 Service primitives

The structure of the service is shown in Table 38.

Table 38 – Unconfirmed abort service parameters

Parameter Name	Req	Ind
Argument		
Locally Generated		M
User Information	U	U

Argument

The argument should transport the specific parameters of the Abort service.

Locally Generated

This parameter should indicate if the Abort request has been generated locally by the supplier of the Sub-MMS services, or if the Abort request has been received. This parameter should be filled in by the Sub-MMS supplier.

User Information

This parameter, when present, should contain additional information concerning the reasons for this Abort.

6.2.2.4.3.3 Service procedure

In the case of an Abort initiated by the Sub-MMS user, the correspondent should be informed of this Abort by the reception of an Abort service indication primitive and the Sub-MMS environment should be deleted.

In the case of an Abort initiated by the Sub-MMS supplier, the two correspondents should be informed of this Abort by the reception of Abort Service indication primitives (if this is possible) and the Sub-MMS environment should be deleted.

After destruction of the Sub-MMS environment, all the objects with a specific link with the Sub-MMS environment in question are deleted.

6.2.2.4.4 Unconfirmed reject service

6.2.2.4.4.1 Functionality

The Reject service is a service initialized by the supplier following a protocol error.

The protocol errors of a unconfirmed service or a confirmed service response does not result in the issue of a Reject PDU.

6.2.2.4.4.2 Service primitives

The structure of the service is shown in Table 39.

Table 39 – Unconfirmed reject service parameters

Parameter Name	Ind
Argument	
Detected here	M
Service instance	M
Reject PDU type	M
Reject Code	M

Argument

This argument transports the specific parameters of the Reject service.

Detected here

This parameter, of the Boolean type, informs the user if the protocol error which resulted in the rejection has been detected by the local or remote Sub-MMS supplier. The true value signifies that the protocol error has been detected locally.

Service instance

This parameter should indicate the service instance for which an error has been detected. This parameter is not significant if the reject concerns a ConcludeReqPDU.

Reject PDU type

This parameter should indicate the type of PDU which has caused the protocol error. The PDU-ERROR value should be used when the PDU is not a valid Sub-MMS PDU. The possible values for a PDU-ERROR are a sub-assembly of those defined in the MMS.

The values of the Reject PDU Type selected are:

- CONFIRMED-REQUESTPDU,

- PDU-ERROR,

- CONCLUDE-REQUESTPDU.

Reject code

This parameter should indicate the reason for which the Reject has been issued as a function of the Reject PDU Type parameter values. The possible values for the Reject Code are a sub-set of those defined in MMS.

The Reject Code parameter values selected are

1- For CONFIRMED-REQUESTPDU:

- OTHER

  This code should be used for errors other than those defined in the Sub-MMS standard;

- UNRECOGNIZED-SERVICE

  This code should be used when the service is not supported or is not recognized.

- INVALID-INVOKE-ID

This code should be used when an invocation number does not follow the recommendations of this standard.

- INVALID-ARGUMENT

  This code should be used when an argument of a service does not follow the recommendations of this standard;

- MAX-SERV-OUTSTANDING-EXCEEDED

  This code should be used when the maximum number of confirmed negotiated services which can be waiting is exceeded on receiving a confirmed service request:

- MAX-RECURSION-EXCEEDED

  This code should be used when the received PDU, in the sense of overlapping of the data structures, exceeds that which has been negotiated.

- VALUE-OUT-OF-RANGE

  This code should be used when the received PDU contains one or more parameters whose values exceed those authorized by this standard.

2- For PDU-ERROR:

- UNKNOWN-PDU-TYPE

  This code should be used when the type of PDU received is not recognized or is not supported;

- INVALID-PDU

  This code should be used when the received PDU is syntaxically incorrect and a more detailed processing is impossible because of the seriousness of the error.

- ILLEGAL-MAPPING

  This code should be used when the an Initiate.Request is received via an existing association.

3- For CONCLUDE-REQUESTPDU:

- OTHER

  This code should be used for errors other than those defined in the Sub-MMS standard;

- INVALID-ARGUMENT

  This code should be used when a service argument does not follow the recommendations of this standard.

6.2.2.4.4.3 Service procedure

If a Sub-MMS supplier receives a RejectPDU, it should be shown by an indication of the Reject service to the Sub-MMS user. He can then use the Abort service to end the Sub-MMS environment abruptly.

A supplier of the Sub-MMS should be capable of transmitting a RejectPDU if it receives a PDU which generates a protocol error.

6.2.2.4.5 Confirmed status service

6.2.2.4.5.1 Service primitives

The structure of the service is shown in Table 40.

Table 40 – Confirmed status service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
extended derivation	M	M (=)		
Result(+) (Comp)			S	S (=)
vmd logical status			M	M (=)
vmd physical status			M	M (=)
local detail			U	U (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Extended derivation:

This parameter indicates the method to be applied to provide the response to the Status request. Two distinct methods can be activated. Their definition is a local issue.

Result(+)

VMD logical status:

This parameter provides the VMD Logical Status attribute value.

VMD physical status:

This parameter provides the VMD Physical Status attribute value.

Local detail:

This parameter can contain the additional information on the VMD status specific to a vendor.

Result(-)

The execution of the service ended in failure. The Error type parameter indicates the cause of the failure.

6.2.2.4.5.2 Service procedure

The procedures necessary for the elaboration of different Status are carried out, and a valid response is built.

6.2.2.4.6 Unconfirmed unsolicited status service

6.2.2.4.6.1 Service primitives

The structure of the service is shown in Table 41.

Table 41 – Unconfirmed unsollicited status service parameter

Parameter Name	Req	Ind
Argument (Comp)		
vmd logical status	M	M (=)
vmd physical status	M	M (=)
local detail	U	U (=)

Argument

VMD logical status:

This parameter provides the VMD Logical Status attribute value.

VMD physical status:

This parameter provides the VMD Physical Status attribute value.

Local detail:

This parameter can contain the additional information specific to a vendor concerning the VMD status.

6.2.2.4.6.2 Service procedure

A SUB-MMS user capable of detecting a change of his status, can take the initiative to transmit the new values of his Status without receiving a request.

6.2.2.4.7 Confirmed identify service

6.2.2.4.7.1 Service primitives

The structure of the service is shown in Table 42.

Table 42 – Confirmed identify service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Result(+) (Comp)			S	S (=)
vendor name			M	M (=)
model name			M	M (=)
revision			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

No specific Parameter in the request.

Result(+)

Vendor name:

This parameter provides the VMD object Vendor Name attribute value.

Model name:

This parameter provides the VMD object Model Name attribute value.

Revision:

This parameter provides the VMD object Revision attribute value.

Result(-)

The service execution ended in a failure. The Error type parameter indicates the cause of the failure.

6.2.2.4.7.2 Service procedure

The information necessary for the construction of the positive response are provided by a VMD object.

6.2.2.4.8 Confirmed get name list service

6.2.2.4.8.1 Service primitives

The structure of the service is shown in Table 43.

Table 43 – Confirmed get name list service paramaters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Object class	M	M (=)		
Continue after	M	M (=)		
Result(+) (Comp)			S	S (=)
List of Identification			C	C (=)
List of Description			C	C (=)
More follows			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Object class

The Object class parameter defines the class of the object. The following values can be specified: Null, Domain, Program Invocation, Variable, Variable list, event and the classes of objects defined by the CS.

Continue after

This parameter gives the identification of the object from which the search should be done.

Result(+)

This selection indicates the successful accomplishment of the service.

List of identification:

This parameter is present if the value of the parameter object class is not null.

This parameter gives a list of object identifications belonging to a specified class in the request.

NOTE If an object supports the double identification (name and index), only the index should be restored.

List of object description:

This parameter is present if the value of the parameter class = null.

This parameter gives the description of each object read.

More follows:

This parameter indicates if there are other objects to be identified.

Result(-)

This selection indicates the failure of the service

Error type

This parameter indicates the cause of the failure

6.2.2.4.8.2 Service procedure

If all the operations are carried out successfully, the service provides, in its positive response, a list of object identifications that belong to the required class. If the class is not indicated (null) and if a common object identification space exists, the server then provides the description for each oject. The first identification begins just after the required identification.

In case of failure of any operation the server provides the reason for the failure in a negative response.

Whatever may be the quality of the channel service via which this service is directed, no access restriction should be applied by the object access protection.

6.2.3 Domain ASE

6.2.3.1 Overview

Sub-MMS defines a Domain object which can pre-exist or be created by a Sub-MMS service. The data or the data complement of a Domain object can be loaded or saved by sub-MMS services.

The domain management services are the following:

1 - Delete Domain

2 - Initiate Download Sequence

3 - Download Segment

4 - Terminate Download Sequence

5 - Initiate Upload Sequence

6 - Upload Segment

7 - Terminate Upload Sequence

8 - Get Domain Attribute

NOTE 1 The Delete Domain service can be locally executed.

NOTE 2 An already existing domain is spontaneously created either with all its data or with a part of its data.

6.2.3.2 Domain class specification

6.2.3.2.1 Domain class model

FAL ASE:	**Domain ASE**
CLASS: Domain	
CLASS ID:	not used
PARENT CLASS:	not used
Object: Domain	
(m) Key Attribute:	Domain name
(m) Key Attribute:	Domain index
(m) Attribute:	List of capabilities
(m) Attribute:	State (LOADING, COMPLETE, INCOMPLETE, READY, IN USE)
(m) Attribute:	Predefined (TRUE, FALSE)
(m) Attribute:	Sharable (TRUE, FALSE)
(m) Attribute:	List Of Program Invocation Identification
(m) Attribute:	Upload In Progress
(o) Attribute:	Domain Access Protection
(m) Attribute:	Password
(m) Attribute:	Access Groups
(m) Attribute:	Access Rights
(o) Attribute: Extension	

6.2.3.2.2 Attributes

Domain name, index:

These attributes allow identifying in an unique way the object domain. The types of identifier supported are either the name or the index or both.

State:

This attribute specifies the state of the Domain object in compliance with the state chart described below.

Predefined:

This attribute can take the values TRUE or FALSE. The FALSE value authorises the service Delete Domain to delete the object. In a pre-existent Domain Object this attribute is initialized at TRUE.

Sharable:

This attribute takes the value TRUE if the Domain object can be used in several program invocations simultaneously.

List of program identifications:

This attribute specifies the list of program invocations which use this Domain object. The list contains only a single identification of PI if the Sharable attribute is at FALSE.

Upload In progress:

This attribute indicates the number of uploads being executed on this domain.

Domain access protection:

This attribute indicates if the object supports an access protection.

Password:

This attribute specifies the value of the password accepted for the access rights.

Access groups:

This attribute specifies if the object belongs to a group of users, as shown in Table 44. If one of the bits is set to 1 it indicates the number of the group to which the object belongs.

Table 44 – Access group attribute description for domain object

Bit name	Access group number
G0	Access group 8
G1	Access group 7
G2	Access group 6
G3	Access group 5
G4	Access group 4
G5	Access group 3
G6	Access group 2
G7	Access group 1

Access Rights:

This attribute specifies the access rights associated with the Domain object, as shown in Table 45. Each bit, positioned to 1 indicates the acceptance conditions of the loading, saving or utilization services in a program invocation.

Table 45 – Access rights attribute description for domain object

Name	Significance
R	UPLOAD authorized to clients who have provided a password identical to the Password attribute of the object.
W	DOWNLOAD, Deleting authorized to clients who have provided a password identical to the Password attribute of the object
U	Utilization in a PI authorized to clients who have provided a password identical to the Password attribute of the object
Rg	Upload authorized to clients belonging to one of the groups specified in the Access Rights attribute of the object.
Wg	Download, Deleting authorized to clients belonging to one of the groups specified in the Access Rights attribute of the object
Ug	Utilization in a PI authorized to clients belonging to one of the groups specified in the Access Rights attribute of the object
Ra	Upload authorized to all the clients
wa	Download, Deleting authorized to all the clients
Ua	Utilization in a PI authorized to all the clients

Extension:

This attribute specifies additional information.

6.2.3.3 Services

6.2.3.3.1 Confirmed delete domain service

6.2.3.3.1.1 Functionality

This service allows a client, under certain conditions, to delete an existing domain.

6.2.3.3.1.2 Service primitives

The structure of the service is shown in Table 46.

Table 46 – Confirmed delete domain service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Domain identification	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

Domain identification:

This parameter specifies the identification by name or by index of a Domain object.

Result(+)

The service is valid and has been correctly executed by the VMD.

Result(-)

The service is invalid or not accepted by the VMD. An error code is provided.

6.2.3.3.1.3 Service procedure

The VMD should perform the following actions:

– check that there is an existing domain with the same identification,

– check that the predefined attribute is set to false,

– check that the Domain is in the ready state and that there is no uploading in progress,

– if the request is transmitted via a channel with the service quality 2, check the compatibility of the access rights of the channel with the password attributes, access group and access rights of the object,

– delete the object Domain in question.

If all the actions are executed with success, a positive report is transmitted, otherwise a negative report is transmitted explaining the reasons for the failure.

6.2.3.3.2 Confirmed initiate download sequence service

6.2.3.3.2.1 Functionality

There are two types of domains, those which are created dynamically by the Initiate Download Sequence service and those which are born partially or completely loaded at power up. In the latter case, the domain is said to be pre-existent.

The Initiate Download Sequence service allows a client to:

1 - initialize the loading of a pre-existent domain if it is in the "predefined" state,

2 - create the domain object then initialize the loading if the domain is inexistent.

6.2.3.3.2.2 Service primitives

The structure of the service is shown in Table 47.

Table 47 – Confirmed initate download sequence service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Domain Identification	M	M (=)		
List of capabilities	M	M (=)		
Sharable	M	M (=)		
Domain Access Protection	C	C (=)		
Password	M	M (=)		
Access Group	M	M (=)		
Access Right	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

Domain identification:

This parameter specifies the identification by name or by index of the domain to be downloaded.

List of Capabilities:

This parameter generally contains specific information concerning limitations of implementation. Its content can be empty.

Sharable:

The "sharable" parameter is a Boolean; if its value is true, the Domain can be divided by several PIs.

Domain Access Protection:

Its presence is compulsory if the service is transmitted via a channel with the quality of service (2).

This parameter specifies the value to be assigned to the Password, Access Groups, Access Rights attributes of the object domain created. If the Domain is of the pre-existent type, this parameter is overlooked.

Result(+)

The service is valid and has been correctly executed by the VMD.

Result(-)

The service is invalid or is not accepted by the VMD. An error code is provided.

6.2.3.3.2.3 Service procedure

The VMD should

- initialize the loading if the domain is Pre-existent on condition that its status has the "predefined" value,
- create an object domain then initialize the loading if the domain is inexistent,
- check the presence of the Domain Access Protection parameter if the request is transmitted via a channel having service quality 2,
- initialize the attributes of the domain as indicated below:
- initialize the attribute identification with the Name given in parameter or the value of the index if the object domain is created,
- initialize the "status" attribute at the "Loading" value,
- the predefined attribute is initialized at "true" if the domain is preexistent. Otherwise it takes the value "false",
- initialize the "Sharable" attribute in compliance with the sharable parameter,
- initialize the Password, Access Group and Access Right attributes of the created domain in compliance with the Domain Access Protection parameter if it is present in the request. If the domain is of the pre-existent type, its Password, Access Group and Access Right attributes remain unchanged.

If all the actions have been executed successfully, a positive report is transmitted. Otherwise a negative report should be transmitted mentioning the reasons for the failure.

6.2.3.3.3 Confirmed download segment service

6.2.3.3.3.1 Functionality

This service allows the Server to ask the Client for a segment of data of the domain.

6.2.3.3.3.2 Service primitives

The structure of the service is shown in Table 48.

Table 48 – Confirmed download segment service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Domain Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
Load Data			M	M (=)
More Follow			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Domain identification:

This parameter specifies the identification by name or by index of an object Domain.

Result(+)

Load data:

This parameter, of the OCTETSTRING type, specifies the data to be loaded.

More follow:

This parameter of the Boolean type, if "true" informs the VMD that data remains to be loaded.

Result(-)

The service is invalid or not accepted by the VMD. An error code is provided.

6.2.3.3.3 Service procedure

The Client should perform the following actions:

– prepare a segment of data of the domain for the "Load data" parameter; the segmentation of the data is a local issue,

– correctly position the value of the "more follow" parameter.

If all the actions are successfully executed, a positive report including the "Load data" and "more follow" parameters is transmitted. If not a negative report should be transmitted explaining the reasons for the failure.

6.2.3.4 Confirmed terminate download sequence service

6.2.3.4.1 Functionality

This service allows the Server:

1 to inform the Client that it is authoritatively stopping the loading because the received data are not consistent,

2 to inform the Client that it has received all the data of the domain and that it has checked their consistency.

6.2.3.4.2 Service primitives

The structure of the service is shown in Table 49.

Table 49 – Confirmed terminate download sequence service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Domain Identification	M	M (=)		
Discard	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Domain identification:

This parameter specifies the identification by name or by index of the Domain.

Discard:

This parameter informs the Client if the loading and consistency of all the data of the domain have been successful, otherwise it indicates the reasons for the failure.

Result(+)

The service is valid and has been correctly executed by the Client.

Result(-)

The service is invalid or has not been accepted by the Client. An error code is provided.

6.2.3.3.4.3 Service procedure

The Server should perform the following actions:

- check that all the data of the domain have been correctly received,
- check the consistency of the data of the domain.

If all the actions have been successfully executed, a request including the parameters "Discard = empty" is transmitted, otherwise a request should be transmitted including the parameter "Discard = reasons for the failure".

In spite of the success of the loading, if the Client answers negatively, the Server should force the status of the domain to the "predefined" value if the latter is of the "pre-existent domain" type. Otherwise it should delete the domain object.

6.2.3.3.5 Confirmed initiate upload sequence service

6.2.3.3.5.1 Functionality

This service allows a Client to upload the data of a domain belonging to a server. This uploading is carried out through an upload object (ULSM) created for this purpose.

6.2.3.3.5.2 Service primitives

The structure of the service is shown in Table 50.

Table 50 – Confirmed initiate upload sequence service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Domain Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
ULSM Index			M	M (=)
List Of Capabilities			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

Domain identification:

This parameter specifies the identification by name or by index of the domain to be saved.

Result(+)

ULSM Index:

This parameter informs the Client of the identification under which the domain saving should take place.

A non-significant value can be transmitted in this parameter to signify that the saving procedure should be performed with identification of the domain.

List of Capabilities:

This parameter generally contains specific information concerning the limitations of implementation. Its content can be empty.

Result(-)

The service is invalid or is not accepted by the VMD. An error code is supplied.

6.2.3.3.5.3 Service procedure

The VMD should perform the following actions:

- check the existence of the domain,
- check that its status has for value "Ready" or "In-Use",
- create an ULSM object and assigning it an identifier called "ULSM index". If the value of the ULSM index is not significant, the continuation of the upload should be done with the identifier forming part of the request.
- prepare the content of the "ULSM index" and "List of capabilities" parameters,
- increment the Upload in Progress attribute of the Domain object.

If all the actions have been executed with success, a positive report including the "ULSM index" and "List of capabilities" parameters is transmitted. Otherwise a negative report should be transmitted including the reasons for the failure.

6.2.3.3.6 Confirmed upload segment service

6.2.3.3.6.1 Functionality

This service allows a client to obtain the content of a Domain.

6.2.3.3.6.2 Service primitives

The structure of the service is shown in Table 51.

Table 51 – Confirmed upload segment service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Upload Identification	M	M (=)		
ULSM Index	S	S		
Domain Identification	S	S		
Result(+) (Comp)			S	S (=)
Load Data			M	M (=)
More Follow			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Upload identification:

This parameter indicates either the identification of the ULSM object of the upload created in the initialization phase or the identification (name or index) of the domain object to be uploaded.

Result(+)

Load Data:

This parameter specifies the data transmitted by the server.

More Follows:

This parameter of the Boolean type has the value "True" if the uploading operation should continue.

Result(-)

The service is invalid or not accepted by the VMD. An error code is provided.

6.2.3.3.6.3 Service procedure

The Sub-MMS Server provides in the response the content of a data segment to be uploaded. The More Follow parameter is initialized at FALSE if there is no more information to be transferred.

6.2.3.3.7 Confirmed terminate upload sequence service

6.2.3.3.7.1 Functionality

This service terminates an uploading sequence.

6.2.3.3.7.2 Service primitives

The structure of the service is shown in Table 52.

Table 52 – Confirmed terminate upload sequence service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Upload Identification	M	M (=)		
ULSM index	S	S (=)		
Domain Identification	S	S (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Upload identification:

This parameter indicates either the identification of the upload USLM object created in the initialization phase or the identification (name or index) of the domain object to be uploaded.

Result(+)

The uploading operation is terminated with success.

Result(-)

The service is invalid or has not been accepted by the VMD. An error code is provided.

6.2.3.3.7.3 Service procedure

The Sub-MMS server deletes the ULSM upload object dedicated to this operation. It decrements the value of the Upload in Progress attribute.

A negative response is transmitted if the Terminate Upload Request service is received whereas the last segment had been transmitted with the More Follow parameter equal to TRUE.

6.2.3.3.8 Confirmed get domain attribute service

6.2.3.3.8.1 Service primitives

The structure of the service is shown in Table 53.

Table 53 – Confirmed get domain attributes service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Domain Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
List of Identification			M	M (=)
Object Class			M	M (=)
List Of Capabilities			M	M (=)
Domain State			M	M (=)
Predefined			M	M (=)
Sharable			M	M (=)
List Of Program Invocation			M	M (=)
Upload in Progress			M	M (=)
Domain Access Protection			C	C (=)
Password			M	M (=)
Access Group			M	M (=)
Access Right			M	M (=)
Extension			U	U (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Domain identification:

This parameter indicates the identification of a domain object (by name or by index) whose attributes should be retrieved in the response.

Result(+)

This parameter indicates that the attribute retrieval operation has been successfully accomplished.

List of Identification:

This parameter gives the identification of the object

Object class:

This parameter indicates the object class

List Of Capabilities:

This parameter indicates the characteristics of the domain.

Domain State:

This parameter indicates the state of the domain.

Predefined:

This parameter indicates if the domain is predefined.

Sharable:

This parameter indicates if the domain can be shared by several PI objects.

List Of Program Invocation:

This parameter indicates the list of PIs sharing the domain.

Upload in Progress:

This parameter indicates if an upload is in progress.

Domain Access Protection:

This parameter is present if the object supports access protection. In this case it comprises the following information:

Password

This parameter indicates the password for the object.

Access Groups

This parameter indicates the groups to which the object belongs.

Access Rights

This parameter indicates the access rights required to have access to the object.

Extension:

This parameter gives additional information.

Result(-)

The service is invalid or has not been accepted by the VMD. An error code is provided.

6.2.3.3.8.2 Service procedure

The Sub-MMS server performs the following operations:

– to check that there is a domain with the same identification,

– read the attributes of the domain.

If all the operations have taken place correctly, the server provides in its positive response the values of the attributes read.

If one of the operations end in a failure, the server provides in the negative response the reasons for the failure.

Whatever the quality of service of the channel via which this service is routed, no access restriction should be applied via the object access protections.

6.2.3.4 Mechanisms

6.2.3.4.1 State charts

Figure 49 shows the Domain management states and state transitions.

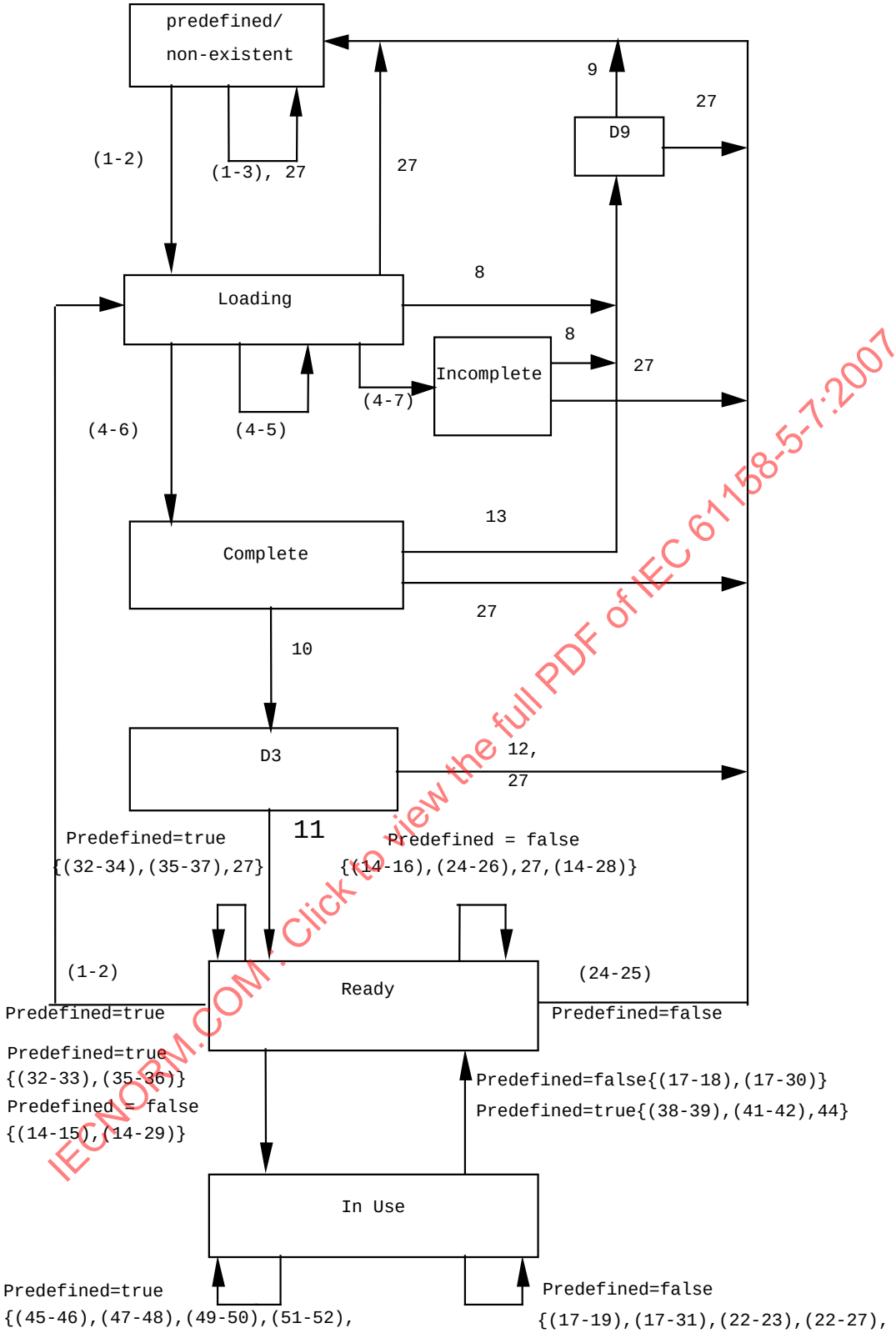


Figure 49 – Domain management state chart

NOTE This state chart is compatible with its MMS and FMS equivalent. The execution of a service indication is done automatically. Consequently only steady states are represented on the diagram. The reading of the attributes of a domain is possible only if the domain is completely loaded. Consequently, only the "ready" or "in-use" states are visible via the network.

The transitions are as follows:

1 - Initiate Download Sequence. Indication

2 - Initiate Download Sequence. Response(+)

3 - Initiate Download Sequence. Response(-)

4 - Download Segment. Request

5 - Download Segment. Confirm(+) More follow=true

6 - Download Segment. Confirm(+) More follow=false

7 - Download Segment. Confirm(-)

8 - Terminate Download Sequence. Request Discard present

9 - Terminate Download Sequence. Confirm(+)or (-)

10 - Terminate Download Sequence. Request Discard not present

11 - Terminate Download Sequence. Confirm(+)

12 - Terminate Download Sequence. Confirm(-)

13 - Terminate Download Sequence. Request Discard present

14 - Create Program Invocation. Indication (program count=0)

15 - Create Program Invocation. Response(+)

16 - Create Program Invocation. Response(-)

17 - Delete Program Invocation. Indication (program count=1)

18 - Delete Program Invocation. Response(+)

19 - Delete Program Invocation. Response(-)

20 - Create Program Invocation. Indication (program count> 0)

21 - Create Program Invocation. Response(+) or (-)

22 - Delete Program Invocation. Indication (program count>1)

23 - Delete Program Invocation. Response (+) or (-)

24 - Delete Domain. Indication

25 - Delete Domain. Response(+)

26 - Delete Domain. Response(-)

27 - Abort. Indication

28 - Abort. Indication (program invocation creation failed)

29 - Abort. Indication (program invocation creation succeeded)

30 - Abort. Indication (program invocation deletion succeeded)

31 - Abort. Indication (program invocation deletion failed).

32 - Start.Indication (Program Count = 0)

33 - Start.Response(+)

34 - Start.Response(-)

35 - Resume.Indication (Program Count = 0)

36 - Resume.Response (+)

37 - Resume.Response (-)

38 - Stop.Indication (Program Count = 1)

39 - Stop.Response (+)

40 - Stop.Response (-)

41 - Kill.Indication (Program Count = 1)

42 - Kill.Response (+)

43 - Kill.Response (-)

44 - Program Stopped or End of Program (Program Count = 1)

45 - Start.Indication (Program Count > 0)

46 - Start.Response (+) or (-)

47 - Resume.Indication (Program Count > 0)

48 - Resume.Response (+) or (-)

49 - Stop.Indication (Program Count > 0)

50 - Stop.Response (+) or (-)

51 - Kill.Indication (Program Count > 0)

52 - Kill.Response (+) or (-)

53 - Program Stopped or End of Program (Program Count > 0).

NOTE This flowchart is compatible with its MMS and FMS equivalent. The service indications for the server are executed automatically. Consequently, only the stable states are included in the flowchart.

Figure 50 shows the Domain upload states and state transitions:

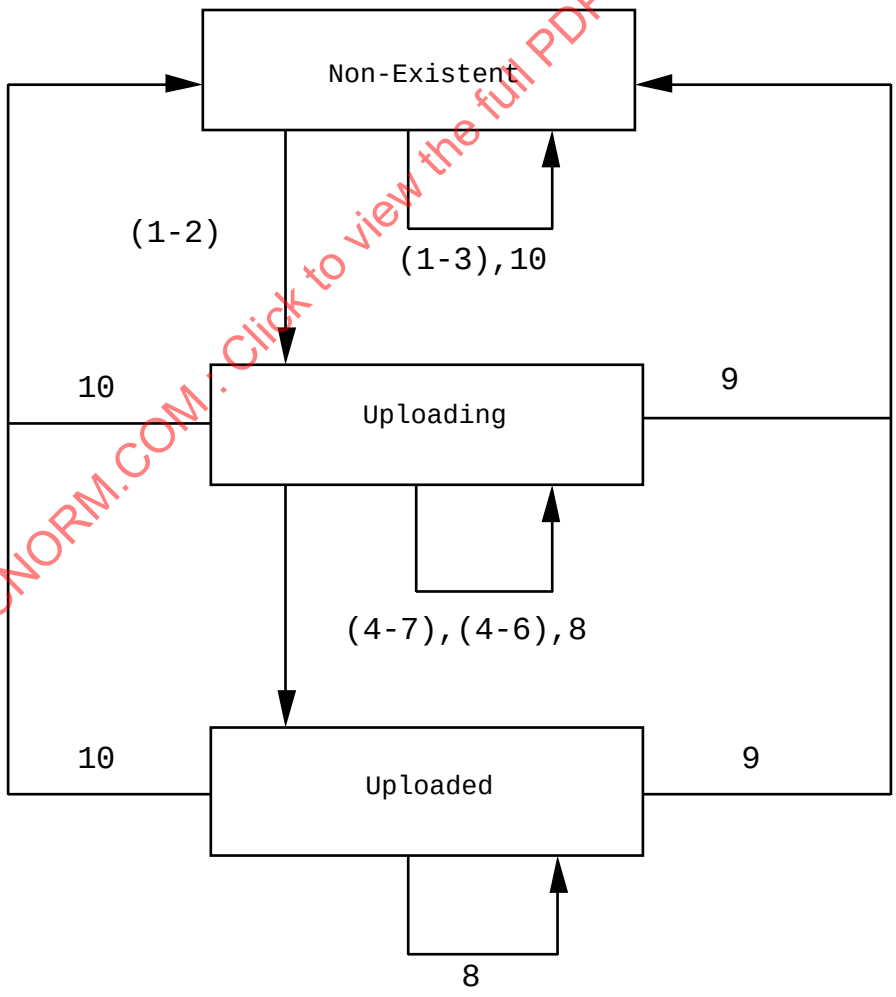


Figure 50 – Domain upload flowchart

Description of the Transitions

1 - Initiate Upload Sequence.Indication

2 - Initiate Upload Sequence.Response(+)

3 - Initiate Upload Sequence.Response(-)

4 - Upload Segment.Indication

5 - Upload Segment.Response(+) more follows = false

6 - Upload Segment.Response(+) more follows = true

7 - Upload Segment.Response(-)

8 - Terminate Upload Sequence.Indication

9 - Terminate Upload Sequence.Response (+) or (-)

10 - Abort.Indication or Abort.Request

6.2.3.4.2 Sequencing of the services

The Initiate Download Sequence, Download Segment and Terminate Download Sequence services should observe the sequence shown in Figure 51.

CLIENT	SERVER
Initiate Download Sequence.Request	---------->
<--------	Initiate Download Sequence.Response
<--------	Download Segment.Request
Download Segment.Response	---------->
••••••••••	••••••••••
••••••••••	••••••••••
<-------	Terminate Download Sequence.Request
Terminate Download Sequence.Response	---------->

Figure 51 – Domain download sequence diagram

The Initiate Upload Sequence, Upload Segment and Terminate Upload Sequence should observe the sequence shown in Figure 52.

CLIENT	SERVER
Initiate Upload Sequence.Request	---------->
<--------	Initiate Upload Sequence.Response
Upload Segment.Request	---------->
<--------	Upload Segment.Response
••••••••••	••••••••••
••••••••••	••••••••••
Terminate Upload Sequence.Request	---------->
<---------	Terminate Upload Sequence.Response

Figure 52 – Domain upload sequence diagram

6.2.4 Program invocation (PI) ASE

6.2.4.1 Overview

Sub-MMS defines an object "Program invocation" which can be created/deleted either by the adequate Sub-MMS services or by a local initiative.

The creation of a PI by the Create PI service and the deleting of a PI by the Delete or Kill services is done atomically (not interruptible by another service)

1 - Create Program Invocation

2 - Delete Program Invocation

3 - Start

4 - Stop

5 - Resume

6 - Reset

7 - Kill

8 - Get Program Invocation Attributes

6.2.4.2 Program invocation class specification

6.2.4.2.1 Program invocation model

FAL ASE: **Program invocation ASE**
CLASS: Program invocation
CLASS ID: not used
PARENT CLASS: not used
Object: Program Invocation
(m) Key Attribute: PI name
(m) key Attribute: PI index
(m) Attribute: PI state (IDLE, STARTING, RUNNING, STOPPING, STOPPED, RESUMING, RESETTING, UNRUNNABLE)
(m) Attribute: List of Domain identification
(m) Attribute: Sub-MMS Deletable (True, false)
(m) Attribute: Reusable(true, false)
(m) Attribute: Execution Argument
(o) Attribute: PI Access Protection
    (m) Attribute: Password
    (m) Attribute: Access Groups
    (m) Attribute: Access Rights
(o) Attribute: Extension

6.2.4.2.2 Attributes

PI Name and index:

These attributes allow unique identification of a Program Invocation object. The types of identifier supported are either a name, or an index or both.

PI State:

This attribute specifies the state of the PI object in compliance with the status diagram described below.

List of Domain Identification:

This attribute specifies the list of Domain objects which compose this program invocation.

Sub-MMS Deletable:

The attribute can take the TRUE or FALSE values. The TRUE value authorizes the Delete PI service to delete the object.

Reusable:

This attribute indicates if the program invocation passes after execution in the IDLE or UNRUNNABLE state. The TRUE value signifies a transition towards the IDLE state and the FALSE value a transition towards the UNRUNNABLE state.

Execution Argument:

This attribute specifies information concerning execution of the program.

PI Access Protection:

This attribute indicates if the object supports the access protection.

Password:

This attribute specifies the value of the password accepted for the access rights.

Access Groups:

This attribute specifies if the object belongs to a group of users, as shown in Table 54. If one of the bits is set to 1, it indicates a group number to which the object belongs.

Table 54 – Access group attribute details for program invocation object

Bit name	Access group number
G0	Access group 8
G1	Access group 7
G2	Access group 6
G3	Access group 5
G4	Access group 4
G5	Access group 3
G6	Access group 2
G7	Access group 1

Access Rights:

This attribute specifies the access rights associated with the Program Invocation object, as shown in Table 55. Each bit set to 1 indicates the acceptance conditions for the services starting, stopping or deleting a program invocation.

Table 55 – Access rights attribute details for program invocation object

Bit name	Significance
S	The Start, Resume, Reset operation is authorized for clients who have provided a password identical to the password attribute of the object
H	The STOP operation is authorized for clients who have provided a password identical to the password attribute of the object
D	PI deletion (Kill, Delete PI) is authorized for clients who have provided a password identical to the password attribute of the object
Sg	The Start, Resume, Reset operation is authorized for clients who belong to one of the groups specified in the Access Groups attribute of the object
Hg	The STOP operation is authorized for clients who belong to one of the groups specified in the Access Groups attribute of the object
Dg	Deletion of the PI (Kill, Delete PI) is authorized for clients belonging to one of the groups specified in the Access Groups attribute of the object
Sa	The Start, Resume, Reset operation is authorized for all the clients
Ha	The STOP operation is authorized for all the clients
Da	Destruction of the PI (Kill, Delete PI) is authorized for all the clients

Extension:

This attribute specifies additional information.

The status chart of a PI is identical to its MMS equivalent. However, in order to make the chart more legible, the intermediate states P1, P2, P3 and P4 concerning the KILL service are not included in the chart. We recall that the Create-PI, Delete-Pi and Kill services should be executed automatically (not interruptible by a service). The intermediate states are never visible via the network and their absence does not change the dynamic behaviour or a PI.

6.2.4.3 Services

6.2.4.3.1 Confirmed create program invocation service

6.2.4.3.1.1 Functionality

The purpose of this service is to allow a client to create, in a VMD, a PI object attached to several Domains.

6.2.4.3.1.2 Service primitives

The structure of the service is shown in Table 56.

Table 56 – Confirmed create program invocation service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Proposed PI Identification	M	M (=)		
List Of Domain Identification	M	M (=)		
Reusable	M	M (=)		
PI Access Protection	C	C (=)		
Password	M	M (=)		
Access Groups	M	M (=)		
Access Rights	M	M (=)		
Extension	U	U (=)		
Result(+) (Comp)			S	S (=)
Final PI Index			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Proposed PI identification:

This parameter specifies the identification proposal per name or per index to be assigned by the VMD to the PI object after creation.

List of Domain Identification:

This parameter specifies the identifications by name or by index of the domains to be attached to the created PI object. The limitation of the number (1 to n) of the domains attached to a local issue.

Reusable:

The "Reusable" parameter should be a Boolean; if its value is true, PI can be executed several times.

PI Access Protection:

This parameter specifies the value of the Password, Access Groups, Access Rights attributes to be assigned to the created PI object.

Its presence is mandatory if the service is transmitted via a channel with the quality of service (2).

Extension:

This parameter gives additional information.

Result(+)

Final PI index:

This parameter allows a server to transmit the definitive index value assigned to the created PI object. This index can take a non-significant value to indicate to the client that the identification proposed by name or by index is accepted.

A PI whose proposed identification is a name could always be identified by this name whatever the value of the Final-PI-index parameter.

Result(-)

This service is invalid or not accepted by the VMD. An error code is provided.

6.2.4.3.1.3 Service procedure

The VMD should

- check that there is no PI with the same identification,
- check that all the domains to be attached to the PI exist and satisfy one of the two conditions: READY state, IN USE state, with the sharable attribute set to TRUE,
- if the request is transmitted via a channel with the service 2 quality, check the presence of PI Access Protection parameter and its compatibility with the password attributes, access groups and access rights of the related domains,
- create a PI object and initialize its attributes as indicated below:
- the identification attribute is initialized with the Name given in parameter or the index value selected (proposed or final) or with both,
- the "state" attribute is initialized to the IDLE value,
- the List of Domain identification attribute is initialized in compliance with the List of Domain Identification parameters,
- the Deletable Sub-MMS attribute is initialized to the TRUE value,
- the "reusable" attribute is initialized in compliance with the Reusable parameter,
- the execution attribute argument is initialized at one empty octet chain,
- the attributes Password, Access Groups and Access Rights attributes are initialized in compliance with the PI Access Protection parameter if it is present in the request,

– Initialize to the value IN USE the state of the domains related to the PI whose value is READY.

If all the actions have been executed with success, a positive report including, if necessary, a final PI index, is transmitted. Otherwise a negative report including the reasons for the failure should be transmitted.

6.2.4.3.2 Confirmed delete program invocation service

6.2.4.3.2.1 Functionality

This service allows deletion of an object of the PI type in a VMD.

6.2.4.3.2.2 Service primitives

The structure of the service is shown in Table 57.

Table 57 – Confirmed delete program invocation service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Program Invocation Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

PI identification:

This parameter specifies the identification by name or by index of a PI object.

Result(+)

The service is valid and has been correctly executed by the VMD.

Result(-)

The service is invalid or not accepted by the VMD. An error code is provided.

6.2.4.3.2.3 Service procedure

The VMD should

- check that there is a PI with the same identification,
- check that the PI is located in one of the following states: IDLE, UNRUNNABLE, STOPPED,
- check that the "Sub-MMS" deletable" attribute is true,
- check the compatibility of the access rights of the channel with the password, access group and access rights attributes of the object if the request is transmitted via a channel with the quality of service 2,
- delete the identification of this PI in the Domain objects composing it,
- delete the PI object in question.

If all the actions are executed successfully, a positive report is transmitted, otherwise a negative report indicating the reasons for the failure should be transmitted.

6.2.4.3.3 Confirmed start service

6.2.4.3.3.1 Functionality

This service allows a Client to activate execution of a PI with a server.

6.2.4.3.3.2 Service primitives

The service structure is shown in Table 58.

Table 58 – Confirmed start service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Program Invocation Identification	M	M (=)		
Execution Argument	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)
Program Invocation State			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

PI identification:

This parameter specifies the identification by name or by index of a PI object.

Execution argument:

The content of the parameter can be empty.The definition of its contents is a local issue.

Result(+)

The service is valid and has been correctly executed by the VMD.

Result(-)

The service is invalid and has not been accepted by the VMD. An error code and the state of the PI are provided.

6.2.4.3.3.3 Service procedure

The VMD should

- check that there is a PI with the same identification,
- check that the PI is in the "Idle" state,
- initialize the Execution Argument attribute of the PI object with the Execution Argument parameter value.

If the request is transmitted via a channel with the quality of service 2, check the compatibility of the channel access rights with the password, access groups and access rights attributes of the object.

- Start the execution of the PI object in question.

If all the actions are executed with success, a positive report is transmitted, otherwise a negative report should be transmitted indicating the reasons for the failure.

6.2.4.3.4 Confirmed stop service

6.2.4.3.4.1 Functionality

This service allows a client to stop execution of a PI at a server.

6.2.4.3.4.2 Service primitives

The structure of the service is shown in Table 59.

Table 59 – Confirmed stop service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Program Invocation Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)
Program Invocation State			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

PI identification:

This parameter specifies the identification by name or by index of a PI object.

Result(+)

The service is valid and has been correctly executed by the VMD.

Result(-)

The service is invalid or has been accepted by the VMD. An error code and the PI state are provided.

6.2.4.3.4.3 Service procedure

The VMD should

– check that there is a PI with the same identification,

– check that the PI is in the RUNNING state, it then transits to the STOPPING state,

– check the compatibility of the access rights of the channel with the password, access groups and access rights attributes of the object if the request is transmitted via a channel having the quality of service 2,

– stop the execution of the PI object in question.

If all the actions are executed with success, a positive report is transmitted and the object PI is placed in the STOPPED state. Otherwise a negative report should be transmitted explaining the reasons for the failure and object PI response to the RUNNING state.

6.2.4.3.5 Confirmed resume service

6.2.4.3.5.1 Functionality

This service allows a Client to resume execution of a PI at a Server from its actual stopping point.

6.2.4.3.5.2 Service primitives

The structure of the service is shown in Table 60.

Table 60 – Confirmed resume service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)		=		
Program Invocation Identification	M	M (=)		
Execution Argument	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)
Program Invocation State			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

PI identification:

This parameter specifies the identification by name or by index of a PI object.

Execution argument:

The content of the parameter can be empty.

Result(+)

The service is valid and has been correctly executed by the VMD.

Result(-)

The service is invalid or has not been accepted by the VMD. An error code and the PI state are provided.

6.2.4.3.5.3 Service procedure

The VMD should

- check that there is PI with the same identification,
- check that the PI is located in the STOPPED state, it then transits to the RESUMING state,
- check the compatibility of the access rights of the channel with the password attributes, access groups and access rights of the object if the request is transmitted via a channel with the quality of service 2,
- resume execution of the PI object in question.

If all the actions are executed with success, a positive report is transmitted and the object PI takes the RUNNING state, otherwise a negative report should be transmitted explaining the reasons for the failure, the object PI can be placed in the UNRUNNABLE or STOPPED state as required.

6.2.4.3.6 Confirmed reset service

6.2.4.3.6.1 Functionality

This service allows resetting a PI at a server in view of a possible start-up on condition that the "Reusable" attribute has the value "TRUE".

6.2.4.3.6.2 Service primitives

The structure of the service is shown in Table 61.

Table 61 – Confirmed reset service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Program Invocation Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)
Program Invocation State			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

PI identification:

This parameter specifies the identification by name or by index of a PI object.

Result(+)

The service is valid and has been correctly executed by the VMD.

Result(-)

The service is invalid or not accepted by the VMD. An error code and the state of the PI are provided.

6.2.4.3.6.3 Service procedure

The VMD should

- check that there is a PI with the same identification,
- check that the PI is in the STOPPED state, it then transits to the RESETTING state,
- check the compatibility of the access rights of the channel with the password, access groups and access rights attributes of the object if the request is transmitted via a channel with the quality of service 2,
- reset the PI object in question.

If all the actions are executed with success, a positive report is transmitted and the object is placed in the IDLE state. Otherwise a negative report should be transmitted explaining the reasons for the failure. The PI can be placed in the UNRUNNABLE or STOPPED state as required.

6.2.4.3.7 Confirmed kill service

6.2.4.3.7.1 Functionality

This service allows forcing a PI at a server to the "Unrunnable" state in view of deletion.

6.2.4.3.7.2 Service primitives

The structure of the service is shown in Table 62.

Table 62 – Confirmed kill service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Program Invocation Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

PI identification:

This parameter specifies the identification by name or by index of a PI object.

Result(+)

The service is valid and has been executed correctly by the VMD.

Result(-)

The service is invalid or has not been accepted by the VMD. An error code is provided.

6.2.4.3.7.3 Service procedure

The VMD should

- check that there is a PI with the same identification,
- check that the PI is located in one of the following states: IDLE, STARTING, RUNNING, STOPPING, RESUMING, STOPPED, RESETTING,
- check the compatibility of the access rights of the channel with the password, access groups and access rights attributes of the object if the request is transmitted via a channel with the quality of service 2,
- force the PI state to the "Unrunnable" value.

If all the actions are executed with success, a positive report is transmitted, otherwise a negative report should be transmitted indicating the reasons for the failure.

Argument

PI identification:

This parameter specifies the identification by name or by index of an object PI.

Result(+)

This selection indicates that the service has been correctly executed by the VMD.

List of identifications:

This parameter gives the identifiers of the PI object.

Object class:

This parameter indicates the class of the object.

Program Invocation State

This parameter indicates the state of the PI.

List Of Domain Identification

This parameter indicates the list of domains attached to the PI.

Sub-MMS Deletable

This parameter indicates if the PI is deletable.

Reusable

This parameter indicates if the PI is rexecutable.

Execution Argument

This parameter indicates the last value supplied as argument either by the Start service or by the Resume service.

PI Access Protection

This parameter is present if the object supports the access protections. In this case it includes the following information:

Password

This information indicates the password of the object.

Access Groups

This information indicates the groups to which the object belongs.

Access Rights

This information indicates the access rights required to obtain access to the object.

Extension:

This parameter gives additional information.

Result(-)

This selection indicates that the service is invalid or not accepted by the VMD.

Error Type

This parameter indicates the reason for the failure.

6.2.4.3.7.4 Service procedure

The VMD should

- check that there is a PI with the same identification,
- read the PI attributes.

If all the actions are successfully executed, a positive report is transmitted, otherwise a negative report should be transmitted explaining the reasons for the failure.

Whatever the quality of the service on the channel by which this service is conveyed, no access restriction should be applied via the object access protections.

6.2.4.4 Mechanisms

Figure 53 shows the Program invocation states and state transitions.

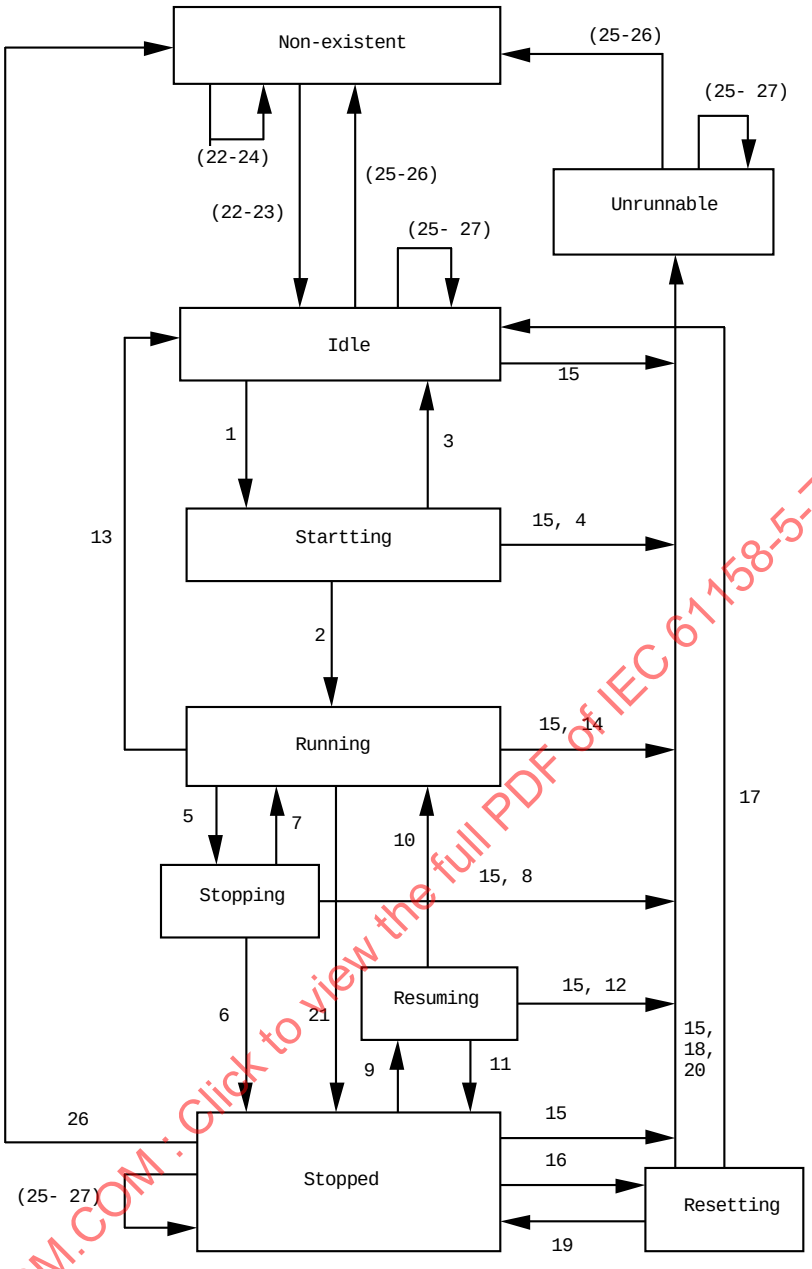


Figure 53 – Program invocation state chart

Description of the transitions:

1 - Start.Indication
2 - Start.Response(+)
3 - Start.Response(-) non-destructive
4 - Start.Response(-) destructive
5 - Stop.Indication
6 - Stop.Response(+)
7 - Stop.Response(-) non-destructive
8 - Stop.Response(-) destructive
9 - Resume.Indication
10 - Resume.Response(+)

11 - Resume.Response(-) non-destructive

12 - Resume.Response(-) destructive

13 - (end of program) & Reusable=true

14 - (end of program) & Reusable=false

15 - Kill.Response(+)

16 - Reset.Indication

17 - Reset.Response(+) Reusable=true

18 - Reset.Response(+) Reusable=false

19 - Reset.Response(-) non-destructive

20 - Reset.Response(-) destructive

21 - (program stopped)

22 - CreateProgramInvocation.Indication

23 - CreateProgramInvocation.Response(+)

24 - CreateProgramInvocation.Response(-)

25 - DeleteProgramInvocation.Indication

26 - DeleteProgramInvocation.Response(+)

27 - DeleteProgramInvocation.Response(-)

6.2.5 Variable ASE

6.2.5.1 Overview

The Sub-MMS defines two classes of objects:

– The Variable object which specifies the access to a simple or structured variable,

– The Variable-List object which specifies access to a series of variable objects.

These two objects can be identified either by name or by index or by both.

Five Sub-MMS services enable manipulating these objects. Total access and partial access are authorized on a Variable object.

The CS can introduce other objects.

The variable management services are as follows:

1 - Read (list of variables or a variable-list)

2 - Write (list of variables or a variable-list)

3 - Information Report (list of variables or a variable-list)

4 - Define Variable-List (a variable-list)

5 - Delete Variable-List (a variable-list)

6 - Get Variable Access Attributes

7 - Get Variable List Attributes

6.2.5.2 Variable class specification

6.2.5.2.1 Variable class model

A variable object is never created or deleted by Sub-MMS services. These operations are local issues.

FAL ASE: **Variable ASE**
CLASS: **Variable**
CLASS ID: not used
PARENT CLASS: not used
Object: Variable-
(m) Key Attribute: Variable Name
(m) key Attribute: Variable Index
(m) Attribute: Type Description
(o) Attribute: Variable Access Protection
(m) Attribute: Password
(m) Attribute: Access groups
(m) Attribute: Access rights
(o) Attribute: Extension

6.2.5.2.2 Attributes

Variable name et index:

These attributes allow a unique identification of variable object. The types of identifier supported are either name, or index or both.

Type Description:

This attribute specifies the abstract type of the variable. The authorized types are those defined by "Type Data Specification" (except the Error Type) (See 6.2.5.2).

Variable Access Protection:

This attribute specifies if the object supports the access protection.

Password:

This attribute specifies the value of the password accepted for the access rights.

Access Groups:

This attribute specifies if the object belongs to a group of users, as shown in Table 63. If one of the bits is set to 1, it indicates the number of a related group.

Table 63 – Access group attribute details for variable object

Bit name	access group number
G0	access group 8
G1	access group 7
G2	access group 6
G3	access group 5
G4	access group 4
G5	access group 3
G6	access group 2
G7	access group 1

Access Rights:

This attribute specifies the access rights associated with the variable object, as shown in Table 64. Each bit set to 1 indicates the acceptable conditions of a read or write service.

Table 64 – Access rights attribute details for variable object

Bit name	Significance
R	Reading authorized to clients who have provided a password identical to the Password attribute of the object
W	Writing authorized to clients who have provided a password identical to the Password attribute of the object.
Rg	Reading authorized to clients belonging to one of the groups specified in the Access Groups attribute of the object.
Wg	Writing authorized to the clients belonging to one of the groups specified in the Access Groups attribute of the object
Ra	Reading authorized for all the clients
wa	Writing authorized for all the clients

Extension:

This attribute specifies additional information.

6.2.5.2.3 Variable-list class specification

A Variable-List object allows access to a list of objects of the Variable class. Each element of the list is a variable which can be identified either by a name or by an index. The partial access to a variable is not authorized in the definition of an object of the Variable list class.

FAL ASE: **Variable ASE**
CLASS: Variable list
CLASS ID: not used
PARENT CLASS: not used
(m) Key Attribute: Variable-List Name
(m) Key attribute: Variable-List index
(m) Attribute: Sub-MMS Deletable (True,false)
(m) Attribute: List of Variable Identification
(o) Attribute: Variable-List Access Protection
    (m) Attribute: Password
    (m) Attribute: Access groups
    (m) Attribute: Access rights
(o) Attribute: Extension

6.2.5.2.4 Attributes

Variable list Name and index:

These attributes allow unique identification of a variable-list object by using either a name or an index, or both.

Sub-MMS Deletable:

This attribute can take true or false values. The true value authorizes the Delete Variable-List service to delete the object.

List of Variable Identification:

This attribute identifies all the variable objects which compose the object Variable-List. Each object can be identified either by a name or by an index or both.

Variable-List Access Protection

This attribute specifies if the object supports the access protection.

Password:

This attribute specifies the value of the password accepted for the access rights.

Access Groups:

This attribute specifies if the object belongs to a group of users, as shown in Table 65. If one of the bits is set to 1, it indicates the number of group to which it belongs.

Table 65 – Access group attribute details for variable list object

Bit name	access group number
G0	access group 8
G1	access group 7
G2	access group 6
G3	access group 5
G4	access group 4
G5	access group 3
G6	access group 2
G7	access group 1

Access Rights:

This attribute specifies the access rights associated with the variable-list object, as shown in Table 66. Each bit set to 1 indicates the acceptance condition of a read, write or delete service of a variable-list object.

Table 66 – Access right attribute details for variable list objects

Bit name	Significance
R	Reading authorized for clients who have provided a password identical to the Password attribute of the object
W	Writing authorized for clients who have provided a password identical to the Password attribute of the object
D	Deletion authorized for clients who have provided a password identical to the Password attribute of the object
Rg	Reading authorized for clients belonging to one of the groups specified in the Access Groups attribute of the object.
Wg	Writing authorized for clients belonging to one of the groups specified in the Access Groups attribute of the object
Dg	Deletion authorized for clients belonging to one of the groups specified in the Access Groups attribute of the object
Ra	Reading authorized for all the clients
wa	Writing authorized for all the clients
Da	Deletion authorized for all the clients

Extension:

This attribute specifies additional information.

6.2.5.3 Services

6.2.5.3.1 Confirmed read service

6.2.5.3.1.1 Functionality

This service is used by a client to request from a VMD either the global/partial reading of one or several variables or the reading of a variable-list.

The limitation by the server of the number of variables to be read in one service invocation (1 to n) is a local issue.

6.2.5.3.1.2 Service primitives

The structure of the service is shown in Table 67.

Table 67 – Confirmed read service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Type With result	M	M (=)		
Variable Access Specification	M	M (=)		
Result(+) (Comp)			S	S (=)
List Of Access Result			M	M (=)
List Of Type Data Specification			C	C (=)
List Of Data Value			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Type With result:

This parameter indicates if the types of variables read should be supplied in the positive response.

Variable Access Specification:

This parameter describes the identification of several variables, elements of variables or of one variable-list to be accessed.

Result(+)

This selection specifies that at least one reading of variable has been carried out with success.

list of Access Results

The list of access results parameter specifies the lists of failures/successes of all the variables read, observing the order given by the Variable Access Specification Parameter. In case of reading of a variable list object, the order of the results is consistent with the description of that object.

List of Type Data Specification:

The content of this parameter is empty if the parameter Type With result of the request is equal to false.

The parameter List of Type Data Specification specifies the descriptors of the type of variables read with success or the descriptor of the ErrorType for the variables read with failure. This information is supplied observing the order given in the Variable Access Specification parameter. In case the variables are of the same type, the list can be reduced to a single description. The authorized types are thus those defined by "Type Data Specification" (See 6.2.5.2).

List Of Data Values:

The List of Data Values parameter specifies the data corresponding to the values of the variables read with success and the error data of the ErrorType for the variables read with failure. This information is supplied according to the order in the Variable Access Specification parameter or in the description of the Variable-List object. (See 6.2.5.2.3 for the authorized variable values).

Result(-)

The service is invalid or not accepted by the VMD. An error code is formed.

6.2.5.3.1.3 Service procedure

For each variable read, the VMD provides a data and a report of the operation, observing the order indicated in the request. It analyses the value of the parameter Type With Result to determine whether it should provide in the response, the description of the type of each variable read.

For a reading operation performed on a channel using access protections (quality of service 2) the VMD should check before each operation of elementary reading, the consistency of the access rights.

For each variable read with success, the VMD provides a positive indicator in the List of Access result, its descriptor type, if necessary, in List of Type data Specification and its value in List of Data Values.

For each variable read unsuccessfully, the VMD provides an error indicator in the List of Access Results, the descriptor of the Error Type in the List of Type data Specification and the value of the error in the list of data values.

6.2.5.3.2 Confirmed write service

6.2.5.3.2.1 Functionality

This service is used by a client to write the global or partial value of one or several variables or the value of a Variable List object.

The limitation by the server of the variable numbers (1 to n) to be written in a service invocation is a local issue.

6.2.5.3.2.2 Service primitives

The structure of the service is shown in Table 68.

Table 68 – Confirmed write service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Variable Access Specification	M	M (=)		
List Of Type Data Specification	U	U (=)		
List Of Data Value	M	M (=)		
Result(+) (Comp)			S	S (=)
List Of Access Result			M	M (=)
List of Error Type			C	C (=)
Result(-)			S	S (=)
Error Type			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

Variable Access Specification:

This parameter describes the identification of several variables, elements of variables or of a Variable-List object to be written.

List Of Data specifications:

The parameter List of Data Specification specifies the descriptors of the type of variables to be written, observing the order in the Variable Access Specification parameter. Its content can be empty when the variables are implicitly known. In case the variables are of the same type, the list can be reduced to a single description. The authorized types are those defined by "Type Data Specification" (except the Error Type) (See 6.2.5.2).

List Of Data Values:

The List of Data Value parameters specifies the values of the variables to be written, observing the order in the Variable Access Specification parameter or in the description of the variable list object. The authorized values are those defined by "Data Value" (See 6.2.5.2.3).

Result(+)

This selection specifies that at least one writing has been carried out with success.

List Of Access Result:

The List of Access Result parameter specifies the write operation reports of the variables, observing the order indicated in the Variable Access Specification parameter.

List Of Error Type:

The parameter List Of Error Type specifies a value of the Error type for each writing of variable ending in a failure, observing the order indicated in the Variable Access Specification parameter.

Result(-)

The service is invalid or not accepted by the VMD. An error code is provided.

6.2.5.3.2.3 Service procedure

The VMD checks the consistency between the variables described in the Variable Access specification and the List of Type Data Specification parameter, if it is provided. In case of inconsistency on the type or the number of variables, a negative result is returned. If there is consistency, the VMD writes the values in each specified variable.

On a channel using the access protections, the VMD should check before each elementary write operation the consistency of the access rights.

A report of the operation is provided in the List of Access Result parameter for each elementary write operation. An error code is given necessarily in the List of Error type parameter for each write failure or insufficiency of access rights.

6.2.5.3.3 Unconfirmed information report service

6.2.5.3.3.1 Functionality

This service is used by a Sub-MMS server to transmit either the global/partial values of one or several variables or the values of a Variable-List object.

The limitation by the server of the number of variables (1 to n) to be transmitted in one service invocation is a local issue.

This service is not confirmed.

6.2.5.3.3.2 Service primitives

The structure of the service is shown in Table 69.

Table 69 – Unconfirmed information report service parameters

Parameter Name	Req	Ind
Argument (comp)		
Local Detail	U	U (=)
Variable Access Specification	M	M (=)
List Of Type Data Specification	U	U (=)
List Of Data Value	M	M (=)

Argument

Local Detail:

This parameter of the OctetString type can contain information dependent on implementations. It can be used to transport a priority level.

Its content can be empty if no specific information is transmitted.

Variable Access Specification:

This parameter describes the identification of the variables, elements of variables or Variable-list object whose values are transmitted.

List of Type Data Specification:

This parameter specifies the descriptors of the types of variables respecting the order given in the Variable Access Specification parameter. Its content can be empty when the variables are implicitly known. In case the variables are of the same type, the list can be reduced to a single description. The authorized types are those defined by "Type Data Specification" (except that of "Error Type") (See 6.2.5.2).

List Of Data Value:

This parameter specifies the values of the variables, observing the order of the variables included in the Variable Access Specification parameter. The authorized values are those defined by "Data Value" (See 6.2.5.2.3).

6.2.5.3.3.3 Service procedure

No service procedure is specified by Sub-MMS.

6.2.5.3.4 Confirmed define variable list service

6.2.5.3.4.1 Functionality

The purpose of this service is to allow a client to create in a VMD a Variable-List object and attach to it a list of variable class objects.

Partial access to the attached variables is not possible.

The limitation by the server of the number of variables to be attached (1 to n) to a Variable-List class object is a local issue.

6.2.5.3.4.2 Service primitives

The structure of the service is shown in Table 70.

Table 70 – Confirmed define variable-list service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Proposed Variable-list Identification	M	M (=)		
List of Variable Identification	M	M (=)		
Variable-list Access protection	C	C (=)		
Password	M	M		
Access Groups	M	M		
Access Rights	M	M		
Extension	U	U (=)		
Result(+) (Comp)			S	S (=)
Final Variable-list Index			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Proposed Variable List identification:

This parameter specifies the proposal for identification by name or by index to be assigned by the VMD to the object variable-list after creation.

List of variable Identification:

This parameter specifies the identifications by name or by index of the variables linked to the object variable-list created. It also fixes the order of the variables composing the list.

Variable List Access Protection:

This parameter specifies the value of the Password, Access Groups, Access Rights attributes to be assigned to the object variable-list created.

It is compulsory if the service is performed on a channel with the quality of service (2). Otherwise it is empty.

Extension:

This parameter gives additional information.

Result(+)

Final Variable-List index:

This parameter allows the server to transmit the value of the final index assigned to the created variable-list object. This index can take a non-significant value to inform the client that the identification proposed by name or by index is accepted.

A variable whose proposed identification is a name, could always be identified by this name whatever the value of the Final-VL-index parameter.

Result(-)

The service is invalid or not accepted by the VMD. An error code is provided.

6.2.5.3.4.3 Service procedure

If the parameters of the request are acceptable, an object variable-list is created and initialized.

The identification attribute is initialized with the Name given in parameter, or the index value selected (proposed or final) or with both.

The Sub-MMS Deletable attribute is assigned to the True value.

The values of the identifications of the variables composing the list are initialized from the List of Variable specification parameter.

On a channel using the quality of service 2, the Password, Access Rights and Access Group attributes are initialized with the value of the parameters.

After this initialization, a positive report is transmitted;

In a context using the quality of service with access protection, the absence of the Password parameters, access rights and access groups results in transmission of a negative result. Besides, the creation of the variable-List object is conditioned by the values of the access rights of each of the list variables which should be compatible with the access rights proposed by the service.

6.2.5.3.5 Confirmed delete variable-list service

6.2.5.3.5.1 Functionality

This service allow deletion in a VMD of a single object of the variable-list type.

6.2.5.3.5.2 Service primitives

The structure of the service is shown in Table 71.

Table 71 – Confirmed delete variable-list service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Variable-list Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
Result(-)			S	S (=)
Error Type			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

Variable-list identification:

This parameter specifies the identification by name or by index of a Variable-list object to be deleted.

Result(-)

The service is invalid or not accepted by the VMD. An error code is provided.

Result(+)

The service is valid and has been correctly executed by the VMD.

6.2.5.3.5.3 Service procedure

If the identification of the Variable-list object corresponds to an existing Variable-list object and the Sub-MMS deletable attribute is true, the delete operation is performed and a positive response is transmitted.

On a channel with access protection, the access protection parameters provided during the opening of the association (negotiated or prenegotiated) are analyzed with respect to the values of the password, access rights and Access groups attributes of the object. If there is inconsistency, the operation is not performed and a negative report is transmitted.

6.2.5.3.6 Confirmed get variable access attributes service

6.2.5.3.6.1 Functionality

This service allows a client to read, in a VMD, the attributes of a Variable object.

6.2.5.3.6.2 Service primitives

The structure of the service is shown in Table 72.

Table 72 – Confirmed get variable access attributes service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Variable Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
List of identifications			M	M (=)
Object class			M	M (=)
Sub-MMS deletable			M	M (=)
Type description			M	M (=)
Variable Access Protection			C	C (=)
Password			M	M (=)
Access Groups			M	M (=)
Access Rights			M	M (=)
Extension			U	U (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Variable Identification:

This parameter specifies the identification of the object whose attributes are to be restored in the response.

Result(+)

This selection indicates that the service was correctly executed.

List of identification:

This parameter gives the identifiers of the variable object.

Object class:

This parameter indicates the class of the object.

Sub-MMS deletable

This parameter indicates if the object is deletable.

Type Description

This parameter indicates the type of variable. The types permitted are those defined by "Type Data Specification" (except Error Type) (See 6.2.5.2).

Variable Access Protection

This parameter is present if the object supports the access protections. In this case it includes the following information:

Password

This information indicates the object password.

Access Groups

This information indicates the groups to which the object belongs.

Access Rights

This information indicates the access rights required for access to the object.

Extension:

This parameter gives additional information

Result(-)

The service is invalid or not accepted by the VMD. An error code is supplied.

6.2.5.3.6.3 Service procedure

The server should perform the following actions:

- check that there is a Variable object,
- read the object attributes.

If all the actions are performed correctly, a positive response is transmitted.

If one of the actions is unsuccessful, a positive response is transmitted.

Whatever the quality of service of the channel via which this service is transmitted, no access restriction will be applied by means of object access protection.

6.2.5.3.7 Confirmed get variable-list attributes service

6.2.5.3.7.1 Functionality

This service allows a client to read in a VMD the attributes of a variable-List object.

6.2.5.3.7.2 Service primitives

The structure of the service is shown in Table 73.

Table 73 – Confirmed get variable-list attributes service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Variable-list Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
List of identification			M	M (=)
Object class			M	M (=)
Sub-MMS deletable			M	M (=)
List of Variable Identification			M	M (=)
Variable List Access Protection			C	C (=)
Password			M	M (=)
Access Group			M	M (=)
Access Rights			M	M (=)
Extension			U	U (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Variable-list identification:

This parameter indicates the variable-list object whose attributes are to be retrieved.

Result(-)

The service is invalid or not accepted by the VMD. An error code is supplied.

Result(+)

The service is valid and has been correctly executed by the VMD.

List of identification:

This parameter ives the identifiers of the variable list object.

Object class:

This parameter indicates the class of the object.

Sub-MMS deletable

This parameter indicates if it is true that the object is destructible.

List of Variable Identification

This parameter indicates the variables attached to the Variable-List object.

Variable-List Access Protection

This parameter is present if the object supports the access protections. In this case it includes the following information:

Password

This information indicates the password of the object.

Access Groups

This information indicates the groups to which the object belongs.

Access Rights

This information indicates the access rights required for access to the object.

Extension:

This parameter gives additional information.

6.2.5.3.7.3 Service procedure

The server should perform the following action:

– check that there is a variable-list object,
– read the attributes of the object.

If all the actions have been correctly executed, a positive response is sent.

If one of the actions is unsuccessful, a negative response is sent.

Whatever the quality of the service of the channel via which this service is conveyed, no access restriction should be applied through the object access protection.

6.2.5.4 Additional concepts

6.2.5.4.1 Definition of the data specification type

The various descriptors of the type of data for variables are given by the above list. The CS and the sub-MMS users can define other types. See Table 74.

Table 74 – Data type specification

Parameter name	Req	Ind	CBB
Type Data Specification	M	M (=)	
Array type	S	S (=)	arr
Number Of element	M	M (=)	
Type Data Specification	M	M (=)	
Structure type	S	S (=)	str
List Of Type Data Specification	M	M (=)	
Simple Boolean type	S	S (=)	
Boolean length	M	M (=)	
Simple Integer type	S	S (=)	
Integer length	M	M (=)	
Simple Unsigned type	S	S (=)	
Unsigned length	M	M (=)	
Simple Bitstring type	S	S (=)	
Bitstring length	M	M (=)	
Simple Boolean array type	S	S (=)	
Boolean Array length	M	M (=)	
Simple Visible type	S	S (=)	
Visible string length	M	M (=)	
Simple Octet String type	S	S (=)	
Octetstring length	M	M (=)	
Simple Time difference type	S	S (=)	
Time difference length	M	M (=)	
Simple Time of day type	S	S (=)	
Time of day length	M	M (=)	
Simple single floating type	S	S (=)	
Single floating length	M	M (=)	
Simple Double floating type	S	S (=)	
Double floating length	M	M (=)	
Simple BCD type	S	S (=)	
BCD length	M	M (=)	
Simple Error type	S	S (=)	
Error length	M	M (=)	

The CBB (Conformance Building Block) parameter shows which are the types of variables negotiable at the opening of an association.

1 - Array type is the description of the type of a table of elements of the same type.

2 - Structure type is the description of the type of a structure of which each field can be of a distinct type.

3 - Boolean type is the description of the type of a Boolean.

4 - Integer type is the description of a type of an Integer.

5 - Unsigned type is the description of the type of an integer whose values are positive.

6 - Bitstring type is the description of the type of bit string. The first bit of the string is positioned at bit 2**7 of the first octet. The second bit is position at bit 2**6 of the first octet, and so on.

7 - Boolean Array is the description of a table of bits. The first element of the table is positioned at bit 2**0 of the first octet. The second is positioned at bit 2**1 of the first octet, and so on.

8 - Visiblestring type is the description of the type of a string of octets whose value is an ASCII character to be taken from {"A" to "Z", "a" to "z", "0" to "9", ".", "_"}.

9 - Octetstring type is the description of the type of a string of octets.

10 - Time difference type is the description of the Time difference type(Unsigned-32 expressing in milliseconds the time elapsed since midnight).

11 - Time of day type is la description of the Time Of Day type

12 - Single floating type is the description of the Single floating point (format IEC 60559).

13 - Double floating type is the description of the Double floating point (format IEC 60559).

14 - BCD type is the description of the type in digit string.

15 - Error type is the description of the Error type.

6.2.5.4.2 Definition of the list of access results

The access result list is a series of indicators indicating the series of successes/failures relative to the access of several variables. This parameter should be a BITSTRING. A bit set to TRUE (value 1) signifies access with success.

6.2.5.4.3 Variable access specification

6.2.5.4.3.1 Variable access specification parameter

The structure of the Variable Access Specification parameter is given in Table 75.

Table 75 – Variable access specification

Parameter name: Variable Access Specification	Req Rsp	Ind Cnf
Kind of Variable	M	M (=)
Variable-List	S	S (=)
Identification	M	M (=)
List Of Variable	S	S (=)
variable Access description	M	M (=)

Kind of variable:

This parameter specifies an access to a series of objects of the Variable class with the possibility of obtaining partial access or access to a single object of the Variable-List class previously defined.

Variable List:

This parameter specifies the identification by name or by index of an object variable list.

List Of variables:

This parameter gives an enumeration of the access descriptions to variable class objects.

6.2.5.4.3.2 Variable access description parameter

The structure of the Variable Access description parameter is given in Table 76.

Table 76 – Variable access description attribute details

Parameter name: Variable Access description	Req Rsp	Ind Cnf
Kind of Access	M	M (=)
Global Access	S	S (=)
Identification	M	M (=)
Partial Access	S	S (=)
Identification	M	M (=)
Path Selection	M	M (=)

Kind of Access:

This parameter indicates if the variable is accessed globally or partially.

Global Access:

This parameter exclusively specifies the name or the index of the variable to be accessed globally.

Partial Access:

This parameter describes the identification of the variable object to be accessed partially followed by the description of the path to be followed to reach one or several elements of the variable.

6.2.5.4.3.3 Path selection parameter

The structure of the path selection parameter is given in Table 77.

Table 77 – Path selection parameters

Parameter name: Path Selection	Req Rsp	Ind Cnf	CBB
Kind of selection	M	M (=)	
Select Access	S	S (=)	
Kind of access	M	M (=)	
(1)st level nesting access	S	S (=)	
Access selection	M	M (=)	
Component index	S	S (=)	Str
Element index	S	S (=)	Arr
(n≥1)th level nesting access	S	S (=)	
Access selection	M	M (=)	
List of components	S	S (=)	Str
Component index	M	M (=)	
List of Elements	S	S (=)	Arr
Element index	M	M (=)	
Range elements	S	S (=)	Arr
start element index	M	M (=)	
Number of elements	M	M (=)	
Select Alternate Access	S	S (=)	
Access selection	M	M (=)	
Component index	S	S (=)	Str
Element index	S	S (=)	Arr
Path selection	M	M (=)	

6.2.6 Event ASE

6.2.6.1 Overview

The objects, the services and the mechanisms of events should be capable of adapting themselves to the modest needs of sensors/actuators as well as to the greater demands of programmable controllers/work stations.

1- Sub-MMS standardizes the following actions:

a) Acknowledging an event resulting from a particular event object; this action uses the Acknowledge Event Notification service. The acknowledgement only has a sense for an event previously notified by the Event Notification service.

b) Validating/invalidating an event object; this action makes use of the Alter Event Condition Monitoring service. A valid object can generate events to be notified. An invalid object cannot generate events. The notification of events (resulting from several Event objects of the same type) uses the Event Notification service.

2- Sub-MMS authorizes the definition of specific actions by the users of Sub-MMS and/or by the Companion Standards. These actions use the Alter Event Condition Monitoring service.

3- Sub-MMS introduces the notion of the event detection mechanism common to several objects.

This detection takes place periodically during a period of time sufficiently short to consider that all these events have been detected at the same instant.

In this case a value which we should call "Common Occurrence ID" is assigned to the "Event Occurrence ID" attribute (if supported) for all the objects for which an event was detected during this time interval.

The events thus detected resulting from objects of the same type are notified to clients in the same invocation of the "Event Notification" service with the value of the "Event Transaction Occurrence ID" parameter equal to "Common Occurrence ID". To save space, each of these events should be represented by the truncated "Event Message" attribute of the "Event Occurrence ID" attribute (if it is supported) since its value is identical to the value of the Event Transaction Occurrence ID.

To meet the requirements of more or less complex event objects, Sub-MMS defines a configurable event object.

This configuration allows the creation of the following types of event objects:

1- Object not supporting any standard action or action specific to the user/C.S. In this case, the "Event Message" attribute is a structure with only one "Event Data" attribute. This object can collaborate in the detection of events common to several objects.

2- Object only supporting the acknowledgement action. In this case the "Event Message" attribute is a structure including three attributes: "Event State", "Event Occurrence ID" and "Event Data". This object can collaborate in the detection of events common to several objects.

3- Object only supporting the validation/invalidation action. In this case, the "Event Message" attribute is a structure with two attributes, "Event State" and "Event Data". This object can collaborate in the detection of events common to several objects.

4- Object supporting the acknowledgement and the validation/invalidation. In this case the "Event Message" attribute is a structure with three attributes, "Event State", "Event Occurrence ID" and "Event Data".

6.2.6.2 Event class specification

6.2.6.2.1 Event class model

FAL ASE:		**Event ASE**	
CLASS:		Event	
CLASS ID:		not used	
PARENT CLASS:		not used	
Object: Event			
(m)	Key Attribute:	Name	
(m)	Key Attribute:	Index	
(m)	Attribute:	Event type	
	(m)	Attribute:	Common event detection support(true, false)

	(m)	Attribute:	Acknowledge action support(true, false)
	(m)	Attribute:	Enable action support(true, false)
	(o)	Attribute:	Extension action support
(m)	Attribute:	Event_Message	
	(c)	Attribute:	Event State
		(m) Attribute:	ack action state(acknowledged, not acknowledged)
		(m) Attribute:	enable action state(validated invalidated)
		(o) Attribute:	extension actions states
	(c)	Attribute:	Event Occurrence ID
	(m)	Attribute:	Event data
(o)	Attribute:	Event Access Protection Support	
	(m)	Attribute:	Password
	(m)	Attribute:	Access Group
	(m)	Attribute:	Access Rights
(o)	Attribute:	Extension	

6.2.6.2.2 Attributes

Key attributes

These attributes indicate if the object supports the identification by Name, Index, or Name and Index at the same time.

Event type

This attribute, of the BITSTRING-16 type, characterizes the object type. It indicates if the object collaborates with other Event-Objects with a common event detection mechanism. It also indicates if the object supports specific actions affecting the "Event State" attribute:

Common event detection support:

It indicates if it is true that the object collaborates with other objects for the same event detection mechanism.

Acknowledge action support:

This attribute indicates if the object supports the action of acknowledgement by using the Acknowledge Event Notification service.

Enable action support:

This attribute indicates if the object supports the "validation/invalidation" action using the Alter Event Condition Monitoring service.

Extension action support:

This attribute indicates if the object supports the specific actions, which are carried out by the Alter Event Condition Monitoring service. These actions could be in conformity with the C.S as well as specific to a user.

Event message

This attribute specifies the dynamic information of the object.

Event state

This attribute, of the BITSTRING-16 type, is present if the object supports the actions. It indicates the last remote actions having acted on this object by the acknowledgement of the Acknowledge Event Notification and/or Alter Event Condition Monitoring services:

a) the "Acknowledge action state" attribute stores the acknowledgement,

b) the "Enable action state" attribute stores the action "validation/invalidation",

c) the "Extension actions states" attribute stores the eventual extension actions.

Event occurrence ID

This attribute indicates the identification of the occurrence of an event specific to the object. The attribute is present if the acknowledgement is supported by the object.

The OCTETSTRING-2 type is permitted; the semantic of its contents should be described in the PICS.

Event data

This attribute indicates the value associated to the object Event. The types permitted are those defined by "Data Value" (See 6.2.5.2.3).

Event access protection support

This attribute if present specifies the information relative to the protection of the object.

Password

This attribute specifies the value of the accepted password for the access rights.

Access group

This attribute specifies if the object belongs to a group of users, as shown in Table 78. If one of the bits is positioned at 1, it indicates an access group number.

Table 78 – Access group attribute detail for event object

bit name	number of access group
G0	Access group 8
G1	Access group 7
G2	Access group 6
G3	Access group 5
G4	Access group 4
G5	Access group 3
G6	Access group 2
G7	Access group 1

Access rights

This attribute specifies the access rights associated with the Event object, as shown in Table 79. Each bit positioned at 1 indicates the acceptance conditions of the Acknowledge Event Notification (acknowledgement) and Alter Event Condition Monitoring (validation/invalidation) services.

Table 79 – Access rights attribute details for event object

bit name	Signification
W	Acknowledgement and event reading (by the Acknowledge Event Notification, Get Alarm Summary services) are permitted to the clients having provided a password identical to the object Password attribute
D	Validation/Invalidation and event reading (by the Alter Event Condition Monitoring, Get Alarm Summary services) are permitted to clients having provided a password identical to the object Password attribute
Wg	Acknowledgement and event reading (by the Acknowledge Event Notification, Get Alarm Summary services) are permitted to clients belonging to one of the specified groups in the object Access Groups attribute
Dg	Validation/Invalidation and event reading (by the Alter Event Condition Monitoring, Get Alarm Summary services) are permitted to clients belonging to one of the specified groups in the object Access Groups attribute
Wa	Acknowledgement and event reading (by the Acknowledge Event Notification, Get Alarm Summary services) are permitted to all the clients
Da	Validation/Invalidation and event reading (by the Alter Event Condition Monitoring, Get Alarm Summary services) permitted to all the clients

Extension:

This attribute specifies additional information.

The event management services are

1 - Event Notification; this service enables notifying the events.

2 - Acknowledge Event Notification; this service enables event acknowledgement.

2 - Alter Event Condition Monitoring; this service enables altering the event status.

3 - Get Alarm Summary; this service enables reading the information associated with one or more events.

4 - Get Event Condition Attribute: this service enables obtaining the values of certain attributes of an object Event.

6.2.6.3 Services

6.2.6.3.1 Unconfirmed event notification service

6.2.6.3.1.1 Functionality

This service allows the server to notify one or more events. Each even is represented by the Event_Message attribute of an object.

The limitation by the server of the number of events to be notified (0 to n) in one service invocation is a local issue.

When and how this service is used is a local issue.

6.2.6.3.1.2 Service primitives

The structure of the service is shown in Table 80 and Table 81.

Table 80 – Unconfirmed event notification service parameters

Parameter name	Req	Ind
Argument (Comp)		
Common Information	M	M (=)
Event Type	M	M (=)
Event Transaction Occurrence ID	M	M (=)
User Extension	U	U (=)
List of Event Identification	M	M (=)
List of Event_Message Type description	U	U (=)
Event State type description	C	C (=)
Event Occurrence ID type description	C	C (=)
Event Data type description	M	M (=)
List of Event_Message value	M	M (=)
Event State value	C	C (=)
Event Occurrence ID value	C	C (=)
Event Data value	M	M (=)

Table 81 – Event type parameter details

Parameter name: Event Type	Req	Ind
Common event detection support	M	M (=)
Acknowledge action support	M	M (=)
Enable action support	M	M (=)
Extension action support	U	U (=)

Argument

Common Information

This parameter, of the OCTETSTRING type, specifies the information relative to the objects in the indicated order:

a) "Event Type" indicates a type common to all the object events. This type is characterized by the following information:
   - "Commun event detection support",
   - "Acknowledge action support",
   - "Enable action support",
   - "Extension action support".

b) "Event Transaction Occurrence ID" of OCTETSTRING-16 type, indicates the identification of the transaction occurrence.

c) User-Extension specifies the information known to the user.

List Of Event Identification

This parameter gives the object list where an Event_Message appears in the service.

List Of Event_Message Type Description

The content of this parameter can be empty.

This optional parameter, when not empty, specifies the descriptor of the Event_Message type for each object identified in the service. It is possible to reduce the list of descriptors to a single descriptor when they are identical. Each descriptor of the structure type specifies the following information in the indicated order:

– the first item of information indicates the "Event State" type; this information is present only if the object supports actions;

– the second item of information indicates the "Event Occurrence ID" type; this event is present only if the "Common event detection support" parameter is false and the "Acknowledge action support" is true (objects supporting acknowledgement but not collaborating with a common event detection mechanism);

– the third item of information indicates the "Event Data" type; this information is always present.

The types permitted are those defined by the type Data Specification (except Error Type). (See 6.2.5.2).

List Of Event Message Value

This parameter specifies the Event_Message value of each object identified in the service. Each value of the structure type specifies the following information in the indicated order:

– the first item of information indicates the "Event State" value; this information is present only if the object supports actions;

– the second item of information indicates the "Event Occurrence ID" value; this information is present only if the "common event detection support" parameter is false and the "Acknowledge action support" parameter is true (objects supporting acknowledgement but not collaborating with a common event detection mechanism);

– the third item of information indicates the "Event Data" value; this information is always present.

(See 6.2.5.2.3 for the permitted data values).

6.2.6.3.1.3 Service procedure

The VMD having to indicate events concerning certain event class objects, should carry out the following operations:

1 - Initialize the "Event Occurrence ID" attribute (if present) of these objects either at a transaction identifying value if the "Common Event Detection Support" attribute has the real value, or at a value specific to event occurrence.

2 - Transmission, in an invocation of the Event Notification service, of the detected events whose "Event-Type" attributes of the objects concerned are identical.

The event Objects whose "Acknowledge Action Support" attributes have the Real value should be acknowledged by the Acknowledge Event Notification service if at least one event occurrence has been transmitted.

6.2.6.3.2 Confirmed acknowledge event notification service

6.2.6.3.2.1 Functionality

This service enable a Client to acknowledge several event occurrences. Each occurrence originates from an event object supporting the acknowledgement.

The limitation by the server of the number of objects to be acknowledged in a service invocation (1 to n) is a local issue.

6.2.6.3.2.2 Service primitives

The structure of the service is shown in Table 82.

Table 82 – Confirmed acknowledged event notification service parameter

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
List Of Event Identification	M	M (=)		
List Of Event Occurrence ID Type Description	U	U (=)		
List Of Event Occurrence ID Values	M	M (=)		
Result(+) (Comp)			S	S (=)
List of Access Result			M	M (=)
List Error Type			C	C (=)
Result(-)			S	S (=)
Error Type			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

List Of Event Identification

This parameter gives the list of Event objects to be acknowledged.

List Of Event Occurrence Id Type Description

The content of this parameter can be empty.

This parameter, when not empty, specifies for each of the objects to be acknowledged, the descriptor of the type of event occurrence. Since all the descriptors are identical, it is possible to reduce the list of descriptors to a single descriptor.

The types permitted are those defined by Type Data Specification (except Error Type). (See 6.2.5.2).

List Of Event Occurrence ID Values

This parameter specifies the value of the event occurences for each of the objects to be acknowledged. (See 6.2.5.2.3 for the permitted data values).

Result(+)

This selection informs the client that the server was able to successfully acknowledge at least one event object among those appearing in the request.

List Of Access Result

This parameter lists the event objects whose acknowledgement was not successful.

List Of Error Type

This parameter indicates the reason for failure for each acknowledgement that ended unsuccessfully.

Result(-)

This selection informs the client that the server was not able to execute the service successfully.

6.2.6.3.2.3 Service procedure

The VMD verifies the coherence between the Event objects described in the List of Event Identification and the List of Event Occurrence ID Type Description parameter, if it is provided. In case of incoherence of type or number a negative result is returned. If there is coherence, the VMD can use the Event Occurrence ID parameter to select the event occurrence(s) to be acknowledged from each of the objects identified by the List of Event Identification parameter.

On a channel using the access protections, the VMD should verify the coherence of access rights before each acknowledgement operation.

An operation report is supplied in the List of access result parameters for each acknowledgement operation. An error code is obligatorily given in the List of Error type parameters for each acknowledgement failure or insufficiency of access rights.

6.2.6.3.3 Confirmed alter event condition monitoring service

6.2.6.3.3.1 Functionality

This service enables modifying the "Event State" attribute of several Event objects.

The limitation by the server of the number of objects to be modified (1 to n) in a service invocation is a local issue.

6.2.6.3.3.2 Service primitives

The structure of the service is shown in Table 83.

Table 83 – Confirmed alter event condition monitoring service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
List of Event State Action ID	M	M (=)		
Event Identification	M	M (=)		
Event State action selection	M	M (=)		
List Of Event State Type description	U	U (=)		
List Of Event State Value	M	M (=)		
Result(+) (Comp)			S	S (=)
List of Access Result			M	M (=)
List Error Type			C	C (=)
Result(-)			S	S (=)
Error Type			M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Argument

List Of Event State Action ID

This parameter gives the event object list to be modified. Each modification is identified with the help of the two parameters mentioned above:

Event Identification

This parameter gives the identification of Event object.

Event State Action Selection

This parameter gives the identification of the chosen action.

List Of Event State Type Description

The content of this parameter can be empty.

This parameter, when not empty, specifies the descriptors of the "Event State" type listed in this service. Since all the descriptors are identical, it is possible to reduce the list of descriptors to a single descriptor.

The types permitted are those defined by type Data Specification (except Error Type). (See 6.2.5.2).

List Of Event State Value

This parameter specifies the "Event State" values listed in this service.

(See 6.2.5.2.3 for the permitted data values).

Result(+)

This selection indicates to the client that the server was able to modify successfully at least one element of the Event state attributes of the objets appearing in the request.

List Of Access Results

This parameter lists the "Event State" modifications which were not successfully carried out.

List Of Error Types

This parameter indicates the reason for failure for each unsuccessful modification.

Result(-)

This selection indicates to the client that the server was not able to carry out the service successfully.

6.2.6.3.3.3 Service procedure

The VMD checks the coherence between the Event objects described in the List of Event State Action ID and the List of Event State Type Description parameter, if it is provided. In case of incoherence on the type or number a negative result is returned. If there is coherence, the VMD carries out the modification of the "Event State" attributes.

On a channel using the access protection, the VMD should check the access right coherence before each modification operation.

An operation report is supplied in the List of access result parameter for each modification operation. An error code is mandatorily given in the List of Error type parameter for each modification failure or insufficiency of access rights.

6.2.6.3.4 Confirmed get alarm summary service

6.2.6.3.4.1 Functionality

This service enables reading the Event_Message attribute of several Event objects The limitation by the server of the number of objects (1 to n) to be read in a service invocation, is a local issue.

6.2.6.3.4.2 Service primitives

The structure of the service is shown in Table 84.

Table 84 – Confirmed get alarm summary service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Type with result	M	M (=)		
List of event Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
List of access result			M	M (=)
List of Event_Message Type Description			C	C (=)
Event State Type Description			C	C (=)
Event Occurrence ID Type Description			C	C (=)
Event Data type Description			M	M (=)
List of Event_Message Value			M	M (=)
Event State Value			C	C (=)
Event Occurrence ID value			C	C (=)
Event Data Value			M	M (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Type with result

This parameter indicates if the Event_Message type attributes read should be supplied in the positive response.

List Of Event Identification

This parameter describes the identification of many Event objects to be accessed.

Result(+)

This selection indicates that at least one "Event_Message" attribute object has been read successfully.

List Of access Results

The List of Access Result parameter specifies the failure/success result of all the Event objects read, respecting the order of appearance in the List of Event Identification parameter.

List Of Event Message Type Descriptions

The content of this parameter is empty if the parameter Type With Result of the request is equal to false.

This parameter, when it is not empty, specifies the descriptor type of Event_Messge attribute of objets read successfully or the ErrorType descriptor for objects read unsuccessfully. This information is provided respecting the order appearing in the List Of Event Identification parameter. In case the "Event_Message" descriptor attributes of the

object read are of the same type, the list can be reduced to the description of a single Event_Message.

Each Event_Message descriptor of the structure type includes in the indicated order, the descriptor of the Event State attribute if it is present, the descriptor of the Event Occurrence ID attribute if it is present and the descriptor of the Event Data attribute. The types permitted are those defined by Type Data Specification (except the Error Type). (See 6.2.5.2).

List Of Event Message Values

The List of Event_Message Values parameter specifies the Event_Message attribute value of the objects read successfully and the ErrorType value for objects read unsuccessfully. This information is provided respecting the order in which it appears in the List Of Event Identification parameter.

Each Event_Message value in the indicated order consists of the Event State attribute value if it is present, the Event Occurrence ID attribute value if it is present and the Event Data attribute value. (See 6.2.5.2.3 for the permitted data values).

Result(-)

The service is invalid or not accepted by the VMD. An error code is provided.

6.2.6.3.4.3 Service procedure

The VMD provides each accessed object with a data and an operation report respecting the order indicated in the request. It analyses the value of the Type With Result parameter to determine if it should provide in the response the type description of each Event_Message attribute read.

In case of a read operation carried out on a channel using access protection (service 2 quality), the VMD should check the coherence of access rights before each reading operation.

For each Event_Message attribute read successfully, the VMD provides a positive indicator in the List of Access result, its descriptor type, if necessary, in the List of Event_Message Type Description and its value in List of Event_Message Value.

For each unsuccessful Event_Message attribute read unsuccessfully, the VMD provides an error indicator in the List of Access Result, the ErrorType descriptor in the List of Event_Message Type Description and the error value in the List of Event_Message value.

6.2.6.3.5 Confirmed get event condition attribute service

6.2.6.3.5.1 Functionality

This service allows reading the attributes of an Event object

6.2.6.3.5.2 Service primitives

The structure of the service is shown in Table 85.

Table 85 – Confirmed get event condition attributes service parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument (Comp)				
Event Identification	M	M (=)		
Result(+) (Comp)			S	S (=)
List of identification			M	M (=)
Object Class			M	M (=)
Event Type			M	M (=)
Event Message type description			M	M (=)
Event Access Protection			C	C (=)
Password			M	M (=)
Access Groups			M	M (=)
Access Rights			M	M (=)
Extension			U	U (=)
Result(-)			S	S (=)
Error Type			M	M (=)

NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.

Argument

Event Identification:

This parameter indicates the object whose attributes are to be retrieved.

Result(+)

List of identification:

This parameter indicates the object identifiers.

Object class:

This parameter indicates the class of the object.

Event Type:

This parameter indicates the event object type

Event Message type description:

This parameter indicates the type of Event-Message attribute of the Event object.

Event Access Protection:

This parameter specifies three items of information if the object is protected.

Password:

This parameter indicates the password of the object.

Access Groups:

This parameter indicates the groups to which the object belong.

Access Rights:

This parameter indicates the protections of the object.

Extension:

This parameter gives additional information

Result(-)

The service is invalid or not accepted by VMD. An error code is provided.

6.2.6.3.5.3 Service procedure

The server carries out the following actions:

– check the existence of the object in question.
– read the object attributes.

If all the actions were performed correctly, the server transmits a positive response.

If an action ends in a failure, the server transmits a negative response.

Whatever the service quality of the channel via which this service is routed no access restriction should be applied through the object access protection.

6.2.7 Directory ASE

To be completed later

6.3 ARs

6.3.1 Messaging common service (MCS) AR ASE

This subclause details a Messaging Common Service Element used in the AL environment for monitoring application associations (its role being equivalent to that of ACSE in the ISO environment), and to enable transfer of data units from the application layer in the associated and non-associated modes.

6.3.1.1 Overview

This application service element is intended for use in the AL environment application layer structure.

This common element has necessarily be integrated with the specific element or elements in the context of an application entity. The MCSE is, indeed, directly projected onto the DLL and provides a user interface for receiving a specific service element such as the SubMMS described.

The services offered by this element feature two additional possibilities constituted by transmission in the associated mode and the non-associated mode.

6.3.1.2 Concepts

6.3.1.2.1 Associated mode transmission

Associated mode transmission–establishment of an application association,

– transfer of data,
– termination of an application association.

In addition to these three stages, which correspond to the phases of its life, an application association has the following characteristics:

– it necessitates establishing and maintaining an agreement concerning the transfer of data between the two application entities communicating and the data link layer on which they are based,

– it enables negotiation, between the two application entities, of the user options and parameters which affect the transmission of data,

– it supplies a context to which the successive data units transmitted are logically related,

– it guarantees transfer by reserving resources and ensuring reliable transmission of the data units exchanged,

– it makes it possible to identify the communicating entities, thus avoiding transmitting their designations at each data transfer.

Associated mode transmissions thus more closely meet the requirements of the applications for long communications with a given interaction traffic, such as, for instance, the remote configuring of an item of equipment, and exchanges of control data between equipment in a distributed application. In such cases, the entities involved agree on the nature of the interactions and resources reserved before proceeding with exchanges of data units.

This agreement is obtained by:

– configuring the two entities involved in the association, in the case of a pre-negotiated association,

– negotiation exchanges between the two entities involved in the association, in the case of a negotiated association.

6.3.1.2.2 Non-associated mode transmission

Non-associated mode transmissions between application entities take place in a single data transmission phase, from the source entity to one or more recipient entities, without establishing an association.

This type of transmission, the lifetime of which is limited to that of the exchange of a data unit, has the following properties:

– it is necessary to attach to the data to be transmitted all the necessary information for relaying and interpreting it, such as, for instance, identification of the expected communicating entities, the quality of service desired, the application context to which the data units logically relate, etc.,

– it makes no presumptions concerning the quality of the service and the contexts which may be applied for one or more successive invocations of a service, between a pair or a set of entities.

Non-associated mode transmissions most closely meet the requirements of application for occasional, spontaneous exchanges, without establishing, maintaining and terminating an association. For instance, for observing variables of an application by equipment randomly connected to the bus.

In such a case, the source entities involved need to have prior knowledge of the recipient entities to which they wish to address themselves, for the transfer of data to take place correctly.

6.3.1.2.3 Quality of service

6.3.1.2.3.1 Quality of service definition

The term quality of service relates to certain observable properties of a transmission, as described by different quality of service parameters.

Some of the quality of service parameters specifically relate to associated mode services and others to non-associated mode services, as shown in Table 86.

Table 86 – Classification of service quality parameters

Mode	Phase	Parameter
Associated	• Establishment • Transfer • Termination	• Establishment duration • PDU size • Transfer rate • Number of retries • Anticipation factor • SDU size • Termination duration
Non-associated	• Data transfer	• Priority

In the associated mode, the quality of service offered for the transfer of data results from the negotiation carried out during the establishment phase of the association.

The negotiation may result in reduction of the quality of service proposed by the user. The reduction in quality of service consists in acting on all or part of its parameters. The semantics of these actions are specific to each parameter, and specified in their definitions.

In the non-associated mode, the quality of service offered for the transfer of data is proposed by the source entity and is complied with by the service provider insofar as possible.

The quality of service parameters relating to the associated mode, whose values can be negotiated at the time of opening and association, are the following and are downgraded as indicated:

– PDU size becomes smaller,
– transfer rate becomes smaller,
– the number of retries becomes smaller,
– anticipation factor becomes smaller,
– SDU size becomes smaller.

Some of these negotiable parameters can effectively be negotiated for a specific association.

However, in the case of a specific association, the unnegotiated parameters will include the unnegotiable ones, the negotiable ones not supported and possibly some negotiable ones whose values are known implicitly by the correspondents.

Finally, in the case of a multipoint association which is necessarily pre-negotiated, only the PDU size and transfer rate parameters are supported.

NOTE In the context of a pre-negotiated association, the service quality parameters should be assigned values consistent with the properties of the devices participating in the association.

6.3.1.2.3.2 Duration of the establishment

The duration of the establishment is the maximum time interval between the association opening demand and confirmation of opening. This time interval includes the time taken for turning round the MCS user called from the correspondent, as well as the time taken for transfer of the request and opening response.

The turn-around time is an intrinsic property of the device, the value of which is utilised by the association opening monitoring mechanism.

NOTE The value of this duration of establishment, which is not proposed by the user, is obtained via network management functions.

6.3.1.2.3.3 PDU size

The size of the data units is the maximum size, as a number of octets, of the protocol data units (MCS-APDU) of the application layer encoded which can be exchanged on an association.

The value of this parameter is negotiated on the basis of the intrinsic properties of the two devices participating in an association.

Indeed, each device globally offers, to all the associations it supports, an application layer protocol data unit maximum size possibility. The maximum size taken in an association is the least of the maximum sizes offered by the two devices participating in it.

The value of this quality of service parameter is thus accounted for by the association monitoring mechanism, which guarantees that all the application data units are smaller than this size.

6.3.1.2.3.4 Transfer rate

The transfer rate is determined, for both directions of transfer of an association, in terms of the guaranteed minimum number of data units of the application layer protocol which it is possible to transmit per unit time.

This transfer rate corresponds to that part of the total rate available in a device for association purposes.

NOTE The allocation of a transfer rate to an association at device level needs to be consistent with the total capacities and rates already allocated.

For reference, the definition of the total rate available in a device at the data link layer is given below. The abbreviations are defined as follows:

Φ equip = Messaging flow available at the equipment level

Φ cyc = messaging flow dependent on the cyclic messaging service available in a device

Φ aper = messaging flow dependent on the aperiodic messaging service available in a device

φ mac = messaging flow reserved by the microcycle in the various messaging windows. This flow is that which is shared between all system devices for aperiodic messaging.

φ dem = message exchange request flow, for a given device. This corresponds to the sum of the flows offered by the various periodic identifiers supporting a message request in a device.

Definitions

Φ equip = Φ cyc + Φ aper

Φ cyc = $\sum_i [1/T_i]$

i = each F_MSGcyc queue of the device,

Ti = period of teh ID_MSG associated to the queue

Φ aper = ((φ dem)*(1/ $\sum_j \varphi$ dem(j)))*(φ mac)

j = each system device

φ dem = request flow of the device in question

It should be noted that

– the cyclic flow depends only on the frequency of scanning of the equipment cyclic queues by the bus arbitrator,
– the aperiodic flow corresponds to the total available flow on the bus divided between the devices on the basis of the request flow.

NOTE The transfer rate thus allocated to an association in both directions is under network management surveillance.

6.3.1.2.3.5 Number of retries

The number of retries is the maximum number of retries allowed to the MCS service provider subsequent to an acknowledged transmission request made by the user in a point-to-point association.

This parameter is defined globally for both transfer directions and negotiated during the association establishment phase.

Retries continue until the value of this parameter is reached, as long as the acknowledged transmission has not been properly completed.

6.3.1.2.3.6 Anticipation factor

The anticipation factor is the maximum number of data transfers that an entity can have initiated at a given time in an association. This makes it possible to optimize the pass-band available for an association.

The anticipation factor concept is defined for each transfer direction. The anticipation factor thus defined is negotiated during the association establishment phase.

6.3.1.2.3.7 SDU size

The SDU size is the maximum size of a service data unit (MCS-ASDU) which can be submitted by an MCS user, to a given association, and it is designed for both directions of transfer.

This size can be negotiated during opening of an association and is expressed as a whole number of maximum size MCS protocol data units.

This parameter is only negotiated if the association implements segmentation, and in this case has a value greater than 1.

6.3.1.2.3.8 Termination duration

The termination duration is the maximum time interval between the association termination request and termination confirmation. This time interval corresponds to the MCS user turn-around time, as well as the times of request transfer and termination response.

This turnaround time is an intrinsic property of the device, the value of which is utilized by the association closure monitoring mechanism.

NOTE The value of this termination duration, which is not submitted by the user, is obtained via network management functions.

6.3.1.2.3.9 Priority

The concept of priority is specific to the non-association transmission mode.

In such transmissions, the priority assigned by the data transfer source concerns the relative importance of one data transfer relative to another so as to allow for the revocation of certain PDUs to free resources for others of higher priority.

NOTE This concept does not apply in associated mode transmissions as the quantity of resources implemented is reserved prior to a transfer of data. Furthermore, there is no global priority of resources of an association, making it possible to allow for its revocation, thus making it possible to free its resources for another association of higher priority. Only network management functions can allow early freeing of resources of an association despite user requirements.

6.3.1.2.4 Identification of correspondents

Each correspondent involved in an associated or non-associated mode transmission is unambiguously identified by the system by a four-part identifier consisting of:

- “AP title” to identify the AP to which the entity belongs in one of the system devices,
- “AE qualifier” which identifies the AE used in the AP context,
- “API identification” which identifies a specific activity of an AP in operation, specific use of the AE implemented for the requirements of a specific API.

During associated and non-associated mode transmissions, these indications are optionally transmitted. They are transmitted, during the establishment of an association or a non-associated mode data transfer, to make it possible for the user called or the destination or destinations to respectively:

[1] Inform them of the identification of the caller of the source.

This identification of the caller can be shortened to “AP title” and “AE qualifier”, the called entity being able to deduce this information from the link address of the entity which initiated the exchange.

In addition, the “API identification” can also not be transmitted if the entity initiating the exchange proceeds to establish a bilateral relationship between the APIs and the link addresses associated with the AE.

NOTE 1 This means that there is no multiplexing of the APIs at a given link address (of ADAE type).

[2] To verify whether the identification considered by this request is that localised by the link address utilised.

NOTE 2 Indeed, every AE of an AP is projected onto one or more link addresses at directory level prior to any transmission.

[3] Selection of one of the APIs to which the transmission is addressed.

NOTE 3 This can be avoided, in the particular case where any link address is used for transmissions to a single API.

[4] To propose an AEI identification, the creation of which is triggered by establishing an association at the called entity (only applies to the associated mode).

NOTE 4 The called entity can opt to accept or refuse this identification proposal made by the calling entity. In the case of refusal, the value attributed by the called entity is returned to the calling entity in the association opening request response.

The parameters relating to the identification are transmitted to meet the requirements indicated above are shown in Table 87.

Table 87 – Identification parameters

Parameter name	Role			
	[1]	[2]	[3]	[4]
Calling AP title	C			
AE qualifier	C			
API identification	C			
AEI identification	M			
Called AP title		M		
AE qualifier		M		
API identification			M	
AEI identification				M

6.3.1.2.5 Application context

Entities which intercommunicate use a set of rules to construct the PDUs sent and to interpret the PDUs received, these constitute the bases of the application context.

The bases of an application context comprise:

– a specification of one of the ASEs, which make up the AE supporting the application context.

NOTE This means that each association can only implement one of the ASEs which can contain an AE in the AL environment.

– a specification, for the ASE considered in this context, of one of the abstract syntax options supported by the latter in the framework of the AE supporting the context,

– a specification for one of the transfer syntax options supported by the AE.

For a specific application context, its bases have a name manipulated by the MCS services.

All ASE standards which offer MCS mapping, propose contingent universally recognised names. Any other application, whether defined on the basis of standards or not, has a scope specific to the distributed application in which it is implemented.

The name of an application context is structured as follows:

Name of context ::= {

- ASE type (Universal; Application)
- ASE name (Identification)
- Abstract syntax type (Universal; Application)
- Abstract syntax name (Identification)
- Transfer syntax type (Universal; Application)
– Transfer syntax name (Identification) }

NOTE The coding of a context name is proposed in the PDUs of MCS.

In associated mode transmissions, the context name is exchanged during the establishment of the association to enable negotiation of bases of the application context of the future association.

Unlike in non-associated mode transmissions, the context name is exchanged during each transfer contiguous to data.

6.3.1.3 Services

6.3.1.3.1 Overview

The nature of the services described below is shown in Table 88.

Table 88 – List of MCS AR ASE services

Service	Type
A_ASSOCIATE	Confirmed
A_RELEASE	Confirmed
A_ABORT	Unconfirmed
A_DATA	Unconfirmed (ack)
A_UNIDATA	Unconfirmed (ack)
NOTE (ack) = acknowledgement possible	

NOTE The A_DATA and A_UNIDATA are unconfirmed within the meaning of the ISO service convention but nevertheless feature local confirmation which is generated with allowance for acknowledgement or non-acknowledgement, depending on whether this has been requested or not.

6.3.1.3.2 A_ASSOCIATE service

6.3.1.3.2.1 Functionality

This service is used to establish an association to negotiate the parameters which will control the transfer of data between the two AEIs participating in it.

6.3.1.3.2.2 Service primitives

The parameters of this service and description of its semantics are described in Table 89 and the accompanying text.

Table 89 – A_Associate service parameters

Parameter name	Req	Ind	Rsp	Cnf
Calling AEI access point	M	M (=)	M	M (=)
Called AEI access point	U	C (=)	M	M (=)
Conformity class	M	M (=)	M	M (=)
"Cyclic flow" option	U	C		
Quality of service	U	C	C	C (=)
Calling entity identification	U	C (=)		
Called entity identification	U	C (=)	U	C (=)
Application context	U	C (=)	U	C (=)
Negotiation result			M	M (=)
User information	M	M (=)	M	M (=)
For data link layer:				
Calling AE address	M	M (=)	M	M (=)
Called AE address	M	M (=)	M	M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

Calling AEI access point:

This parameter corresponds to the link address (of ADAEI type), making it possible to localise the AEI of the calling entity. The calling entity obtains it by means of its directory function.

Called AEI access point:

This parameter corresponds to the link address (of ADAEI type), making it possible to localise the AEI of the called entity. This access point can be proposed by the calling entity for use by the called entity. However, whether it is proposed or not, the called entity returns, in its response, the value of the access point used for its AEI.

Conformity class:

This parameter designates the MCS provider elements which are used in the context of a specific association.

The elements involved, which are designated by the associated conformity parameter names, are the following:

– association termination (PV_RE),
– acknowledgement (PH_AK),
– resend/antiduplication (PH_RA),
– segmentation/reassembly (PH_SR),
– flow control (PH_CF).

“Cyclic flow” option:

The cyclic flow option indicates the type of flow required for the association. Indeed, the flow offered by the association, depending on whether it is based on cycling messaging link services for both transmission directions or aperiodic services for either of the directions, will or will not be independent of the messaging requirements of the other system devices (6.3.1.2.3.4).

By means of this parameter, the calling entity can specify, as a condition of establishment, that the association offers a cyclic flow.

Quality of service:

For this parameter, quality of service is a subset of those defined earlier (see 6.3.1.2.3):

– transfer rate,
– number of retries,
– anticipation factor,
– SDU size.

These quality of service parameters are those which are negotiable on the basis of a value proposed by the calling entity.

Identification of the calling entity:

A calling entity is fully identified by four parameters:

– AP title,
– AE qualifier,
– API identifier,
– AEI identifier.

However, of these four parameters, the only ones present will be those which are necessary in view of the operating conditions of the system (see 6.3.1.2.4).

NOTE All these parameters may be absent. Furthermore, no modifications are made by the MCS provider.

Identification of the called entity:

A called entity is fully identified by four parameters:

– AP title,
– AE qualifier,
– API identifier,
– AEI identifier.

A called entity is fully or partially identified in an association establishment request, depending on the requirements for verification and selection adopted, in view of the operating conditions of the system (see 6.3.1.2.4).

NOTE All these parameters may be absent. Furthermore, no modifications are made by the MCS provider.

Application context:

This parameter corresponds to the name of the application context proposed by the calling entity.

The responder returns the same application context name or a different one, but compatible with that proposed.

Negotiation result:

This parameter makes it possible to report the result reserved for the association opening request, i.e. ACCEPT, REJECT_PROVIDER, REJECT_USER.

This result (acceptance or refusal) is exclusively related to the parameters exchanged in the establishment request.

User information:

The specific service elements which invoke this service use this parameter to negotiate other information characterising the association from their point of view.

For the data link layer:

List of the service parameters proposed by the calling entity for the requirements of the data link layer.

Called AE address:

This parameter corresponds to a link address (of the ADAE type) used to localise the AE of the calling entity (see 6.1.7.4.2).

Called AE address:

This parameter corresponds to the link address (of ADAE type) used to localise the AE of the called entity (see 6.1.7.4.2).

6.3.1.3.2.3 Service procedure

The chaining of the primitives leading to successful establishment of the association is shown in Figure 54.

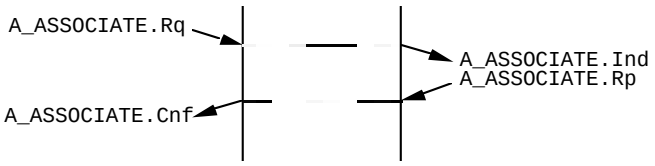


Figure 54 – A_Associate service procedure

This chaining diagram can fail:

– either due to the MCS provider being incapable of addressing the requirements of the calling entity,
– or due to disagreement by the called entity concerning its requirements,
– or due to transmission faults,
– or due to the implementing of ABORT services on this association during establishment.

NOTE In the first three cases, the service is only completed with negative confirmation, whereas in the last there is no confirmation.

The A_ABORT service is, indeed, the only which can be implemented on an association during establishment. This may be due to the action of:

– the calling entity awaiting confirmation of establishment,
– the called entity prior to its establishment response,
– the MCS provider, on specific faults or as a result of intervention at network management level.

NOTE Simultaneous opening is not considered possible as it is postulated that it is impossible to allocate, for two distinct entities, the same access points for localising the association ends. The network management is relied on to avoid this type of problem.

The A_ASSOCIATE service primitives, which are implemented on running the association establishment procedure, perform the following operations:

• A_ASSOCIATE.Rq primitive:

Of the primitives transiting via the calling entity, only the following are processed by the requester element before transmission in the encoded PDU:

- The “conformity class” can be reduced, provided the conformity constraints are complied with , depending on the capacities available.
- The “cyclic rate” option, when it has the value YES, can impose a pure cyclic messaging flow at link level. Consequently, one of the cyclic messaging identifiers for which this device is configured is adopted for the association in accordance with the criteria specific to the requesting element.

– The following quality of service parameters can be reduced:

“Transfer flow”

“Anticipation factor”

“SDU size”

“Number of retries”

• A_ASSOCIATE.Ind primitive:

Of the parameters received from the association establishment requester, only the following are processed by the responder element.

- The “conformity class” can be deduced, provided the conformity constraints are complied with , from the capacities available.
- The “cyclic rate” option, when it has the value YES, can impose a pure cyclic messaging flow at link level. Consequently, one of the cyclic messaging identifiers for which this device is configured is adopted for the association in accordance with the criteria specific to the responder element.
- The following quality of service parameters can be reduced:

“PDU size”

“Transfer flow”

"Anticipation factor"

"SDU size"

"Number of retries"

- The "called entity identification" is utilised to make the checks arising owing to the presence of the different elements of which it is composed (see 6.3.1.2.4). If the check reveals a discrepancy between this called entity identification considered by the calling entity and the called entities effectively residing in the device which receives the association establishment request, a negative result is returned in the association establishment response.
- The "Application context" can be reduced in accordance with the reduction rules specific to the service element (see 6.1.8.3). If reduction is not possible, a negative result is returned in the association establishment response.

• A_ASSOCIATE.Rp primitive:

Of the parameters which transit via the called entity, the result is the first that has to be considered. Depending on whether it is positive or negative, the nature of the processing carried out before transmission of the establishment response is different.

If the result is negative, any resources previously reserved on reception of the establishment request are freed.

If the result is positive, the quality of service parameters are considered and, for the following, the values are adjusted:

"Transfer rate"

"Anticipation factor"

"SDU size"

"Number of retries"

• A_ASSOCIATE.Cnf primitive:

Of the parameters received from the called entity, the result is the first which has to be considered. Depending on whether it is positive or negative, the nature of the processing performed is different.

If the result is negative, the resources previously reserved on receiving the establishment request are freed.

If the result is positive, the other parameters are considered and, for each of the following, the values are adjusted on the basis of the value received from the responder element.

The parameters adjusted are the following:

- The quality of service:

"PDU size"

"Transfer rate"

"Anticipation factor"

"SDU size"

"Number of retries"

6.3.1.3.3 A_RELEASE service

6.3.1.3.3.1 Functionality

This service is used to terminate an association and, consequently, to free the resources implemented by it, as soon as its current data transfers are completed. Success of this service depends on the willingness of the responder user of this service.

6.3.1.3.3.2 Service primitives

The parameters of this service and description of its semantics are described in Table 90 and the accompanying text.

Table 90 – A_Release service parameters

Parameter name	Req	Ind	Rsp	Cnf
User information Result	U	C (=)	U M	C (=) M (=)
NOTE The method by which a confirm primitive is correlated with its corresponding preceding request primitive is a local matter. The method by which a response primitive is correlated with its corresponding preceding indication primitive is a local matter. See 1.2.				

User information:

The specific service elements which invoke this service can use this parameter to exchange additional information concerning revocation.

Result:

Simply indicates whether or not the responder user has accepted this association closure. This parameter takes the following values:

– Accepted

– Refused

– Other (definitions pending)

6.3.1.3.3.3 Service procedure

Chaining of the primitives leading to successful association termination is as indicated in Figure 55.

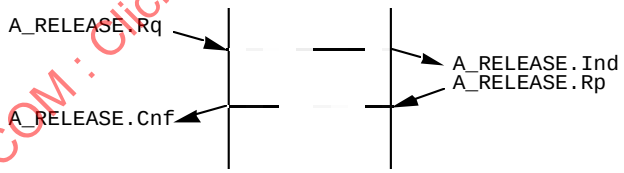


Figure 55 – A_Release service procedure

The termination request can be invoked by either the calling entity or the called entity of the association.

Simultaneous termination requests can occur when a termination request has been made by one of the correspondents of the association which receives a termination request from the other correspondent before it has received confirmation of its request.

In the case of simultaneous requests being detected by the provider, whether at called entity or calling entity level, the termination primitive chaining is broken. The provider proceeds with revocation of the association and freeing of the associated resources, after having invoked the ABORT.Rq service.

It is important to note that the requester of termination of an association cannot activate any primitive of the association other than A.ABORT.Rq until it has received confirmation of termination. Primitives can then be exchanged once more in the association if termination is refused by the corresponding entity.

NOTE It needs to however be borne in mind that a termination request can only be considered for a previously-established association, and that pre-negotiated associations can only be revoked by disabling their configuration under network management control.

Similarly, the responder to the termination request cannot activate any primitive other than A_ABORT.Rq in the association after it has received a termination indication. Primitives can once more be exchanged in the association if the responder local user refuses termination on this indication.

As soon as successful termination is confirmed, the association is revoked and the resources freed. In addition, either of the users of the association can always interrupt the running of the service by using the ABORT service, which revokes the association with the contingent loss of current data transfer.

NOTE Similarly, the running of this service can be interrupted if the network management opts to revoke the association during its creation.

6.3.1.3.4 A_ABORT service

6.3.1.3.4.1 Functionality

This service is invoked either by the user or directly by the MCS provider to revoke an association. Such revocation results in immediate freeing of the resources implemented by the association, which necessarily causes total loss of all current data transfers in it.

6.3.1.3.4.2 Service primitives

The parameters of this service and description of its semantics are shown in Table 91 and the accompanying text.

Table 91 – A_Abort service parameters

Parameter name	Req	Ind
Origin of invocation		M (=)
User information	U	C (=)

Origin of invocation:

Indicates the origin of invocation of this service, which can be invoked by the user or the provider, both at calling entity and called entity levels.

User information:

The specific application service elements which invoke this service utilise this parameter to exchange data units associated with the services that they offer. A service of this type does not guarantee their relaying.

6.3.1.3.4.3 Service procedure

The chaining of primitives relating to the revocation of an association is shown in Figure 56.

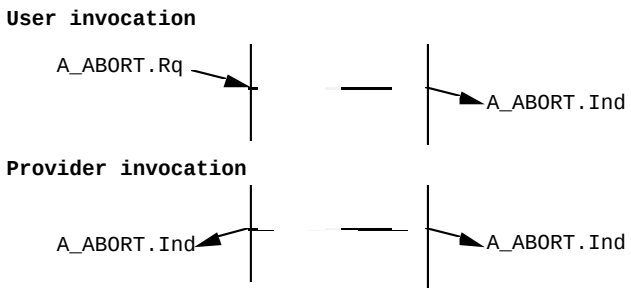


Figure 56 – A_Abort service procedure

A certain number of chaining variants, leading to special chaining, need to be considered:

– The simultaneous requesting of revocation by both the called user and the calling user results in the absence of a revocation indication at both ends.
– The simultaneous invoking of this service by the user and the provider results in there only being the option of invocation of the revocation request by the user who activates this revocation and indication of provider origin revocation at the other end of the association.

Furthermore, it is important to note that this service can be invoked by the provider if a collision should occur during termination of an association.

NOTE It should be borne in mind that a revocation request, whether at the initiative of the user or the provider, can only be considered for a pre-established association and that pre-negotiated associations can only be revoked by disabling their configuration under network management control.

6.3.1.3.5 A_DATA service

6.3.1.3.5.1 Functionality

This service enables exchange of MCS user data in an association. This transfer of data may or may not be acknowledged depending on the request, and only if the association uses the acknowledgement provider element.

In such cases, an acknowledgement provider element multipoint association cannot be used.

NOTE Acknowledgement may be requested systematically, occasionally or never in a given environment, depending on the desired trade-off between reliability of transmission and pass-band usage. However, in the case where segmentation is adopted and/or the anticipation factor is greater than 1, acknowledgement is implicitly utilised whatever the user requests.

6.3.1.3.5.2 Service primitives

The parameters of this service and description of its semantics are described in Table 92 and the accompanying text.

Table 92 – A_Data service parameters

Parameter name	Req	Ind	Cnf
Acknowledgement request	M	M (=)	
User information	M	M (=)	
Transfer result			M

Acknowledgement request:

The acknowledgement of a data unit transferred using this service can be requested using this parameter in the service. The value of this parameter indicates whether the acknowledgement is requested or not. The acknowledgement, if it is requested, is sent by the receiver MCS provider of the data transfer.

This parameter is meaningless when segmentation is adopted, as in this case the acknowledgement provider element is implicitly utilised.

User information:

The specific application service elements which invoke this service utilise this parameter to exchange data associated with the services that they offer.

Transfer result:

This parameter indicates whether transfer has taken place correctly or not. When an acknowledgement request is made, this result provides an indication concerning whether it has arrived in due time, whereas when it is not requested it is a local information item qualifying the sending process.

6.3.1.3.5.3 Service procedure

The chaining of primitives leading to a successful data transfer depending on whether acknowledgement is requested or not is shown in Figure 57.

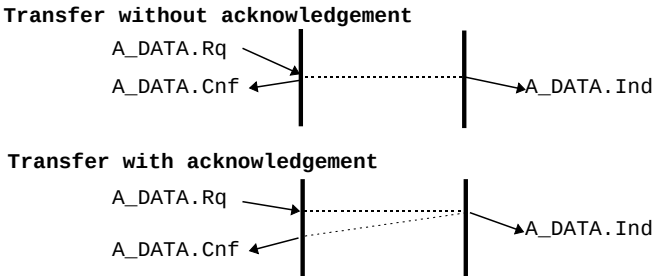


Figure 57 – A_Data service procedure

The chaining of these primitives cannot take place correctly except within the limits set by the quality of service adopted for the association.

NOTE It is thus that, for instance, exceeding of the data unit size by the sources, exceeding of the transfer rate adopted, or exceeding of the service size will result in non-completion of this chaining of primitives.

In the case of a transfer of data with acknowledgement, retries can occur to compensate for the loss of DLPDUs.

In addition, requests and acknowledgements of a number of transfers can be interlaced to optimise the bus traffic.

Finally, in the case of service requests of sizes greater than those of the protocol data units, they are segmented so that they can be relayed.

6.3.1.3.6 A_UNIDATA service

6.3.1.3.6.1 Functionality

This service enables the exchange of MCS user data outside an application association. It may or may not be the subject of an acknowledgement, depending on the operating conditions of the latter.

6.3.1.3.6.2 Service primitives

The parameters of this service and description of its semantics are described in Table 93 and the accompanying text.

Table 93 – A_Unidata service parameters

Parameter name	Req	Ind	Cnf
Acknowledgement request	M	M (=)	
Quality of service	M	M	
Source identification	U	C (=)	
Destination identification	U	C (=)	
Application context	U	C (=)	
User information	M	M (=)	
Transfer result			M
For the data link layer:			
Calling AE address	M	M (=)	
Called AE address	M	M (=)	

Acknowledgement request:

The acknowledgement of a data unit transferred with this service can be requested by means of this service parameter. The value of this parameter indicates whether acknowledgement is requested or not. This acknowledgement, if it is requested, is sent by the receiver MCS provider of the data transfer. Acknowledgement can only be requested for point-to-point data transfers.

Quality of service:

Quality of service is based on the parameters detailed in §4 of this section. Of these service quality parameters, during transfer of data in the non-associated mode, only one priority is transported contiguous with the data.

This priority is proposed by the source MCS user of this service, and determines the exchange of this service at provider level. The latter makes the commitment at source level to

– account for invoking this service if the resources are free or occupied by other invocations of lower priority not yet transmitted,

– transmit this service invocation unless the invocation is that of lowest priority pending, should another invocation of higher priority occur and all the resources be occupied.

The latter makes the commitment at destination level to

– receive an invocation of the service if the resources are free or occupied by invocations of lower priority not yet restored to the user,

– restore the invocation to the user, unless the invocation is that of lowest priority pending, when another of higher priority is received and all the resources are occupied.

Source identification:

The source identification identifies the initiator of the non-association data transfer service. A source in the context of the application layer architecture is fully identified by four parameters:

– AP title,

– AE qualifier,

– API identifier,

– AEI identifier.

Of these four parameters, the only ones present will be those necessary in view of the operating conditions of the system.

NOTE It is thus that the API identification is not necessary if each AE qualifier is operated by one single API.

Optionally, the value of this parameter is proposed by the user, which initialises a non-associated mode data transfer request. These parameters are present in the non-associated mode data transfer indication at called entity level, if they were present in the request. As for the value of these parameters in the data transfer indication, it is equal to that proposed by the source in the request.

Destination identification:

The destination identification identifies the receiver of the non-associated mode data transfer. A destination, in the context of the application layer architecture, is totally identified by four parameters:

– AP title,

– AE qualifier,

– API identifier,

– AEI identifier.

A called entity is fully or partially identified in a non-associated mode data transfer request, depending on the requirements of verification and selection adopted, in view of the system operating conditions.

The verification and selection possibilities offered at called entity level are described in 6.3.1.2.4.

NOTE For instance: (1) transit of the destination AP title makes it possible to verify that the AP localised by a link address is effectively that sought by the application; (2) transit of the API identifier makes it possible to select from it one of all those associated with the same AE qualifier.

Optionally, the value of this parameter is proposed by the operator who initialises a non-associated mode point-to-point data transfer request. It cannot be proposed in the case of multipoint data transfers.

Application context:

This parameter identifies the application context in which the source specifies to the destination that the non-associated data transfer should be interpreted.

The destination, depending on whether or not it supports this context, may or may not carry out the service requested.

Optionally, this parameter is proposed by the user who initialises a non-associated mode data transfer.

NOTE This parameter can, for instance, be transmitted on an *ad hoc* basis during an initial exchange between two entities to enable the source to verify that the destination is capable of interpreting the data units transmitted.

User information:

The specific application service elements which invoke this service utilise this parameter to exchange data units associated with the services that they offer.

Transfer result:

This parameter indicates whether the transfer has taken place correctly or not. Depending on whether or not an acknowledgement request has been made, this indication respectively qualifies the effective arrival of the acknowledgement in due time or the proper local execution of the transmission.

In the event of failure, this information states the type of fault, so as to indicate whether it has occurred locally or at the remote correspondent.

NOTE A fault detected by a remote correspondent provider necessarily results in the latter not supplying data to its user.

For the data link layer:

List of the service parameters proposed by the source user of the service for the requirements of the data link layer.

Source AE address:

This parameter corresponds to a link address (of the ADAE type) used to localise the source AE of the service (see 6.1.7.4.2).

Destination AE address:

This parameter corresponds to the link address (of ADAE type) used to localise the destination AE of the service (see 6.1.7.4.2).

6.3.1.3.6.3 Service procedure

The primitive chaining leading to successful data transfer, depending on whether or not acknowledgement is requested, is shown in Figure 58.

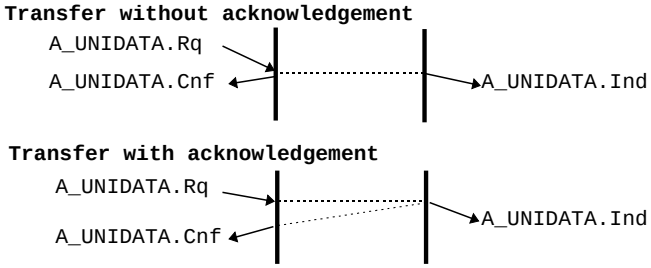


Figure 58 – A_Unidata service procedure

The primitive chaining can only take place correctly if the application context necessary for interpretation of the data is supported at destination level, and if the destination of which the identification is specified exists at receiver entity level.

When the priority requested by the source is not supported by the destination, the latter still carries out primitive chaining, using the priority level closest to that requested.

6.3.1.4 Mechanisms

6.3.1.4.1 MCS state machines

All MCS sequences which can take place, from the point of view of the user, are described by two state machines.

One machine for associated mode transmission services and the other for non-associated mode transmission services.

6.3.1.4.2 Associated mode primitive sequence

The service sequences, shown in Figure 59, depend on the state of the association with which they interact (see 6.1.6).

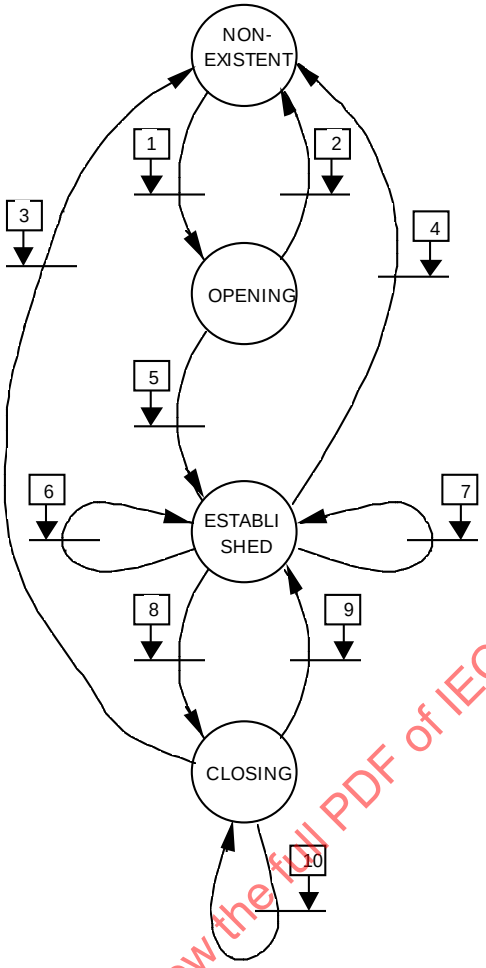


Figure 59 – Associated mode service state chart

List of events:

1. A_ASSOCIATE.Rq(); A_ASSOCIATE.Ind()
2. A_ASSOCIATE.Cnf(-); A_ASSOCIATE.Rp(-)
   A_ABORT.Ind(Provider, User); A_ABORT.Rq()
3. A_RELEASE.Cnf(+); A_ABORT.Ind (Provider, User); A_ABORT.Rq()
4. A_ABORT.Rq(); A_ABORT.Ind(Provider, User)
5. A_ASSOCIATE.Cnf(+); A_ASSOCIATE.Rp(+)
6. A_DATA.Rq
7. A_DATA.Ind(); A_DATA.Cnf(+/-)
8. A_RELEASE.Rq(); A_RELEASE.Ind()
9. A_RELEASE.Cnf(-); A_RELEASE.Rp(-)
10. A_DATA.Ind(); A_DATA.Cnf(+/-)

6.3.1.4.3 Non-associated mode primitive sequence

These primitive sequences are not constrained and can be carried out within the limits of the resources available, as shown in Figure 60.

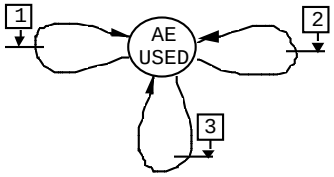


Figure 60 – Non-associated mode service state chart

List of events:

1. A_UNIDATA.Rq()
2. A_UNIDATA.Cnf(+/-)
3. A_UNIDATA.Ind()

Bibliography

IEC 61158-6-7, *Industrial communication networks – Fieldbus specifications – Part 6-7: Application layer protocol specification – Type 7 elements*

IEC 61784-1 (Ed.2.0), *Industrial communication networks* – Profiles – Part 1: *Fieldbus profiles*

_____

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 5-7: Définition des services des couches d'application – Éléments de Type 7

AVANT-PROPOS

1) La CEI (Commission Électrotechnique Internationale) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, la CEI - entre autres activités - publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études; aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant des questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toute divergence entre toute Publication de la CEI et toute publication nationale ou régionale correspondante doit être indiquée en termes clairs dans cette dernière.
5) La CEI n'a prévu aucune procédure de marquage valant indication d'approbation et n'engage pas sa responsabilité pour les équipements déclarés conformes à une de ses Publications.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de propriété intellectuelle ou de droits analogues. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

NOTE L'utilisation de certains des types de protocoles associés est limitée par les détenteurs de leurs droits de propriété intellectuelle. Dans tous les cas, l'engagement à un abandon limité des droits de propriété intellectuelle pris par les détenteurs de ces droits permet d'utiliser un type particulier de protocole de couche liaison de données avec des protocoles de couche physique et de couche application dans des combinaisons de types telles que spécifiées de façon explicite dans la série CEI 61784. L'utilisation des divers types de protocoles dans d'autres combinaisons peut exiger la permission donnée par les détenteurs respectifs de leurs droits de propriété intellectuelle.

La Norme internationale CEI 61158-5-7 a été établie par le sous-comité 65C: Réseaux de communication industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette première édition et ses parties connexes de la sous-série CEI 61158-5 annulent et remplacent la CEI 61158-5:2003. Cette édition de la présente partie constitue une révision éditoriale.

Cette édition de la CEI 61158-5 comporte les modifications significatives suivantes par rapport à l'édition précédente:

a) suppression de l'ancien bus de terrain de Type 6 à cause d'un manque d'adéquation avec le marché;

b) ajout de nouveaux types de bus de terrain;

c) partition de la partie 5 de la troisième édition en plusieurs parties numérotées -5-2, -5-3, …

La présente version bilingue (2014-12) correspond à la version anglaise monolingue publiée en 2007-12.

Le texte anglais de cette norme est issu des documents 65C/475/FDIS et 65C/486/RVD.

Le rapport de vote 65C/486/RVD donne toute information sur le vote ayant abouti à l'approbation de cette norme.

La version française de cette norme n'a pas été soumise au vote.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de maintenance indiquée sur le site web de la CEI sous http://webstore.iec.ch dans les données relatives à la publication recherchée. À cette date, la publication sera:

- reconduite;
- supprimée;
- remplacée par une édition révisée, ou
- amendée.

NOTE La révision de la présente norme sera synchronisée avec les autres parties de la série CEI 61158.

La liste de toutes les parties de la série CEI 61158, publiées sous le titre général *Réseaux de communication industriels – Spécifications des bus de terrain*, peut être consultée sur le site web de la CEI.

INTRODUCTION

La présente partie de la CEI 61158 est l'une d'une série produite pour faciliter l'interconnexion de composants de systèmes d'automation. Elle est liée à d'autres normes dans l'ensemble tel que défini par le modèle de référence des bus de terrain "à trois couches" décrit dans le rapport technique CEI/TR 61158-1.

Le service d'application est fourni par le protocole d'application utilisant les services disponibles de la couche liaison de données ou autre couche immédiatement inférieure. La présente norme définit les caractéristiques des services d'application que les applications de bus de terrain et/ou la gestion de système peuvent exploiter.

Dans l'ensemble de normes relatives aux bus de terrain, le terme "service" désigne une fonctionnalité abstraite fournie par une couche du modèle de référence de base de l'interconnexion des systèmes ouverts (OSI, Open Systems Interconnection) à la couche immédiatement supérieure. Ainsi, le service de la couche Application défini dans la présente norme est un service architectural conceptuel, indépendant des divisions administratives et de mise en œuvre.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 5-7: Définition des services des couches d'application – Éléments de Type 7

1 Domaine d'application

1.1 Vue d'ensemble

La couche application de bus de terrain (FAL, Fieldbus Application Layer) fournit aux programmes d'utilisateur un moyen d'accéder à l'environnement de communication du bus de terrain. À cet égard, la FAL peut être vue comme une «fenêtre entre des programmes d'application correspondants».

La présente norme fournit des éléments communs pour les communications de messagerie à temps critique ou non critique entre des programmes d'application dans un environnement d'automation. Le terme “à temps critique” sert à représenter la présence d'une fenêtre temporelle, dans les limites de laquelle une ou plusieurs actions spécifiées sont tenues d'être parachevées avec un niveau défini de certitude. Le manquement à parachever les actions spécifiées dans les limites de la fenêtre temporelle risque d'entraîner la défaillance des applications qui demandent ces actions, avec le risque concomitant pour l'équipement, l'installation et éventuellement pour la vie humaine.

La présente norme définit de manière abstraite le service observable de l'extérieur fourni par la couche Application de bus de terrain de Type 7 en termes

a) d'un modèle abstrait pour définir des ressources (objets) d'application capables d'être manipulées par les utilisateurs par l'intermédiaire de l'utilisation du service FAL,

b) des actions et événements primitifs du service,

c) des paramètres associés à chaque action primitive et événement primitif, et la forme qu'ils prennent, et

d) de l'interrelation entre ces actions et événements, et leurs séquences valides.

La présente norme vise à définir les services mis en place pour

1) l'utilisateur de la FAL, à la frontière entre l'utilisateur et la couche Application du modèle de référence de bus de terrain, et

2) la Gestion des systèmes, à la frontière entre la couche Application et la Gestion des systèmes selon le modèle de référence de bus de terrain.

La présente norme spécifie la structure et les services de la couche application des bus de terrain de la CEI, en conformité avec le Modèle de référence de base de l'OSI (ISO/CEI 7498) et la Structure de la couche application de l'OSI (ISO/CEI 9545).

Les services et protocoles de la FAL sont fournis par des entités d'application (AE, application entity) de la FAL contenues dans les processus d'application. L'AE de la FAL se compose d'un jeu d'Éléments de service application (ASE, Application Service Element) orientés objet et d'une Entité de gestion de couche (LME, Layer Management Entity) qui gère l'AE. Les ASE fournissent des services de communication qui fonctionnent sur un ensemble de classes d'objets de processus d'application (APO, application process object) connexes. L'un des éléments ASE de la FAL est un ASE de gestion qui fournit un ensemble commun de services destinés à la gestion des instances des classes de la FAL.

Bien que ces services spécifient, du point de vue des applications, la manière dont la demande et les réponses sont émises et délivrées, ils n'incluent pas de spécification relative à ce que les applications qui demandent et qui répondent doivent en faire. En d'autres termes, les aspects comportementaux des applications ne sont pas spécifiés; seule une définition des demandes et réponses que ces applications peuvent envoyer/recevoir est établie. Cela permet une plus grande flexibilité aux utilisateurs de la FAL pour normaliser un tel comportement d'objet. En plus de ces services, certains services d'appui sont également définis dans la présente norme pour fournir l'accès à la FAL afin de commander certains aspects de son fonctionnement.

1.2 Spécifications

L'objectif principal de la présente norme est de spécifier les caractéristiques des services conceptuels de la couche application qui sont adaptés à des communications à temps critique et, donc, complètent le Modèle de référence de base OSI en guidant le développement des protocoles de couche application pour les communications à temps critique.

Un deuxième objectif est de fournir des trajets de migration à partir de protocoles préexistants de communications industrielles. C'est ce dernier objectif qui donne naissance à la diversité de services normalisés tels que les divers types de la CEI 61158.

La présente spécification peut être utilisée comme la base pour les interfaces de programmation d'applications (Application Programming Interfaces) formelles. Néanmoins, elle n'est pas une interface de programmation formelle et il sera nécessaire pour toute interface de ce type de traiter de questions de mise en œuvre qui ne sont pas couvertes par la présente spécification, y compris

a) les tailles et l'ordonnancement des octets pour les divers paramètres de service à plusieurs octets, et

b) la corrélation de primitives appariées "request-confirm" (c'est-à-dire: demande et confirmation) ou "indication-response" (c'est-à-dire: indication et réponse).

1.3 Conformité

La présente norme ni ne spécifie de mises en œuvre individuelles ou de produits individuels ni ne contraint les mises en œuvre d'entités de couche application au sein des systèmes d'automation industriels.

Il n'y a pas de conformité d'équipement à la présente norme de définition de services de couche application. Au contraire, la conformité est obtenue par la mise en œuvre de protocoles conformes de couche Application qui satisfont aux services de couche Application de Type 7 tels que définis dans la présente norme.

2 Références normatives

Les documents de référence suivants sont indispensables pour l'application du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

CEI 60559, *Arithmétique binaire en virgule flottante pour systèmes à microprocesseurs*

CEI/TR 61158-1 (Ed.2.0), *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 1: Présentation et lignes directrices des séries CEI 61158 et CEI 61784*

CEI 61158-3-7, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 3-7: Définition des services de couche liaison de données – Éléments de Type 7*

CEI 61158-4-7, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 4-7: Spécification de protocole de couche liaison de données – Éléments de Type 7*

ISO/CEI 7498-1, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base – Partie 1: Le modèle de base*

ISO/CEI 7498-3, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base – Partie 3: Dénomination et adressage*

ISO/CEI 8822, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Définition du service de présentation*

ISO/CEI 8824, *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN-1): Spécification de la notation de base*

ISO/CEI 9545, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Structure de la couche application*

ISO/CEI 10731, *Technologie de l'information – Interconnexion de systèmes ouverts (OSI)– Modèle de référence de base – Conventions pour la définition des services OSI*

3 Termes, définitions, symboles, abréviations et conventions

Pour les besoins du présent document, les termes suivants, tels que définis dans ces publications s'appliquent.

3.1 Termes de l'ISO/CEI 7498-1

Pour les besoins du présent document, les termes suivants tels que définis dans l'ISO/CEI 7498-1 s'appliquent:

a) entité d'application
b) processus d'application
c) unité de donnée de protocole application
d) élément de service application
e) invocation d'entité d'application
f) invocation de processus d'application
g) transaction d'application
h) système ouvert réel
i) syntaxe de transfert

3.2 Termes de l'ISO/CEI 8822

a) syntaxe abstraite
b) contexte de présentation

3.3 Termes de l'ISO/CEI 9545

a) association d'applications
b) contexte d'application
c) nom de contexte d'application
d) invocation d'entité d'application
e) type d'entité d'application

f) invocation de processus d'application
g) type de processus d'application
h) élément de service application
i) élément de service de contrôle d'application

3.4 Termes de l'ISO/CEI 8824

a) élément de service de contrôle d'application
b) type

3.5 Termes liés à la couche liaison de données des bus de terrain

Les termes suivants tels que définis dans la CEI 61158-3-7 et la CEI 61158-4-7 s'appliquent.

a) DLPDU de réponse d'acquittement
b) cycle de base
c) transaction de base
d) bus administrateur (BA, bus-arbitrator)
e) champ de contrôle
f) adresse destination
e) segment de DL, liaison locale
g) identificateur de DLCEP
h) DLPDU d'identificateur de DLCEP
i) DLPDU d'indication de fin de transaction de message
j) variable identifiée (ou simplement « variable »)
k) identificateur de DLCEP non valide
l) macrocycle
m) identificateur de DLPDU du message
n) DLPDU de réponse de message
o) balayage périodique de variables
p) variable identifiée éditée
q) DLPDU de réponse à une demande
r) adresse source
s) variable identifiée abonnée
t) balayage déclenché de messages
u) balayage périodique déclenché de messages
w) balayage périodique déclenché de variables
x) balayage déclenché de variables
y) temps de changement
z) DLPDU de réponse de variable

Les symboles et abréviations suivants tels que définis dans la CEI 61158-3-7 et la CEI 61158-4-7 s'appliquent.

a) BA
b) B_Dat_Cons
c) B_Dat_Prod
d) B_Req1/2

e) ID_DAT
f) ID_MSG
g) ID_RQ1/2
h) PRT
i) Q_IDMSG
j) Q_IDRQ1/2
k) Q_Msg_Aper
l) Q_Req1/2
m) Q_RPRQ
n) RQ_Inhibit
o) RP_ACK
p) RP_DAT
q) RP_DAT_MSG
r) RP_DAT_RQ1/2
s) RP_DAT_RQ1/2_MSG
t) RP_MSG_ACK
u) RP_MSG_NOACK
v) RP_NAK
w) RP_END
x) STT
y) TI

3.6 Définitions spécifiques à la couche application des bus de terrain

Pour les besoins de la présente norme, les termes et définitions suivants s'appliquent.

3.6.1
protection d'accès
limitation de l'usage d'un objet d'application à un seul client

3.6.2
objet actif de commande de connexion
instance d'une certaine classe de la FAL qui abstrait la fonctionnalité d'interconnexion (en tant que Consommateur et Fournisseur) d'un appareil d'automation

NOTE L'abréviation «FAL» est dérivée du terme anglais développé correspondant «Fieldbus Application Layer»

3.6.3
table d'affectation d'adresses
mapping du stockage d'un objet de données E/S interne du client vers des éléments de données d'entrée et de sortie décentralisés

3.6.4
allouer
prendre une ressource à partir d'une zone commune et affecter cette ressource à l'usage exclusif d'une entité spécifique

3.6.5
application
fonction ou structure de données pour laquelle des données sont consommées ou produites

3.6.6
interopérabilité de la couche application
capabilité des entités d'application à effectuer des opérations coordonnées et coopératives en utilisant les services de la FAL

NOTE L'abréviation «FAL» est dérivée du terme anglais développé correspondant «Fieldbus Application Layer»

3.6.7
objets d'application
classes d'objets multiples qui gèrent et assurent un échange de messages pendant le mode exécution à travers le réseau et à l'intérieur de l'appareil de réseau

3.6.8
processus d'application
partie d'une application distribuée sur un réseau, située sur un appareil et associée à une adresse non ambiguë

3.6.9
identificateur de processus d'application
distingue de multiples processus d'application utilisés dans un appareil

3.6.10
objet de processus d'application
composant d'un processus d'application qui est identifiable et accessible par l'intermédiaire d'une relation d'applications de la FAL

NOTE 1 Les définitions d'objet de processus d'application se composent d'un ensemble de valeurs pour les attributs de leur classe (voir la définition de "Définition de classe d'objets de processus d'application"). Les définitions d'objet de processus d'application peuvent être accédées à distance à l'aide des services de l'ASE "Object Management" (gestion d'objet) de la FAL. Les services de gestion d'objet de la FAL peuvent être utilisés pour charger ou mettre à jour des définitions d'objet, pour lire des définitions d'objet, et pour créer et supprimer de manière dynamique des objets d'application et leurs définitions correspondantes.

NOTE 2 L'abréviation «FAL» est dérivée du terme anglais développé correspondant «Fieldbus Application Layer»

NOTE 3 L'abréviation «ASE» est dérivée du terme anglais développé correspondant «Application Service Element»

3.6.11
classe d'objets de processus d'application
classe d'objets de processus d'application définie en termes de l'ensemble de leurs attributs et services accessibles par le réseau

3.6.12
relation d'applications
association de type coopératif entre deux ou plusieurs invocations d'entités d'application, dans un but d'échange d'informations et de coordination de leur fonctionnement conjoint

NOTE Cette relation est activée soit par l'échange d'unités de données de protocole d'application, soit suite à des activités de pré-configuration.

3.6.13
élément de service application de relation d'applications
élément de service d'application qui fournit le moyen exclusif d'établir et de terminer toutes les relations d'applications

3.6.14
point d'extrémité de relation d'applications
contexte et comportement d'une relation d'applications tels que vus et maintenus par l'un des processus d'application impliqués dans la relation d'applications

NOTE Chaque processus d'application impliqué dans la relation d'applications maintient son propre point d'extrémité de relation d'applications.

3.6.15
attribut
description d'une caractéristique ou d'une fonction, visible par un observateur externe, d'un objet

NOTE Les attributs d'un objet contiennent des informations relatives aux parties variables d'un objet. En règle générale, ils fournissent les informations de statut ou régissent le fonctionnement d'un objet. Les attributs peuvent également influer sur le comportement d'un objet. Les attributs se répartissent en deux catégories: les attributs de classe et les attributs d'instance.

3.6.16
comportement
indication de la manière dont un objet réagit à des événements particuliers

3.6.17
numéro de bit
désigne le numéro d'un bit dans une chaîne de bits ou dans un octet

3.6.18
voie
simple liaison physique ou logique d'un objet d'application d'entrée ou de sortie d'un serveur au processus.

3.6.19
classe
ensemble d'objets qui représentent tous la même sorte de composant système

NOTE Une classe est une généralisation d'un objet; un modèle pour définir des variables et des méthodes. Tous les objets dans une classe ont une forme et un comportement identiques, mais contiennent en général des données différentes dans leurs attributs

3.6.20
attributs de classe
attribut qui est partagé par tous les objets au sein de la même classe

3.6.21
code de classe
identificateur unique affecté à chaque classe d'objets

3.6.22
service spécifique à une classe
service défini par une classe d'objets particulière pour accomplir une fonction requise qui n'est pas accomplie par un service commun

NOTE Un objet spécifique à une classe est unique pour la classe d'objets qui le définit.

3.6.23
client
a) objet qui utilise les services d'un autre objet (serveur) pour accomplir une tâche

b) initiateur d'un message auquel un serveur réagit

3.6.24
objets de communication
composants qui gèrent et assurent l'échange de messages pendant le mode d'exécution à travers le réseau

EXEMPLES: Objet Connection Manager (Gestionnaire de Connexion), objet Unconnected Message Manager (Gestionnaire de Message non Connecté) (UCMM) et objet Message Router (Routeur de Message).

3.6.25
identificateur de configuration
représentation d'une partie de données d'entrée/sortie d'un seul module d'un serveur d'entrée et/ou de sortie

3.6.26
connexion
liaison logique entre des objets d'application qui peuvent être au sein du même appareil ou dans des appareils différents

NOTE 1 Les connexions peuvent être point à point ou multipoint.

NOTE 2 La liaison logique entre le puits et la source d'attributs et services au niveau de différentes interfaces personnalisées d'ASE RT-Auto est appelée interconnexion. Il y a une distinction entre les interconnexions de données et d'événements. La liaison logique et le flux de données entre le puits et la source d'éléments de données d'automation est appelée interconnexion de données. La liaison logique et le flux de données entre le puits (méthode) et la source (événement) des services opérationnels sont appelés interconnexion d'événements.

3.6.27
voie de connexion
description d'une connexion entre un puits et une source d'éléments de données

3.6.28
identificateur de connexion (CID)
identificateur attribué à une émission associée à une connexion particulière entre producteurs et consommateurs, fournissant un nom pour une information d'application spécifique.

NOTE L'abréviation «CID» est dérivée du terme anglais développé correspondant «Connection IDentifier»

3.6.29
chemin de connexion
flux d'octets qui définit l'objet d'application auquel une instance de connexion s'applique

3.6.30
point de connexion
tampon qui est représenté comme une sous-instance d'un objet Assembly (Assemblage)

3.6.31
consommation
acte consistant à recevoir des données d'un producteur

3.6.32
consommateur
nœud ou puits qui reçoit des données d'un producteur

3.6.33
application consommatrice
application qui consomme des données

3.6.34
consumerID
identificateur non ambigu relevant du domaine d'application de l'ACCO, attribué par le consommateur pour reconnaître les données internes d'un collecteur d'interconnexion configuré

NOTE L'abréviation «ACCO» est dérivée du terme anglais développé correspondant «Active connection control object».

3.6.35
commandes de contrôle
invocations d'action transférées du client au serveur pour effacer des sorties, geler des entrées et/ou synchroniser des sorties

3.6.36
trajet d'acheminement
flux unidirectionnel d'unités APDU, d'un bout à l'autre d'une relation entre applications

NOTE L'abréviation «APDU» est dérivée du terme anglais développé correspondant «Application Protocol Data Unit».

3.6.37
cyclique
répétitif de manière régulière

3.6.38
cohérence de données
moyen pour une émission et un accès cohérents de l'objet de donnée d'entrée ou de sortie au sein du client et du serveur et entre eux

3.6.39
AR dédiée
AR utilisée directement par l'utilisateur de la FAL

NOTE 1 Sur les AR dédiées, seuls l'en-tête de la FAL et les données utilisateur sont transférés.

NOTE 2 L'abréviation «AR» est dérivée du terme anglais développé correspondant «Association Relationship».

NOTE 3 L'abréviation «FAL» est dérivée du terme anglais développé correspondant «Fieldbus Application Layer».

3.6.40
appareil
matériel physique connecté à la liaison

NOTE Un appareil peut contenir un ou plusieurs nœuds.

3.6.41
profil d'appareil
ensemble d'informations et de fonctionnalités dépendant de l'appareil qui garantit la cohérence entre des appareils similaires appartenant au même type d'appareil

3.6.42
nœud d'extrémité
nœud producteur ou consommateur

3.6.43
point d'extrémité
l'une des entités communicantes impliquées dans une connexion

3.6.44
erreur
discordance entre une valeur ou un état calculé(e), observé(e) ou mesuré(e) et la valeur ou l'état spécifié(e) ou théoriquement correct(e)

3.6.45
classe d'erreur
regroupement général pour des définitions d'erreurs connexes et des codes d'erreurs correspondants

3.6.46
code d'erreur
identification d'un type spécifique d'erreur dans une classe d'erreurs

3.6.47
événement
instance d'un changement de conditions

3.6.48
variable FIFO
classe d'objets variables composée d'un jeu d'éléments de type homogène, dans laquelle le premier élément écrit est le premier élément qui peut être lu

NOTE 1 Dans les bus de terrain, un seul élément complet peut être transféré suite à une invocation de service.

NOTE 2 L'abréviation «FIFO» est dérivée du terme anglais développé correspondant «First-in, First-out».

3.6.49
trame
synonyme déconseillé de DLPDU

3.6.50
groupe
a) <généralités> terme général pour un ensemble d'objets.

b) <adressage> lors de la description d'une adresse, adresse qui identifie plus d'une entité.

3.6.51
interface
a) frontière partagée entre deux unités fonctionnelles, définies par les caractéristiques fonctionnelles, caractéristiques de signal ou autres caractéristiques selon le cas approprié.

b) ensemble d'attributs et de services de classe de la FAL qui représente une vue spécifique sur la classe de la FAL.

NOTE L'abréviation «FAL» est dérivée du terme anglais développé correspondant «Fieldbus Application Layer»

3.6.52
langage de définition d'interface
syntaxe et sémantique de description de paramètres de service de façon structurée

NOTE Cette description correspond à l'entrée du modèle ORPC, spécifiquement pour le protocole filaire ORPC.

Note 1 à l'article: L'abréviation «ORPC» est dérivée du terme anglais développé correspondant «Object Remote Procedure Call»

3.6.53
interface pointer (pointeur d'interface)
Attribut clé pour adresser sans ambiguïté une instance d'interface d'objet

3.6.54
invocation
acte consistant à utiliser un service ou une autre ressource d'un processus d'application

NOTE Chaque invocation représente un fil (thread) de commande distinct qui peut être décrit par son contexte. Une fois que le service s'achève ou que l'utilisation de la ressource est libérée, l'invocation cesse d'exister. Pour des invocations de service, un service a été lancé, mais n'est pas encore achevé s'appelle invocation de service en cours. Aussi, pour des invocations de service, un "Invoke ID" peut être utilisé pour identifier sans ambiguïté l'invocation de service et la différencier d'autres invocations de service en cours.

3.6.55
index (indice)
adresse d'un objet au sein d'un processus d'application

3.6.56
instance
occurrence physique effective d'un objet d'une classe, permettant d'identifier un objet parmi d'autres au sein de la même classe d'objets

EXEMPLE «California» (Californie) est une instance de l'état de la classe d'objets.

NOTE Les termes "objet", "instance" et "instance d'objet" sont utilisés pour se référer à une instance spécifique.

3.6.57
attributs d'instance
attribut qui est propre à une instance d'objet et n'est pas partagé par la classe d'objets

3.6.58
instancié
objet qui a été créé dans un appareil

3.6.59
appareil logique
certaine classe de la FAL qui abstrait un composant logiciel ou un composant micrologiciel comme une fonction monobloc autonome d'un appareil d'automation

NOTE L'abréviation «FAL» est dérivée du terme anglais développé correspondant «Fieldbus Application Layer».

3.6.60
ID de fabricant
identification de chaque fabrication de produit par un numéro unique

3.6.61
informations de gestion
informations accessibles par le réseau qui prennent en charge la gestion du fonctionnement du système de bus de terrain, y compris la couche application

NOTE La gestion inclut des fonctions telles que la commande, la surveillance et le diagnostic.

3.6.62
membre
partie d'un attribut qui est structurée comme un élément d'une matrice

3.6.63
méthode
synonyme pour un service opérationnel qui est fourni par l'ASE serveur et invoqué par un client

NOTE L'abréviation «ASE» est dérivée du terme anglais développé correspondant «Application Service Element»

3.6.64
module
composant matériel ou logique d'un appareil physique

3.6.65
connexion multipoint
connexion d'un nœud à plusieurs autres nœuds

NOTE Les connexions multipoint permettent aux messages d'un seul producteur d'être reçus par plusieurs nœuds consommateurs.

3.6.66
réseau
ensemble de nœuds reliés par un certain type de support de communication, notamment des répétiteurs intermédiaires, ponts, routeurs et passerelles de couches inférieures

3.6.67
objet
représentation abstraite d'un composant particulier au sein d'un appareil, habituellement un ensemble de données connexes (sous la forme de variables) et de méthodes (procédures) pour opérer sur les données en question qui ont une interface et un comportement clairement définis

3.6.68
appel de procédure distante orientée objet
modèle relatif à l'invocation de méthode à distance, orientée objet ou basée sur un composant

3.6.69
service spécifique à un objet
service propre à la classe d'objets qui le définit

3.6.70
initiateur
client qui est chargé d'établir un chemin de connexion vers la cible

3.6.71
homologue
rôle d'un point d'extrémité de l'AR dans lequel il est capable d'agir en tant que client et serveur

NOTE L'abréviation «AR» est dérivée du terme anglais développé correspondant «Association Relationship».

3.6.72
appareil physique
appareil d'automation ou autre appareil de réseau

3.6.73
connexion point-à-point
connexion qui existe entre exactement deux objets d'application

3.6.74
point d'extrémité prédéfini de l'AR
point d'extrémité de l'AR qui est défini localement à l'intérieur d'un appareil sans utilisation du service Create

NOTE 1 Les AR prédéfinies qui ne sont pas préétablies sont établies avant leur utilisation.

NOTE 2 L'abréviation «AR» est dérivée du terme anglais développé correspondant «Association Relationship».

3.6.75
point d'extrémité préétabli de l'AR
point d'extrémité de l'AR qui est mis dans un état établi pendant la configuration des AE qui commandent ses points d'extrémité

NOTE 1 L'abréviation «AR» est dérivée du terme anglais développé correspondant «Association Relationship».

NOTE 2 L'abréviation «AE» est dérivée du terme anglais développé correspondant «Application Entity».

3.6.76
données de processus
objet(s) déjà prétraité(s) et transféré(s) de façon acyclique à des fins d'informations ou de traitement supplémentaire

3.6.77
produire
acte consistant à envoyer des données devant être reçues par un consommateur

3.6.78
producteur
nœud qui est chargé d'envoyer des données

3.6.79
fournisseur
source d'une connexion de données

3.6.80
éditeur, publicateur
rôle d'un point d'extrémité de l'AR qui émet des APDU sur le bus de terrain en vue de leur consommation par un ou plusieurs abonnés

NOTE 1 Un éditeur peut ne pas être conscient de l'identité ou du nombre d'abonnés et il peut éditer ses APDU à l'aide d'une AR dédiée.

NOTE 2 L'abréviation «AR» est dérivée du terme anglais développé correspondant «Association Relationship»

NOTE 3 L'abréviation «APDU» est dérivée du terme anglais développé correspondant «Application Protocol Data Unit»

3.6.81
ressource
fonction de traitement d'informations d'un sous-système

3.6.82
serveur
a) rôle d'un AREP dans lequel il retourne au client qui a initié la demande une APDU de réponse de service confirmé

b) objet qui fournit des services à un autre objet (client)

NOTE 1 L'abréviation «AREP» est dérivée du terme anglais développé correspondant «Application Relationship End Point».

NOTE 2 L'abréviation «APDU» est dérivée du terme anglais développé correspondant «Application Protocol Data Unit».

3.6.83
service
opération ou fonction qu'un objet et/ou une classe d'objets réalise(nt) à la demande d'un autre objet et/ou d'une autre classe d'objets

3.6.84
abonné
rôle d'un AREP dans lequel il reçoit des APDU produites par un éditeur

NOTE 1 L'abréviation «AREP» est dérivée du terme anglais développé correspondant «Application Relationship End Point».

NOTE 2 L'abréviation «APDU» est dérivée du terme anglais développé correspondant «Application Protocol Data Unit».

3.7 Abréviations et symboles

AE Application entity (Entité d'application)

AL Application layer (Couche d'application)

ALME Application layer management entity (Entité de gestion de couche Application)

ALP Application layer protocol (Protocole de couche application)

APO Application object (Objet d'application)

AP Application process (Processus d'application)

APDU Application protocol data unit (Unité de donnée de protocole application)

API Application process identifier (Identificateur de processus d'application)

AR Application relationship (Relation d'applications)

AREP Application Relationship End Point (Point d'extrémité de relation d'applications)

ASCII American Standard Code for Information Interchange (Code américain normalisé pour l'échange d'informations)

ASE Application service element (Élément de service d'application)

CID Connection ID (Identifiant de connexion)

CIM Computer integrated manufacturing (Production intégrée par ordinateur)

CIP Control and Information Protocol (Protocole de commande et d'informations)

Cnf Confirmation

COR Connection originator (Initiateur de connexion)

CR Communication relationship (Relation de communication)

CREP Communication Relationship End Point (Point d'extrémité de relation de communication)

DL- (comme préfixe) de liaison de données

DLC Data Link Connection (Connexion de liaison de données)

DLCEP Data Link Connection End Point (Point d'extrémité de connexion de liaison de données)

DLL Data Link Layer (Couche liaison de données)

DLM Data Link-management (Gestion de liaison de données)

DLSAP Data Link Service Access Point (Point d'accès au service de liaison de données)

DLSDU DL-service-data-unit (Unité de données de service liaison de données)

DNS Domain Name Service (Service de noms de domaine)

DP Decentralised Peripherals (Périphériques décentralisés)

FAL Fieldbus application layer (Couche application des bus de terrain)

FIFO First in first out (premier entré, premier sorti)

HMI/IHM Human-Machine Interface (Interface homme-machine)

ID Identifier (Identificateur)

IDL Interface Definition Language (Langage de définition d'interface)

CEI Commission électrotechnique internationale

Ind Indication

IP Internet Protocol (Protocole Internet)

ISO International Organization for Standardization (Organisation internationale de normalisation)

LDev Logical Device (Appareil logique)

LME Layer management entity (Entité de gestion de couche)

ORPC Object Remote Procedure Call (Appel de procédure distante orientée objet)

OSI Open Systems Interconnect (Interconnexion des systèmes ouverts)

PDev Physical Device (Appareil physique)

PDU Protocol data unit (Unité de données de protocole)

PL Physical Layer (Couche physique)

QoS Quality of Service (Qualité de service)

Req Request (Demande)

Rsp Response (Réponse)

RT Runtime (Durée d'exécution)

SAP Service access point (Point d'accès au service)

SCL Security Level (Niveau de sécurité)

SDU Service Data Unit (Unité de données de service)

SMIB System Management Information Base (Base d'informations de gestion de système)

SMK System Management Kernel (Noyau de gestion de système)

STD State transition diagram (diagramme des transitions d'état, diagramme états-transition), utilisé pour décrire le comportement d'un objet

S-VFD Simple Virtual Field Device (Appareil de terrain virtuel simple)

VAO Variable Object (Objet Variable)

3.8 Conventions

3.8.1 Vue d'ensemble

La couche FAL est définie comme étant un ensemble d'ASE orientés objet. Chaque ASE est spécifié dans un paragraphe distinct. Chaque spécification d'ASE est constituée de deux parties, à savoir sa spécification de classe et sa spécification de service.

La spécification de classe définit les attributs de la classe. Les attributs sont accessibles à partir des instances de la classe qui utilise les services de l'ASE Object Management spécifiés dans l'Article 5 de la présente Norme. La spécification de service définit les services qui sont fournis par l'ASE.

3.8.2 Conventions pour les définitions de classes

Les définitions de classes sont décrites à l'aide de modèles. Chaque modèle se compose d'une liste d'attributs associés à la classe. La forme générale du modèle est montrée ci-dessous:

FAL ASE: **Nom de l'ASE**

CLASS: **Nom de la classe**

CLASS ID: #

PARENT CLASS: nom de la classe mère

ATTRIBUTES:

1 (o) Key Attribute (Attribut clé): identificateur numérique

2 (o) Key Attribute: nom

3 (m) Attribute (Attribut): nom d'attribut(valeurs)

4 (m) Attribute: nom d'attribut(valeurs)

4.1 (s) Attribute: nom d'attribut(valeurs)

4.2 (s) Attribute: nom d'attribut(valeurs)

4.3 (s) Attribute: nom d'attribut(valeurs)

5. (c) Constraint (Contrainte): expression de la contrainte
5.1 (m) Attribute: nom d'attribut(valeurs)
5.2 (o) Attribute: nom d'attribut(valeurs)
6 (m) Attribute: nom d'attribut(valeurs)
6.1 (s) Attribute: nom d'attribut(valeurs)
6.2 (s) Attribute: nom d'attribut(valeurs)

SERVICES:

1 (o) OpsService: nom du service
2. (c) Constraint: expression de la contrainte
2.1 (o) OpsService: nom du service
3 (m) MgtService: nom du service

(1) La rubrique "FAL ASE:" est le nom de l'élément ASE de la couche FAL (FAL ASE) qui fournit les services pour la classe spécifiée.

(2) La rubrique "CLASS:" est le nom de la classe étant spécifiée. Tous les objets définis à l'aide de ce modèle seront une instance de cette classe. La classe peut être spécifiée par la présente norme ou par un utilisateur de la présente norme.

(3) La rubrique "CLASS ID:" est un numéro qui identifie la classe étant spécifiée. Ce numéro est unique au sein de l'ASE de la FAL qui fournira les services pour cette classe. Lorsqu'il est qualifié par l'identité de son ASE de la FAL, il identifie sans ambiguïté la classe relevant du domaine d'application de la FAL. La valeur "NULL" indique que la classe ne peut pas être instanciée. Les Class ID (identificateurs de classe) compris entre 1 et 255 sont réservés par la présente norme pour identifier des classes normalisées. Ils ont été affectés pour maintenir la compatibilité avec des normes nationales existantes. Les CLASS ID compris entre 256 et 2048 sont alloués pour identifier les classes définies par l'utilisateur.

(4) La rubrique "PARENT CLASS:" est le nom de la classe mère pour la classe étant spécifiée. Tous les attributs définis pour la classe mère et hérités par celle-ci sont hérités pour la classe définie, et ils n'ont donc pas à être redéfinis dans le modèle pour cette classe.

NOTE La classe mère "TOP" indique que la classe étant définie est une définition de classe initiale. La classe mère "TOP" est utilisée comme point de départ à partir duquel toutes les autres classes sont définies. L'usage de "TOP" est réservé pour les classes définies par la présente norme.

(5) L'étiquette "ATTRIBUTES" indique que les entrées suivantes sont des attributs définis pour la classe.

a) Chacune des entrées d'attribut contient un numéro de ligne dans la colonne 1, un indicateur dans la colonne 2: obligatoire (m) / facultatif (o) / conditionnel (c) / sélecteur (s), une étiquette de type d'attribut dans la colonne 3, un nom ou une expression conditionnelle dans la colonne 4, et, facultativement, une liste de valeurs énumérées dans la colonne 5. Dans la colonne suivant la liste de valeurs, la valeur par défaut pour l'attribut peut être spécifiée.

b) Les objets sont normalement identifiés par un identificateur numérique et/ou par un nom d'objet. Dans les modèles de classe, ces attributs-clés sont définis sous l'attribut-clé.

c) Le numéro de ligne définit la séquence et le niveau d'imbrication de la ligne. Chaque niveau d'imbrication est identifié par un point (period). L'imbrication est utilisée pour spécifier

i) des champs d'un attribut structuré (4.1, 4.2, 4.3),

ii) des attributs conditionnés à un énoncé de contrainte (5). Les attributs peuvent être obligatoires (5.1) ou facultatifs (5.2) si la contrainte est vraie. Tous les

attributs facultatifs n'exigent pas d'énoncés de contraintes comme le fait l'attribut défini en 5.2,

iii) les champs sélection d'un attribut de type choix (6.1 et 6.2).

(6) L'étiquette "SERVICES" indique que les entrées suivantes sont des services définis pour la classe.

a) Un (m) dans la colonne 2 indique que le service est obligatoire pour la classe, alors qu'un (o) indique qu'il est facultatif. Un (c) dans cette colonne indique que le service est conditionnel. Lorsque tous les services définis pour une classe sont définis comme étant facultatifs, l'un au moins doit être sélectionné quand une instance de la classe est définie.

b) L'étiquette "OpsService" désigne un service opérationnel (1).

c) L'étiquette "MgtService" désigne un service de gestion (2).

d) Le numéro de ligne définit la séquence et le niveau d'imbrication de la ligne. Chaque niveau d'imbrication est identifié par un point (period). L'imbrication dans la liste de services sert à spécifier des services conditionnés à un énoncé de contrainte.

3.8.3 Conventions pour les définitions de service

3.8.3.1 Généralités

La présente norme utilise les conventions descriptives données dans l'ISO/CEI 10731.

Le modèle de service, les primitives de service et les diagrammes temps-séquence utilisés sont des descriptions totalement abstraites; ils ne constituent pas une spécification pour une mise en œuvre.

3.8.3.2 Paramètres de service

Les primitives de service sont utilisées pour représenter les interactions entre utilisateur de service et fournisseur de service (ISO/CEI 10731). Elles acheminent des paramètres qui indiquent des informations disponibles dans l'interaction entre utilisateur et fournisseur.

NOTE 1 Voir la note en 3.8.3.3 relative à la non-inclusion des paramètres de service qui sont adaptés à une spécification de protocole, une spécification d'interface de programmation ou une spécification de mise en œuvre, mais pas à une définition de service abstrait.

La présente norme utilise un format de tabulaire pour décrire les paramètres de composants des primitives de service. Les paramètres qui s'appliquent à chaque groupe de primitives de service sont définis dans des tableaux dans toute la suite de la présente norme. Chaque tableau comporte jusqu'à six colonnes: une colonne pour le nom du paramètre de service et une colonne pour chacune des primitives et chacun des sens de transfert de paramètres utilisés par le service. Les six colonnes possibles sont:

1) le nom de paramètre;
2) les paramètres d'entrée de la primitive "request";
3) les paramètres de sortie de la primitive "request";

NOTE 2 Cette fonctionnalité est rarement utilisée. Sauf spécification contraire, les paramètres de la primitive "request" sont des paramètres d'entrée.

4) les paramètres de sortie de la primitive "indication";
5) les paramètres d'entrée de la primitive "response"; et
6) les paramètres de sortie de la primitive "confirm".

NOTE 3 Les primitives "request" (demande), "indication", "response" (réponse) et "confirm" (confirmation) sont aussi connues respectivement comme des primitives requestor.submit, acceptor.deliver, acceptor.submit et requestor.deliver (voir ISO/CEI 10731).

Un paramètre (ou un composant de celui-ci) est énuméré dans chaque ligne de chaque matrice. Dans les colonnes appropriées de la primitive de service, un code est utilisé pour spécifier le type d'usage du paramètre sur la primitive spécifiée dans la colonne:

M le paramètre est obligatoire pour la primitive

U le paramètre est une option de l'utilisateur et peut ou peut ne pas être fourni, en fonction de l'usage dynamique de l'utilisateur du service. Lorsqu'il n'est pas fourni, une valeur par défaut est supposée pour le paramètre.

C le paramètre est conditionné à d'autres paramètres ou à l'environnement de l'utilisateur du service.

— (blanc/vide) le paramètre n'est jamais présent.

S le paramètre est un élément sélectionné.

Certaines entrées sont par ailleurs qualifiées par des éléments entre parenthèses. Ces entrées peuvent être:

a) une contrainte spécifique à un paramètre:
"(=)" indique que le paramètre équivaut, du point de vue de la sémantique, au paramètre de la primitive de service située immédiatement à sa gauche dans la matrice;

b) une indication qu'une certaine note s'applique à l'entrée:
"(n)" indique que la note "n" suivante contient des informations complémentaires relatives au paramètre et à son utilisation.

3.8.3.3 Procédures de services

Les procédures sont définies en termes

- d'interactions entre entités d'application par l'échange d'unités de données de protocole d'application de bus de terrain, et
- d'interactions entre un fournisseur de service de couche application et un utilisateur de service de couche application dans le même système par l'invocation de primitives de service de couche application.

Ces procédures sont applicables à des instances de communication entre systèmes qui prennent en charge des services de communication à contrainte temporelle à l'intérieur de la couche Application de bus de terrain.

NOTE La série de normes CEI 61158-5 définit des ensembles de services abstraits. Il ne s'agit ni de spécifications de protocole, ni de spécifications de mise en œuvre, ni de spécifications d'interface de programmation concrète. Par conséquent, la mesure dans laquelle les procédures de service peuvent être mandatées dans les parties de la CEI 61158-5 est soumise à des restrictions. Les aspects de protocole qui peuvent varier parmi différentes spécifications de protocole ou différentes mises en œuvre qui instancient les mêmes services abstraits ne sont pas appropriés à être inclus dans ces définitions de service, excepté au niveau d'abstraction qui est nécessairement commun à toutes ces expressions.

Par exemple, le moyen par lequel des fournisseurs de service apparient des PDU de demande et de réponse est approprié pour une spécification dans une norme de spécification de protocole de la CEI 61158-6, mais pas dans une norme de définition de services abstraits de la CEI 61158-5. De même, les méthodes de mise en œuvre locales par lesquelles un fournisseur de service ou un utilisateur de service apparie des primitives "request" et "confirm(ation)' ou des primitives "indication" et "response" sont adaptées à une spécification de mise en œuvre ou une spécification d'interface de programmation, mais pas à une norme de service abstrait ou une norme de protocole, excepté à un niveau d'abstraction qui est nécessairement commun à tous les modes de réalisation de la norme de spécification. Dans tous les cas, la définition abstraite n'est pas autorisée à spécifier outre mesure la réalisation d'instanciation plus concrète.

Des informations supplémentaires relatives aux procédures de service conceptuel d'une mise en œuvre d'un protocole qui réalise les services de l'une des définitions de service abstrait de la CEI 61158-5 peuvent être consultées dans la CEI/TR 61158-1, 9.6.

4 Concepts

Les concepts et modèles communs utilisés pour décrire le service de couche Application dans ce bus de terrain sont détaillés dans la CEI/TR 61158-1, Article 9.

5 ASE "Data type"

5.1 Vue d'ensemble

Une vue d'ensemble de l'ASE de type de données et des relations entre types de données est fournie dans la CEI/TR 61158-1, 10.1

5.2 Définition formelle des objets de types de données

Le modèle utilisé pour décrire la classe de types de données est détaillé dans la CEI/TR 61158-1, 10.2. Il inclut la structure spécifique à l'élément ASE et la définition de ses attributs.

5.3 Types de données définis de la FAL

5.3.1 Types Fixed length (longueur fixe)

Les types de primitives qui sont sélectionnés et leurs caractéristiques sont principalement spécifiés dans la norme ISO 8824.

Les types de données utilisés et les divergences entre la norme 8824 et la présente norme sont énumérés ci-après.

5.3.1.1 Boolean

Conforme à la norme 8824.

5.3.1.2 FLOATING POINT

CLASS:			**Data type**
ATTRIBUTES:			
1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	FloatingPoint
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	Boolean

Ce type de primitive définit la représentation de la valeur pour les nombres réels positifs et négatifs, y compris, zéro, moins l'infini et plus l'infini.

Les valeurs aux bornes (0, +• -•), ainsi que les différents formats de représentation sont définis dans la CEI 60559.

Pour ce type, l'attribut Octet length, de type boolean, indique si le format de représentation est de simple précision "4 octets" (FALSE) ou de double précision "8 octets" (TRUE).

5.3.1.3 BCD

Conforme à l'ISO/CEI 8824.

5.3.1.4 INTEGER

CLASS:	**Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	Not used
2	Data type Name	=	Integer
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	n

La définition de ce type de primitive est la même que la définition de type integer dans la norme ISO 8824. Pour ce type, l'attribut Octet length définit le nombre de bits qui sont nécessaires pour avoir une représentation en complément à deux de toutes les valeurs possibles.

Exemple : "Octet lentgh" = 1 si -128 ≤ n ≤ 127 ; 2 si -32768 ≤ n ≤ 32767; .....

5.3.1.5 UNSIGNED

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	Unsigned
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	n

La définition de ce type de primitive est la même que la définition de type integer dans l'ISO 8824 à l'exclusion des nombres négatifs. Pour ce type, l'attribut Octet Length définit le nombre de bits qui sont nécessaires pour avoir une représentation en complément à deux de toutes les valeurs possibles.

Exemple : 1 : pour unsigned8 ; 2 : pour unsigned16 ; 4 : pour unsigend32 ; 8 pour unsigned64

5.3.1.6 GENERALISED TIME

Conforme à l'ISO/CEI 8824.

5.3.1.7 BINARY TIME

CLASS: **Data type**

ATTRIBUTES:

1	Data type Numeric Identifier	=	non utilisé
2	Data type Name	=	BinaryTime
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	n

Ce type de primitive est défini comme étant le type non signé de la présente norme, dont la valeur correspond à un nombre de temps élémentaires.

Pour ce type, chaque valeur de l'attribut *Octet Length* définit la valeur d'un temps élémentaire ainsi qu'un nombre de bits utilisés pour représenter toute valeur temporelle binaire possible (voir Tableau 1).

Tableau 1 – Codage binaire temporel

Taille	10 ms	0,1 ms	1 ms	10 ms
16 Bits	0	1	2	3
32 Bits	4	5	6	7
48 Bits	8	9	—	—

5.3.2 Types String (chaîne)

5.3.2.1 BIT STRING

Conforme à l'ISO/CEI 8824.

5.3.2.2 OCTET STRING

Conforme à l'ISO/CEI 8824.

5.3.2.3 VISIBLE STRING

Conforme à l'ISO/CEI 8824.

5.3.3 Types Array (matrice)

Ils sont définis comme des types de données définis par l'utilisateur pour l'application spécifique. Les types de données d'utilisateurs sont pris en charge par la présente norme comme des instances des classes de types de données.

Les types définis par l'utilisateur sont spécifiés de la même manière dont tous les objets de FAL sont spécifiés. Ils sont définis en donnant des valeurs aux attributs spécifiés pour leur classe.

5.3.4 Types Structure

Ils sont définis comme des types de données définis par l'utilisateur pour l'application spécifique. Les types de données d'utilisateurs sont pris en charge par la présente norme comme des instances des classes de types de données.

Les types définis par l'utilisateur sont spécifiés de la même manière dont tous les objets de FAL sont spécifiés. Ils sont définis en donnant des valeurs aux attributs spécifiés pour leur classe.

6 Spécification du modèle de communication

6.1 Concepts

6.1.1 Environnement AL

Ce modèle général présente la relation entre l'environnement AL et les services de la couche application, qui est l'objet de la présente norme.

L'environnement AL représente à la fois la commande de processus et la fabrication de pièces discrètes. Dans ces systèmes, les appareils de fabrication sont contrôlés par un système qui exécute les fonctions d'automation.

6.1.2 Services d'application

Les services d'application peuvent être divisés en deux ensembles. Voir Figure 1.

La norme spécifie l'ensemble de services d'application MPS (Manufacturing Periodic/aperiodic Services, c'est-à-dire Services Périodiques/ non périodiques de Fabrication). L'autre ensemble de services correspond à un sous-ensemble (subMMS) de celui spécifié dans la norme MMS.

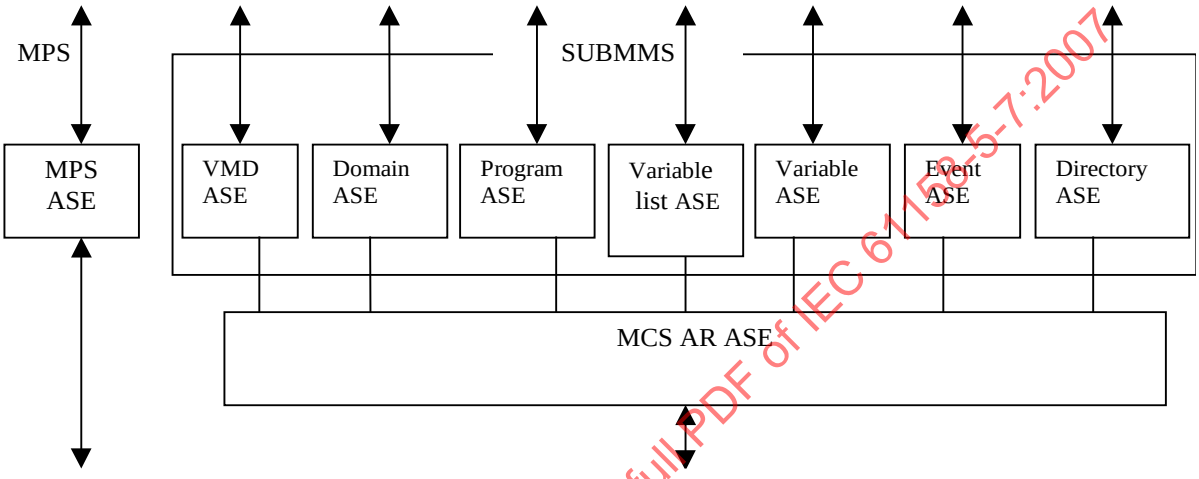
Les services MPS sont particulièrement bien adaptés aux exigences de temps réel d'une application distribuée dans la mesure où ils prennent en charge:

– La mise à jour de diffusion périodique d'une base de données distribuée dans les différents appareils du système de commande.

– L'élaboration de qualificatifs temporels associés à la base de données.

– L'élaboration de qualificatifs de cohérence associés aux ensembles de données à l'intérieur de cette base de données.

SubMMS et MCS sont essentiellement considérés comme satisfaisant aux exigences relatives à la gestion du mode de configuration et d'exploitation au sein de l'application distribuée.



Légende

Anglais	Français
MPS ASE	ASE MPS (services périodiques/non périodiques industriels)
MCS AR ASE	ASE d'AR MCS (association de relations du service commun de messagerie)
MPS	MPS (services périodiques/non périodiques industriels)
SUBMMS	SUBMMS (sous-spécification de messagerie industrielle)
VMD ASE	ASE "VMD "(Modèle virtuel d'appareil)
Domain ASE	ASE "Domain" (domaine)
Program ASE	ASE "Program" (programme)
Variable ASE	ASE "Variable"
Event ASE	ASE "Event " (événement)
Directory ASE	ASE "Directory" (répertoire)
Variable list ASE	ASE "Variable list" (liste de variables)

Figure 1 – Organisation des ASE et AR

Une application distribuée est caractérisée par diverses tâches de traitement d'informations sur différents appareils d'un système de communication pour les exigences de la vérification d'une application de commande de système de production.

La coopération entre les tâches appartenant à différents appareils est modélisée sous la forme d'interactions entre des processus d'application (AP).

Une application distribuée est constituée de deux ou plusieurs AP, chacun des appareils du système de communication pouvant en avoir zéro, un ou plusieurs.

Comme prévu par la présente norme, la communication entre deux AP passe par le réseau.

NOTE La décomposition d'une application distribuée en plusieurs AP au niveau d'un appareil est décidée par l'utilisateur en fonction de ses propres critères (Méthodologie, structuration fonctionnelle de l'application, paramétrage en fonctionnalités de services, etc.).

6.1.3 Processus d'application

6.1.3.1 Vue d'ensemble d'un processus d'application

Un AP est modélisé par une ou plusieurs tâches de traitement d'informations dans un appareil pour les exigences d'une fonction particulière de l'application distribuée.

Les aspects d'un AP, pris en compte par l'AL, ne concernent que ses capacités de communication qui sont modélisées par les entités d'application (AE, Application Entities). Chaque AP peut avoir au maximum une AE d'un type donné.

D'autre part, la coopération entre AP dans le cadre d'une application distribuée est effectuée via la mise en place des relations entre les invocations des AP (API) qui représentent leurs activités. Un AP peut, à un instant donné, désigner zéro une ou plusieurs API. En outre, chaque API a une utilisation particulière de l'AE (AEI) qui répond aux exigences de communication.

6.1.3.2 Modèle AP

6.1.3.2.1 Description du modèle AP

Le but du modèle AP est de montrer tous les aspects d'un AP qui sont pris en compte par la communication AL.

NOTE Les autres aspects concernant le type d'information, le traitement et l'exécution de cette dernière ne sont pas modélisés puisqu'ils ne sont pas impliqués au niveau de la communication.

***Class*: APPLICATION PROCESS (AP)**

Attribut-clé: AP title

Attribut: AP type

Attribut: List of Inherit from Application Entity

Attribut: List of Reference API

6.1.3.2.2 Attributs:

AP title:

Un titre est attribué à un AP pour l'identifier parmi tous les autres AP, d'une manière non ambiguë dans un réseau AL multisegment. Pour les exigences relatives à l'identification universelle d'un AP dans l'environnement AL, ce titre d'AP est impliqué en tant que constituant du nom ISO approprié au site utilisateur AL.

Cette identification peut être structurée à partir du nom de l'appareil auquel elle appartient.

NOTE Pour structurer cette identification, il convient que cet AP reste en permanence dans l'équipement.

AP type:

Le type d'un AP permet de désigner un ensemble d'AP auquel il appartient.

NOTE Ce type n'est pas couvert par un titre qui peut être manipulé dans l'environnement AL.

List of Inherit from Application Entity

La classe Application Process (c'est-à-dire Processus d'application) hérite d'une liste de classes Application Entity (c'est-à-dire Entité d'application) qui lui donne la capacité de communiquer dans l'environnement AL.

Ainsi, un AP a une ou plusieurs entités d'application, qui sont nécessairement de différents types.

List of Reference API:

Désigne les listes d'API qui modélisent les activités de l'AP. Cette liste peut varier dynamiquement, mais à un instant donné zéro, peut désigner une ou plusieurs API.

6.1.3.3 Modèle API

6.1.3.3.1 Description du modèle API

Un modèle API décrit les activités d'un AP.

***Class*: APPLICATION PROCESS INVOCATION (API)**
Key Attribute: API identification
Attribut: Reference Application Process
Attribute: *List of Reference* AEI

6.1.3.3.2 Attributs:

API identification:

Identifie une invocation de processus d'application d'une manière non ambiguë dans le domaine AP donné. Il est souligné que le domaine d'application de cette identification est d'ordre global pour l'AP et indépendant des règles pour la désignation du titre d'AP.

***Reference* Application process**

Le but de cet attribut est de montrer que cette API appartient à un AP donné.

La durée de vie d'une API est nécessairement inférieure ou égale à celle de l'AP auquel elle appartient.

List of Reference AEI:

Cette liste désigne les utilisations d'AE qui sont réservées pour les exigences de l'API.

Une API peut utiliser zéro, une ou plusieurs AEI pour chaque AE de l'AP auquel elle appartient.

Le nombre d'AEI référencées peut varier dynamiquement au fil du temps. Par conséquent, une API, à un moment donné, peut avoir une ou plusieurs AEI (une définition complète d'une AEI se trouve dans le paragraphe suivant).

6.1.4 Entité d'application

Une AE représente un ensemble de capacités de communication pour les exigences d'un AP. Ces capacités de communication sont définies par un ensemble d'Éléments de Service d'Application (ASE, Application Service Elements). Les utilisations particulières des capacités de communication d'une AE sont représentées par des invocations d'AE (AEI, AE invocations).

Une AEI modélise en conséquence les fonctions de communication et leur état pour les activités particulières d'une API. Les communications entre les API ont lieu par l'intermédiaire d'AEI qui, au besoin, peuvent ou peuvent ne pas être liées par des associations d'applications

(AA, Application Association) ou par des relations d'applications (AR, Application Relationship) dans le contexte de la couche application.

Une AEI dans l'environnement AL donne à une API la possibilité exclusive de communiquer avec ou sans association. Une AEI ne peut participer qu'à une et une seule application d'association.

Les communications sans association permettent à une AEI d'échanger des informations avec une ou plusieurs autres AEI.

Une application d'associations permet à une AEI de communiquer avec une ou plusieurs autres AEI dans le cadre d'une application de contexte (CA, Context Application) commune.

Ce contexte peut être négocié ou prédéfini par la configuration pour chaque AEI, pour une association point à point qui permet l'utilisation d'une paire d'AEI. Le contexte est nécessairement négocié au préalable pour chaque AEI.

L'état d'une AEI est donc directement lié au fait que celle-ci participe ou non à une association et à l'état de l'association si elle participe à une association négociée.

En général, on peut dire que la durée de vie d'une AEI est équivalente à celle du service échangé pour les communications sans association et équivalente à celle de l'association pour les communications avec associations. Cependant, pour une association négociée, la durée de vie de la paire d'AEI est allongée par leur ouverture et leur fermeture transitoires.

La durée de vie d'une AEI est commandée par l'API qu'elle représente dans l'environnement de la couche application. Une API, d'autre part, peut avoir une durée de vie beaucoup plus longue que celle de ses AEI, égale à celle de l'AP auquel elle appartient.

Des AR particulières sont prédéfinies par configuration, ces AR ne sont utilisées que par l'ASE du MPS pour échanger des données simples (variables). Tous les paramètres relatifs à la variable sont préétablis à la configuration, ces paramètres ne changent jamais. Seule une nouvelle configuration peut annuler ou modifier ces AR particulières. L'identification de cette AR particulière est effectuée par un numéro unique sur le réseau, ce numéro fait référence à l'ensemble des paramètres de l'AR et des ressources DLL impliquées dans la communication. Ce numéro similaire à un DLCEP Id est appelé Identifier (Identificateur).

6.1.5 Modèle de l'AE

6.1.5.1 Description du modèle de l'AE

Description des capacités de communication offertes par une AE.

***Class*: APPLICATION ENTITY (AE)**
Attribute: AE type
Key Attribute: Qualifier
Attribut: List of Inherit from ASE
Attribute: List of Reference AEI

6.1.5.2 Attributs:

Qualifier:

Un "AE qualifier" (qualificatif d'AE) identifie une AE de façon non ambiguë dans le champ d'un processus d'application. La combinaison de cette AE qualifier avec l'AP title (titre d'AP) forme un AE title (titre d'AE) qui identifie cette AE de façon non ambiguë dans l'environnement AL.

Cet AE title permet la récupération des adresses de liaisons à partir du répertoire d'informations d'adressage.

Un AE title peut avoir "Synonyms of the AE title" (c'est-à-dire: Synonymes de titres d'AE) à distinguer d'une ou de plusieurs autres AE par les différents titres.

Une AE peut être identifiée de façon globale avec d'autres AE à l'aide d'une "désignation de groupe d'AE".

AE type:

L'attribut AE type (type d'AE) désigne un ensemble d'AE partageant les mêmes caractéristiques de communication. Différentes AE du même type héritent de la même liste d'éléments de service d'application.

List of Inherit from ASE

La classe Application Entity (c'est-à-dire Entité d'application (AE)) hérite d'une liste de classe ASE qui lui donne la spécificité de ses fonctions de communication. Une AE peut prendre en charge un ou plusieurs ASE qui sont nécessairement de différents types.

List of Reference AEI

Désigne la liste des AEI qui modélisent les utilisations d'AE. Cette liste qui peut varier dynamiquement peut désigner à un moment donné, zéro, une ou plusieurs AEI.

6.1.6 Modèle d'AEI

6.1.6.1 Description du modèle d'AEI

Un modèle AEI est utilisé pour décrire une utilisation particulière de l'AE.

***Class*: APPLICATION ENTITY INVOCATION (AEI)**
Key attribute (Attribut clé): AEI identification
Attribute (Attribut): Reference AE
Attribute: *Reference* API
Attribute: Local AE address
Attribute: Remote AE address
Attribute: Communication mode
[ASSOCIATED; NON-ASSOCIATED]
Constraint (Contrainte): Communication mode = ASSOCIATED
Attribute: Association state
[OPENING;
ESTABLISHED;
CLOSED]
Attribute: Association type
[NEGOTIATED; PRENEGOTIATED]
Attribute: Association type
[NEGOTIATED; PRENEGOTIATED]
Attribute: Local AEI access point
Attribute: Remote AEI access point
Attribute: *Inherit from* Application context

6.1.6.2 Attributs:

AEI identification:

Permet l'identification non ambiguë d'une invocation d'entité d'application dans le sous-champ d'une AE réservée à une API d'un AP donné.

La désignation non ambiguë d'une AEI dans l'environnement nécessite d'indiquer les paramètres suivants: AP title / API identification / AE qualifier /AEI identification.

Reference AE:

Cette référence est utilisée pour désigner l'objet AE auquel cette AEI appartient,

La durée de vie d'une AEI est nécessairement inférieure ou égale à celle de l'AE à laquelle elle appartient.

Reference API:

Cette référence permet de désigner l'objet API qui est représenté dans l'environnement AL par cette AEI. Une AEI peut représenter une et une seule API dans l'environnement AL.

Local AE address:

Correspond à l'adresse de liaison qui permet de localiser l'AE à laquelle cette AEI appartient.

Cette adresse est complétée à l'aide de fonctions de répertoire lors de la création de cette AEI et reste statique pendant sa durée de vie.

Remote AE address:

Correspond, soit à une adresse de liaison qui permet de localiser l'AE à laquelle correspond cette AEI, pour les échanges point à point, soit à une adresse de liaison qui permet de localiser un groupe d'AE auquel cette AEI correspond pour les échanges multipoints.

Cette adresse est remplie au niveau de l'AEI lors de l'ouverture d'une association avec une autre AEI différente ou lors de l'échange de données dans un mode non associé.

Communication mode [ASSOCIATED; NON-ASSOCIATED]

(Le mode de communication dans l'état ASSOCIATED indiquait que cette AEI permet à une API de communiquer dans un contexte d'application qui a été précédemment défini ou négocié. Le type d'échanges entre les AEI, possibles dans un tel mode de communication, est point à point ou multipoint.

Lorsque le mode de communication est NON-ASSOCIATED, cette AEI permet à une API de communiquer hors du contexte préétabli. La nature des échanges possibles dans un mode de communication non associé est point à point ou multipoint.

Association state

[OPENING;
ESTABLISHED;
CLOSING]:

Dans le mode associé, les possibilités de communication d'une AEI avec une autre AEI dépendent de l'état de l'association. Les différents états correspondent aux phases de la vie d'une application d'association:

OPENING:

Correspond à une AEI qui, à la demande de l'API qu'elle représente, négocie un contexte d'application avec une autre AEI, pour les exigences d'une association. Dans cet état, aucun échange ne peut avoir lieu en dehors de ceux qui sont réservés à la négociation du contexte.

ESTABLISHED:

Correspond à une AEI qui offre à l'API qu'elle représente, la possibilité de communiquer avec une autre API via l'une de ses AEI déterminées, dans le cadre d'un contexte d'application. Ce contexte d'application a été obtenu soit par la négociation entre la paire d'AEI, soit par prédéfinition locale au niveau des AEI.

CLOSING:

Correspond à une AEI qui, à la demande de l'API qu'elle représente, ou à laquelle elle a été associée, a été libérée de son contexte d'application.

Association type

[NEGOTIATED;
PRENEGOTIATED]

Ce type indique si l'association a été négociée ou pas. Cet attribut, s'il prend la valeur PRENEGOTIATED, impose à l'association de rester exclusivement dans l'état ESTABLISHED.

Local AEI access point:

Chaque AEI, dans le mode ASSOCIATED, peut être directement localisée par son point d'accès d'AEI.

Cette information d'adressage est remplie à l'aide des fonctions de répertoire lors de la création d'une instance d'AEI et reste statique pendant toute sa durée de vie.

AEI remote access point:

Chaque AEI qui est dans le mode ASSOCIATED, dispose de cette information d'adressage qui localise l'AEI avec laquelle elle est en correspondance pour des échanges point à point. Cette information est remplie pour une AEI à l'ouverture d'une association.

Cette information localise les AEI avec lesquels elle est en correspondance pour les échanges multipoints.

Inherit from Context Application

Cet héritage permet de désigner l'objet du contexte d'application qui conditionne les communications avec cette AEI.

Les attributs de ce contexte d'application sont évalués au cours de l'installation d'une AEI sur la base des valeurs proposées par l'utilisateur dans le mode associé négocié et le mode non associé, alors qu'ils sont remplis par la configuration dans le mode associé négocié au préalable.

6.1.7 Contexte d'application

6.1.7.1 Vue d'ensemble du contexte d'application

Une paire d'AEI exige une connaissance commune des règles qui régissent ses communications. Cet ensemble de règles est appelé "Context Application" (c'est-à-dire application de contextes). Le contexte d'application qui peut être pris en charge par une AEI est nécessairement dérivé à partir des possibilités offertes par les différents ASE possédés par l'AE à laquelle il appartient

6.1.7.2 Éléments de service application

6.1.7.2.1 Description de l'ASE

Les possibilités offertes par les différents ASE d'une AE sont définies par la spécification (dans ASN1) d'un ou plusieurs ensembles d'unités de données de protocole application (APDU, Application Protocol Data Unit) qui forment les syntaxes abstraites. En outre, l'utilisation d'une syntaxe abstraite pour dialoguer entre deux AE nécessite la mise en œuvre d'une syntaxe de transfert qui définit les règles d'encodage des APDU sur le bus.

NOTE Parmi les syntaxes abstraites associées à la spécification MMS de l'ASE, la syntaxe abstraite générale ISO-9506-MMS-1 peut être notée ainsi que d'autres syntaxes abstraites telles que celles destinées aux commandes numériques ISO-9506-NCCS-1 et à la commande de processus ISO-9506-PROCESS-1.

Classe: ELEMENT SERVICE APPLICATION

Attribut: ASE type

Attribut: List of Abstract syntax

Attribut: List of Transfer syntax

6.1.7.2.2 Attributs:

ASE type:

Plusieurs types d'ASE peuvent être satisfaits dans l'environnement AL. La présente norme est couverte par la définition de SubMMSE provenant de la spécification MMS de l'ASE définie par l'ISO.

Le service MCSE (Messaging Common Service Element «élément de service commun de messagerie») correspond à l'ASE de commande qu'il convient d'utiliser nécessairement pour projeter une autre application ASE sur la messagerie de la couche liaison. Cet élément de service a des règles d'encodage uniques qui sont universellement connues dans l'environnement de messagerie AL.

NOTE Les règles d'encodage du MCS ne sont pas mentionnées dans l'ASN1. Elles sont directement données dans la représentation utilisée pour le transfert.

List of Abstract Syntax:

Un élément de service application peut avoir une ou plusieurs syntaxes abstraites. Chacune d'elles correspond à un ensemble d'APDU décrit dans un module ASN1 universellement identifié dans l'environnement AL.

NOTE Par exemple, une syntaxe abstraite définie dans l'ASN1 présente l'apparence suivante:

<nom du module> <identificateur universel du module>

Definitions::= BEGIN{

IMPORTS <..>, <..>, ...FROM <module externe>

EXPORTS <..>, <..>, ...

PDU module ::= CHOICE {

{<nom de la production> [N°] <production>} +

}}

END

La présente norme considère l'AL SUBMMS-1, provenant de la base de références, comme l'une de ses syntaxes abstraites pour l'élément de service MMS.

List of Transfer syntax:

Une syntaxe de transfert définit les règles d'encodage utilisées de la syntaxe abstraite. Un ASE pourrait prendre en charge plusieurs syntaxes de transfert dans le cadre d'une AE, qui

pourraient être appliquées selon les exigences qui dépendent des possibilités de leurs correspondants.

Une des syntaxes de transfert prises en compte par cette norme d'environnement AL pour la mise en œuvre de la syntaxe abstraite AL, l'AL SUBMMS -1, est proposée par les règles d'encodage de base (BER, Basic Encoding Rules) de l'ISO.

Comme les syntaxes abstraites, les syntaxes de transfert sont universellement identifiées dans l'environnement AL.

6.1.7.3 Type de contexte d'application

6.1.7.3.1 Description du contexte d'application

Le contexte d'application, appartenant à une AEI, conditionne les éventuelles communications possibles avec cette AEI.Dans les informations contenues dans le contexte d'application, certaines sont spécifiques à l'architecture de l'AE, c'est-à-dire avec le type d'ASE reposant sur MCS ainsi qu'avec la syntaxe de transfert abstraite choisie. D'autres informations sont spécifiques au choix effectué sur le niveau de l'API et correspondant aux restrictions relatives à la syntaxe abstraite négociée et à la quantité de ressources de communication utilisées.

***Class*: APPLICATION CONTEXT**
Attribute: Context name
Attribut-clé: Context identification
Attribut: Reference ASE
Attribute: Abstract syntax
Attribute: Transfer Syntax
Constraint: Com. mode = ASSOCIATED
Attribute: Application reductions
Attribut: Service quality

6.1.7.3.2 Attributs:

Identification of the context:

Permet d'identifier un contexte d'application dans le champ d'un AE. Il a ainsi un rôle local exclusif.

Context name:

Utilisé pour identifier, dans l'environnement AL, les bases d'un contexte constitué de l'ASE choisie, d'une syntaxe abstraite et d'une syntaxe de transfert. Ce nom est utilisé au cours de la négociation de l'association ou lors d'un transfert de données en mode non associé.

Reference ASE:

Un des éléments de service application de l'AE qui est choisi par ce contexte d'application.

Abstract syntax:

Une syntaxe abstraite parmi celles de l'ASE dans le cadre de l'AE, qui est choisie par ce contexte.

Transfer syntax:

Syntaxe de transfert choisie pour coder les PDU de l'ASE, mise en œuvre dans le cadre de cette AE, qui est choisie par ce contexte.

Application reductions:

La syntaxe abstraite choisie au niveau d'un contexte d'application d'AEI pour un élément de service donné peut être utilisée d'une manière réduite. Cette possibilité est utilisée uniquement pour l'émission en mode associé.

Service quality:

Il s'agit de la qualité de service choisie.

6.1.7.4 Répertoire

6.1.7.4.1 Vue d'ensemble du répertoire

Afin de communiquer mutuellement, les AEI font usage de fonctions de répertoire qui leur donnaient les informations d'adressage dont elles ont besoin. Ces fonctions de répertoire sont nécessaires uniquement lors de l'ouverture de l'association ou durant une émission en mode sans association.

6.1.7.4.2 Modèle d'adressage

La couche liaison de données offre des services de transfert de données en mode non connecté et un espace d'adressage permettant la localisation des entités d'application. Cet espace d'adressage permet de localiser les entités individuellement et/ou par groupes et d'autre part de les localiser à l'aide d'adresses physiques ou logiques. Ces deux types d'adresses permettent respectivement de tenir compte ou de ne pas tenir compte de la topologie du réseau dans l'emplacement des entités d'application.

La couche application a une double exigence de localisation, celle relative aux AE lors de négociation des associations ou lors de l'émission de données sans association, et celle relative aux AEI lors de l'émission de données dans le cadre d'une association.

Par conséquent, les adresses des AE et les points d'accès des AEI qui nécessitent d'être localisés sont associés aux adresses de liaisons à partir de l'espace d'adressage de liaisons simples.

Dans le cas d'échanges point à point, les adresses d'AE et les points d'accès d'AEI sont désignés à l'aide d'adresses de liaisons individuelles. Dans le cas d'échanges multipoint, l'adresse de l'AE et le point d'accès de l'AEI localisant l'émetteur sont désignés par des adresses de liaisons individuelles alors que l'adresse de l'AE et le point d'accès de l'AEI localisant les récepteurs sont désignés par des adresses de liaisons de groupe.

Les règles d'attribution d'adresses de liaisons utilisées pour les adresses d'AE et les points d'accès d'AEI ne sont pas définies dans la présente norme.

Les abréviations utilisées pour désigner ces deux utilisations d'adresses de liaison sont: ADAE pour celles qui sont associées aux adresses d'AE et ADAEI pour celles qui sont associées aux points d'accès d'AEI.

NOTE Les ADAEI sont attribuées par les fonctions de répertoire d'un appareil aux AEI, dans le sous-ensemble qui a été attribué à une AE.

Les valeurs d'adresses contenues dans ce sous-ensemble peuvent être par exemple choisies en divisant l'espace d'adressage en fonction de l'architecture du réseau ou bien choisies en fonction d'autres critères satisfaisant au mieux aux exigences de communication de l'application utilisateur.

Règles d'adressage:

Rule_1: Chaque AE est identifiée par un et un seul "titre d'AE" qui est dans une relation bi-univoque avec une ADAE, lui permettant d'être localisée dans l'environnement AL.

Rule_2: Chaque AE peut être désignée par un ou plusieurs synonymes, chacun étant en relation avec une et une seule adresse.

Rule_3: Chaque AE peut être désignée dans le cadre d'une ou plusieurs "désignations de groupes d'AE", chacune étant en relation avec une et une seule adresse.

NOTE Il est important de noter que la biunivocité n'est pas une propriété des règles 2 et 3.

Rule_3: Chaque AEI qui est identifiée par un paramètre "AEI identification" (identification d'AEI) dans le champ d'une AE, une API, d'un AP donné, est en relation biunivoque avec une ADAEI, lui permettant d'être localisée dans l'environnement AL.

6.1.7.4.3 Fonctions de répertoire

La projection entre les noms permettant d'identifier les différents objets (AE et AEI) dans l'application, ainsi que les adresses de liaison (respectivement ADAE et ADAEI) utilisées pour les localiser dans l'environnement AL, est gérée par le service répertoire.

Pour toute initiative de communication au niveau d'une AEI, celle-ci appelle les fonctions de répertoire qui lui donnent les adresses nécessaires aux échanges.

- Fonction répertoire de l'initiateur (IDF, Initiator Directory Function):

Cette fonction est mise en œuvre pour une AEI initialisant un échange concernant une ouverture d'association ou une émission de données en mode non associé.

La fonction IDF est utilisée pour:

– obtenir une paire d'ADAE permettant la localisation des AE locales et distantes en fonction de leurs titres d'AE,

– allouer une ADAEI pour le point d'accès d'AEI locale et éventuellement une autre pour le point d'accès d'AEI distante. Cette allocation n'a lieu que lors de la création d'une association.

<Paramètres d'entrée> ::=

<titre d'AE locale>
<titre d'AE distante
| Synonyme du titre d'AE
| Désignation du groupe d'AE>
<Mode d'émission>

avec <Mode d'émission>::=<ASSOCIATED ï NON-ASSSOCIATED>;

<Paramètre de sortie> ::=

<ADAE locale>
<ADAE distante>
[<ADAEI locale>] (*)
[<ADAE≥ distante] (**)

(*) Cette ADAEI est retournée seulement dans le cas d'un mode d'émission ASSOCIATED".

(**) Cette ADAEI est retournée ou pas selon le titre d'AE distante.

Fonction répertoire du répondeur (RDF, Responder Directory Function):

Cette fonction est mise en œuvre au niveau d'une AE appelée, pour l'ouverture d'une association.

RDF:

Utilisée pour allouer une ADAE au niveau de l'entité appelée lors de la phase de réponse à une demande d'ouverture d'association.

<paramètres d'entrée> ::=

<ADAE locale>

<ADAE distante>

<Paramètres de sortie> ::=

<ADAEI locale>

NOTE Cette fonction n'aurait pas été utilisée au niveau d'un répondeur si l'initiateur de la communication avait déjà effectué l'allocation par utilisation de la fonction.

6.1.8 Coordination des AEI

6.1.8.1 Description des communications d'AEI

L'initiative et le type de communications appartiennent aux API. Les quatre types de communication sont:

– Échanges d'informations dans le mode non associé.
– Demandes d'ouverture d'association.
– Échanges d'informations dans le mode associé.
– Demandes de terminaison d'association

NOTE Pour chacun de ces types de communication, les objets de l'environnement AL sont mis en œuvre de manière spécifique qui est décrite dans les paragraphes suivants.

6.1.8.2 Échanges d'informations dans le mode non associé

Lorsqu'une API désire initialiser un transfert de données en mode non associé, elle procède de la manière suivante:

Scénario d'échange:

• elle attribue des adresses, le cas échéant:

– soit à l'AE source identifiée par son <AE title>, indiquant le destinataire (par son <AE title> ou par un <AE synonym title> ou par un <AE group designation> et le <Application context>,
– soit à une AEI dont le mode est NON-ASSOCIATED, si ce dernier a été créé pour permettre une série d'échanges.

• La procédure locale qui a lieu au niveau de l'AE après cette demande par l'API est la suivante:

Si l'API adresse l'AE, la procédure locale suivante est exécutée:

– Création d'un objet AEI dans le mode NON-ASSOCIATED,
– Entrée des attributs de cet objet:

- <Application context> prend la valeur indiquée par la demande,
- <Remote AE address> prend la valeur retournée par IDF pour le mode de communication non associé,
- <Local AE address>: prend la valeur retournée par ID pour le mode de communication non associé.
- Alors quel que soit l'adressage qui a eu lieu, un service de transfert de données en mode non associé MCSE a lieu dans le cadre de cette AEI. Les données utilisateur encapsulées dans ce transfert de données sont celles du service application utilisateur MCS qui a été demandé.
- L'AEI ainsi utilisée pourrait être maintenue ou annulée après cet échange. En cas d'annulation, cela sera immédiatement effectif si le service n'est pas confirmé ou après réception de la réponse et le passage à l'API des données qu'elle contient, si le service est confirmé.

• La procédure suivante est exécutée au niveau de l'AE destinataire:

- Selon qu'une AEI n'existe pas encore ou existe déjà, les opérations suivantes sont exécutées ou pas:
- Création d'un objet AEI dans le mode NON-ASSOCIATED.

• Remplissage des attributs de cet objet:

- <Application context>: prend la valeur indiquée dans la PDU reçue, si elle est prise en charge par l'AE.
- <Remote AE address> prend la valeur contenue dans la PDU reçue.
- <Local AE address> prend la valeur contenue dans la PDU reçue.
- Identification d'un objet API désigné implicitement ou explicitement dans la PDU.
- Passage du <Application service> contenu dans la PDU reçue à l'API identifiée pour l'exécution.
- L'AEI ainsi utilisée peut être annulée ou conservée à la fin de cet échange. En cas d'annulation, cela sera immédiatement effectif si le service n'est pas confirmé ou après émission de la réponse s'il est confirmé.

6.1.8.3 Demande d'ouverture d'association

Si une API désire initialiser une demande d'ouverture d'association, elle procède de la manière suivante:

Scénario d'ouverture:

• L'API adresse localement l'AE appelante identifiée par son <AE title>, indiquant:

- l'<AE title> appelé,
- le <opening service>, c'est-à-dire service d'ouverture, (Initiate Rq en MMS) qui contient les <Application reductions> c'est-à-dire réductions d'application, de l'association qui sera négociée,
- la description du <Application context> c'est-à-dire contexte d'application, autre que les réductions d'applications qui seront négociées au niveau du MCS.

La procédure locale qui sera exécutée au niveau de l'AE après cette demande d'API, est la suivante:

- Création d'un objet AEI dans le mode ASSOCIATED dans l'état OPENING.
- Entrée des attributs de cet objet:
- <Remote AE address; local AEI access point>: prennent la valeur retournée par IDF.
- Considération du contexte d'application demandé par l'utilisateur, et réduction éventuelle de cela, si son AE locale ne peut pas l'offrir.
- Service A_Associate.Rq appelé par MCSE dans le cadre de cette AEI avec le contexte d'application donné par l'utilisateur, ainsi que la PDU utilisateur de la demande d'ouverture.
- À la réception de la réponse, les autres attributs de l'AEI peuvent être remplis:
- <Remote AEI access point>: prend la valeur contenue dans la PDU.
- <Application context> prend la valeur contenue dans la réponse qui peut correspondre à une réduction de celle qui est demandée.

– Passage de l'état de l'AEI à ESTABLISHED,

• La procédure qui a lieu au niveau de l'AE qui répond à la réception d'une demande d'ouverture émise par la procédure source est la suivante:

- Création d'un objet AEI dans le mode ASSOCIATED dans l'état OPENING.
- Remplissage des attributs de cet objet:
- <Application Context>: prend la valeur indiquée dans la PDU reçue, ou une réduction, si l'AE ne la prend pas en charge.
- <Local AE address> prend la valeur reçue dans la PDU.
- Le <Local AEI acces point>, c'est-à-dire point d'accès de l'AEI locale, peut prendre la valeur facultative contenue dans la PDU ou celle retournée par la fonction répertoire du répondeur (RDF, Responder Directory Fonction).
- <Remote AE address; Remote access point> prennent les valeurs reçues dans la PDU.
- Adressage de l'API de son identification implicite ou explicite dans la PDU.
- Passage à l'utilisateur de la PDU de la demande d'ouverture.
- Lorsque l'API donne sa réponse, un appel du service A_ASSOCIATE.Rp de MCSE dans le cadre de cette AEI, avec l'utilisateur de PDU, en tant qu'une réponse à la demande d'ouverture et une mise à jour éventuelle du contexte s'il a été réduit par l'API.

– Passage de l'état de l'AEI à ESTABLISHED.

6.1.8.4 Échanges d'informations dans le mode associé

Lorsqu'une API désire initialiser un échange en mode associé, elle procède comme suit.

Scénario d'échange:

• Elle adresse l'AEI localement dans l'état ESTABLISHED, identifiée par son paramètre <AEI identifier> relatif à sa zone en indiquant:

– l'<application service> qu'il convient d'exécuter avec l'entité appelée.

• La procédure locale qu'il convient d'effectuer au niveau de l'AEI est comme suit:

- Lors de la réception de la réponse, pour un service confirmé, envoi de cette réponse à l'utilisateur.

– Appel du service de transfert de données en mode associé MCSE dans le cadre de l'AEI, avec comme données utilisateur, la PDU associée au service.

• La procédure exécutée au niveau de l'AEI qui répond à la réception de la demande de service est comme suit:

– Passage de l'<Application service> contenu dans la PDU de l'API.

– Appel de service et transfert d'informations en mode associé MCSE, dans le cadre de l'AEI, avec comme données utilisateur, la PDU de réponse associée à ce service.

6.1.8.5 Demande de terminaison d'une association

Lorsqu'une API désire initialiser une demande de terminaison d'une association, elle procède comme suit:

Scénario de terminaison:

• Elle adresse localement l'AEI identifiée par son <AEI identification> dans sa zone, indiquant:

– le <Closing application service>, c'est-à-dire Service d'application de fermeture.

• La procédure locale qui est exécutée au niveau de l'AEI initialisant cette fermeture est comme suit:

– Mettre l'AEI à l'état CLOSING (Blocage des transferts de données au-delà de ce qui a été engagé).

– Appel du service MCSE A-RELEASE pour émettre le service de demande de fermeture dans le cadre de cette AEI;

– À la réception de la réponse positive, annulation de l'objet AEI, retour à l'état ESTABLISHED dans les autres cas;

• La procédure distante qui est effectuée au niveau de l'AEI qui répond est comme suit:

– Mettre l'AEI à l'état CLOSING.

– (Blocage des transferts de données hors ceux engagés).

– Passage à l'utilisateur du service de demande de fermeture.

– à une réponse positive par l'utilisateur, annulation de l'objet AEI et envoi de la réponse à l'émetteur en utilisant le MCSE.

6.1.9 Concepts pour sub_MMS

6.1.9.1 Généralités

Les communications Sub-MMS peuvent être établies à travers une association négociée ou prédéfinie ainsi que sans association.

L'attribution de droits d'accès à l'association ainsi que les protections d'objets permettent et la protection d'objets autorise un accès sélectif aux objets.

6.1.9.2 Voies de communication Sub-MMS

1- Une communication de la Sub-MMS peut être réalisée sans association:

Ces communications peuvent avoir lieu en point à point (services confirmés/ non confirmés), en multipoint ou en distribution (services non confirmés).

Dans ce cas, il convient qu'une demande de services (autres que les services Initiate, Conclude et Abort) soit refusée si le service n'est pas pris en charge, si les ressources sont insuffisantes, s'il n'est pas en conformité avec le protocole, etc.

Les services Initiate, Conclude et Abort n'ont pas de signification.

Ce type de communication ne prend pas en charge le mécanisme de protection d'accès sur les objets.

2- Une communication de la Sub-MMS peut être réalisée dans une association prédéfinie:

Ces communications peuvent avoir lieu en point à point (services confirmés/ non confirmés), en multipoint ou en distribution (services non confirmés).

Dans ce cas, il convient qu'une demande de services (autres que les services Initiate, Conclude et Abort) soit refusée si le service n'est pas pris en charge, s'il n'est pas en conformité avec le protocole ou si les ressources réservées pour l'association sont insuffisantes, etc.

Les services Initiate, Conclude et Abort n'ont pas de signification.

Ce type de communication prend ou ne prend pas en charge le mécanisme de protection d'accès sur les objets.

3- Une communication de la Sub-MMS peut être réalisée dans une association négociée:

Ces communications peuvent avoir lieu en point à point (services confirmés/ non confirmés),

Dans ce cas, il convient qu'une demande de services (autres que les services Initiate, Conclude et Abort) soit refusée si le service n'est pas pris en charge, s'il n'est pas en conformité avec le protocole, ou si les ressources réservées pour l'association sont insuffisantes, etc.

Ce type de communication prend ou ne prend pas en charge le mécanisme de protection d'accès sur les objets.

6.1.9.3 Mécanisme de protection d'accès

6.1.9.3.1 Introduction du niveau de protection

L'accès à l'objet Sub-MMS peut être refusé à certains clients. Pour cela, nous définissons les niveaux de protection affectés à un objet et un droit d'accès affecté à l'association.

6.1.9.3.2 Niveau de protection affecté à un objet

La définition d'objets autres que l'objet VMD comprend un attribut FACULTATIF appelé "ACCESS PROTECTION" qui détermine si un objet est protégé ou non.

Un objet qui n'est pas protégé a son attribut "ACCESS PROTECTION" forcé à la valeur "FALSE".

Un objet qui est protégé a son attribut "ACCESS PROTECTION" forcé à la valeur "TRUE".

1 – Password,
2 – Access Groups,
3 – Access Rights.

Password:

L'attribut "Password" (mot de passe) permet de restreindre l'accès uniquement aux clients détenant le mot de passe.

Access Groups:

L'attribut "Access Groups" (groupes d'accès) permet de restreindre l'accès seulement aux clients qui appartiennent à l'un des groupes définis par l'attribut.

Access Rights:

L'attribut "Access Rights" (droits d'accès) indique les diverses possibilités pour obtenir l'accès à l'objet.

Il indique également pour chacune de ces possibilités d'accès, les services qui peuvent être exécutés ainsi que les conditions qui doivent être remplies (voir Tableau 2).

Tableau 2 – Protection d'accès

Classe d'objets	**Opérations exécutables**
Domain	Load (charger), Upload (charger/téléverser), Delete (supprimer), Connect (connecter) à une invocation de programme
Program invocation	Execute (exécuter), Initialize (initialiser), Stop (arrêter), Delete (supprimer)
Variable	Read (lire), Write (écrire)
Variable-List	Read (lire), Write (écrire), Delete (supprimer)
Événement	Read (lire), Alter (altérer), Acknowledge (aquitter)
Accès accordé à cause de mot de passe	**Conditions à remplir**
Oui	a) Le mot de passe de l'association est identique à celui de l'objet.
Non	b) La valeur du mot de passe de l'association est ZERO (zéro); condition (a) n'est pas remplie
Accès accordé à cause d'appartenance au groupe	**Conditions à remplir**
Oui	a) Le mot de groupe pour l'association et l'objet ont au moins un numéro de groupe identique
Non	b) La valeur du mot de groupe de l'association est ZERO; condition (c) n'est pas remplie
Accès accordé à tous	**Condition à remplir**
Oui	e) L'attribut "Access Rights" (droits d'accès) autorise l'accès à tous les clients.

6.1.9.3.3 Niveau des droits d'accès affectés à une association

Il est possible d'affecter des droits d'accès à des associations prédéfinies ou négociées. Dans ce cas, ces droits d'accès sont caractérisés par les paramètres "Password" et "Access Groups".

Les paramètres "Password" et "Access Group" sont accordés implicitement à l'association prédéfinie.

Les paramètres "Password" et "Access Group" sont accordés à l'association négociée par le service Initiate.

Les paramètres "Password" et "Access Group" caractérisent ainsi les droits d'accès du Client.

6.1.9.3.4 Mécanismes de contrôle d'accès pour des objets protégés

L'accès aux objets peut être accordé:

a) si l'attribut "Password" de l'objet est identique au paramètre "Password" de l'association

b) si l'attribut "Access Groups" de l'objet et le paramètre "Access Groups" de l'association ont au moins un numéro de groupe en commun.

c) ou à tous les clients si l'attribut "Access Rights" le permet.

NOTE Aucune restriction d'accès n'est imposée pour l'utilisation des services suivants: - Get Name List,
- Get Domain Attributes,
- Get Program Invocation Attributes,
- Get Variable Access Attributes,
- Get Variable List Attributes,
- Get Event Condition Attributes,
- Get OD Header/Data type Attributes.

6.1.9.4 Gestion de répertoires

6.1.9.4.1 Généralités

Tous les descripteurs d'objets sub-MMS pris en charge par une application forment un répertoire.

La disposition des descripteurs dans un répertoire peut être soit normalisée, soit libre.

La normalisation du répertoire est effectuée par un objet appelé «OD» (Object Dictionary, c'est-à-dire dictionnaire d'objets).

6.1.9.4.2 Utilisation de l'objet "OD"

Il n'est pas obligatoire qu'un appareil prenne en charge l'objet "OD".

6.2 ASE

6.2.1 MPS ASE (Services de fabrication périodiques/non périodiques)

6.2.1.1 Vue d'ensemble

Dans un environnement MPS, un certain nombre d'objets d'application est mis en œuvre. Ces objets proviennent directement de l'instanciation des différentes classes décrites précédemment.

Il convient que tous ces objets d'application représentant la configuration du réseau soient décrits partiellement ou globalement afin de permettre:

- à l'utilisateur d'avoir accès à la configuration d'application (par exemple: Accès au type de variable),
- la vérification de la cohérence de configuration de l'application entre plusieurs abonnés (par exemple: cohérence de configuration entre le producteur et les consommateurs d'une variable).

En outre, cette description est spécifiée dans la présente norme afin d'assurer l'interopérabilité dans les échanges de configuration entre les différentes mises en œuvre.

6.2.1.2 Concepts d'objets

6.2.1.2.1 Objets d'application

La description des services MPS périodiques/non périodiques conduit à décrire toutes les ressources impliquées dans la couche application. Cette spécification définit la partie statique de la spécification.

Les aspects réactifs des spécifications décrivent les différentes actions d'éléments de services sur les ressources de la couche application:

Les objets d'application de l'environnement MPS sont divisés en deux ensembles:

• objets liés aux variables,

• objets liés aux listes de variables.

Les variables sont des éléments abstraits qui associent une syntaxe de transfert et une sémantique à des données.

Les variables et les informations connexes sont fournies à l'utilisateur par le système de communication, soit individuellement soit par des listes.

D'autres variables sont fournies avec une caractéristique d'événement à laquelle l'utilisateur peut associer des actions relatives à l'exécution de l'application distribuée.

6.2.1.2.2 Adressage des objets d'application

Les attributs-clés (A_name) utilisés pour attribuer des adresses aux objets de la couche application MPS appartiennent tous au même espace d'adressage.

Chacun de ces noms dans l'espace d'adressage est défini dans la limite d'une longueur maximale de 16 caractères.

Les caractères appartiennent à l'ensemble défini pour le type 'visible string' de la notation de syntaxe abstraite ASN1 (ISO 8824).

À l'intérieur de cet ensemble, les caractères suivants doivent être utilisés:

Lettres majuscules (A..Z)

Lettres minuscules (a..z)

Chiffres (0..9)

Trait de soulignement(_)

Symbole monétaire ($)

Le caractère 'espace' ne doit pas être utilisé.

Les lettres majuscules et minuscules sont considérées comme des caractères différents.

Il convient qu'un A_Name ne commence pas par un chiffre.

Le caractère de soulignement comme deuxième caractère définit un nom réservé à la norme.

6.2.1.3 Sous-ASE d'accès de variable

6.2.1.3.1 Concepts

6.2.1.3.1.1 Variables de l'environnement MPS

Une variable de l'environnement MPS est une abstraction dans l'environnement MPS, dans lequel elle est référencée par un nom ou une description unique. Les données associées à cette variable correspondent à la représentation traitée par les services de liaison de données, qui est échangée sur le réseau.

6.2.1.3.1.2 Mémoire réseau

Les appareils du système de commande fournissent à un AP une valeur de chaque variable concernée dont une image est stockée dans la mémoire locale du réseau.

6.2.1.3.1.3 Concepts producteur/consommateur/tierce partie

Les entités d'application qui reconnaissent une variable sont munies d'une image locale de celle-ci.

Dans une application distribuée, une entité d'application unique est déclarée en tant que productrice de la valeur de la variable alors qu'une ou plusieurs entités d'application sont déclarées en tant que consommatrices de cette valeur.

En outre, certaines entités d'application, tout en n'étant ni productrices, ni consommatrices de la valeur de la variable, peuvent en avoir une connaissance, afin d'invoquer la mise à jour de sa valeur. Ces entités d'application sont appelées des entités tierces.

6.2.1.3.1.4 Concepts de périodicité et de non-périodicité

L'échange de la valeur de la variable entre l'entité productrice et les entités consommatrices peut être effectué périodiquement ou non périodiquement.

Une mise à jour périodique est initiée par le réseau et est définie pendant l'étape de configuration.

la période de mise à jour est déterminée par

• les exigences de l'utilisateur à remplir l'application distribuée,

• les contraintes liées au réseau.

La mise à jour non périodique est initiée par un utilisateur, comme l'entité productrice, une entité consommatrice ou une entité tierce.

6.2.1.3.1.5 Propriétés synchrones et synchrones

Les entités d'application traitent des variables élaborées selon la validité de production pour les variables produites, et selon la validité d'émission pour les variables consommées.

Définitions:

La validité de production et la validité d'émission associées à chaque variable ont un caractère asynchrone, car elles sont élaborées indépendamment de l'activité du réseau.

La validité de production et la validité d'émission associées à chaque variable ont un caractère synchrone lorsqu'elles comportent des ordres de synchronisation émis à partir du réseau.

Un ordre de synchronisation correspond à l'événement associé à l'émission ou à la réception d'une variable de la classe SYNCHRONIZATION.

6.2.1.3.2 Spécification de classe Variable

6.2.1.3.2.1 Modélisation de la classe Variable

Une variable dans l'environnement MPS est modélisée à l'aide des objets suivants (voir Figure 2):

- un objet *Produced Variable Local Image* qui inclut tous les attributs nécessaires à un utilisateur producteur.

- un ou plusieurs objets *Consumed Variable Local Image* qui incluent tous les attributs nécessaires à un ou plusieurs utilisateurs consommateurs.

Ces deux types d'objets sont issus de l'héritage de plusieurs classes:

Les objets *Produced Variable Local Image* sont instanciés à partir de la classe *Produced Variable*.

La classe *Produced Variable* comprend tous les attributs propres à une entité productrice et hérite de la classe *Identified Variable*. Cette dernière classe comprend tous les attributs qui sont communs à chaque image locale associée à cette variable.

En outre, la classe *Produced Variable* hérite facultativement de deux classes *Refreshment* et *Punctual Refreshment* qui incluent tous les attributs nécessaires à l'élaboration des informations relatives à la validité de la production associée à cette variable, au sein de l'entité d'application productrice.

Les objets Consumed Variable Local Image sont instanciés à partir de la classe Consumed Variable.

La classe *Consumed Variable* comprend tous les attributs variables propres à une entité consommatrice et hérite de la classe *Identified Variable.* Cette dernière classe comprend tous les attributs communs à chaque image locale associée à cette variable.

En outre, la classe *Consumed Variable* hérite facultativement de deux classes *Promptness* et *Punctual Promptness* qui incluent les attributs nécessaires à l'élaboration des informations relatives à la validité d'émission associée à la variable, au sein des entités d'application consommatrices.

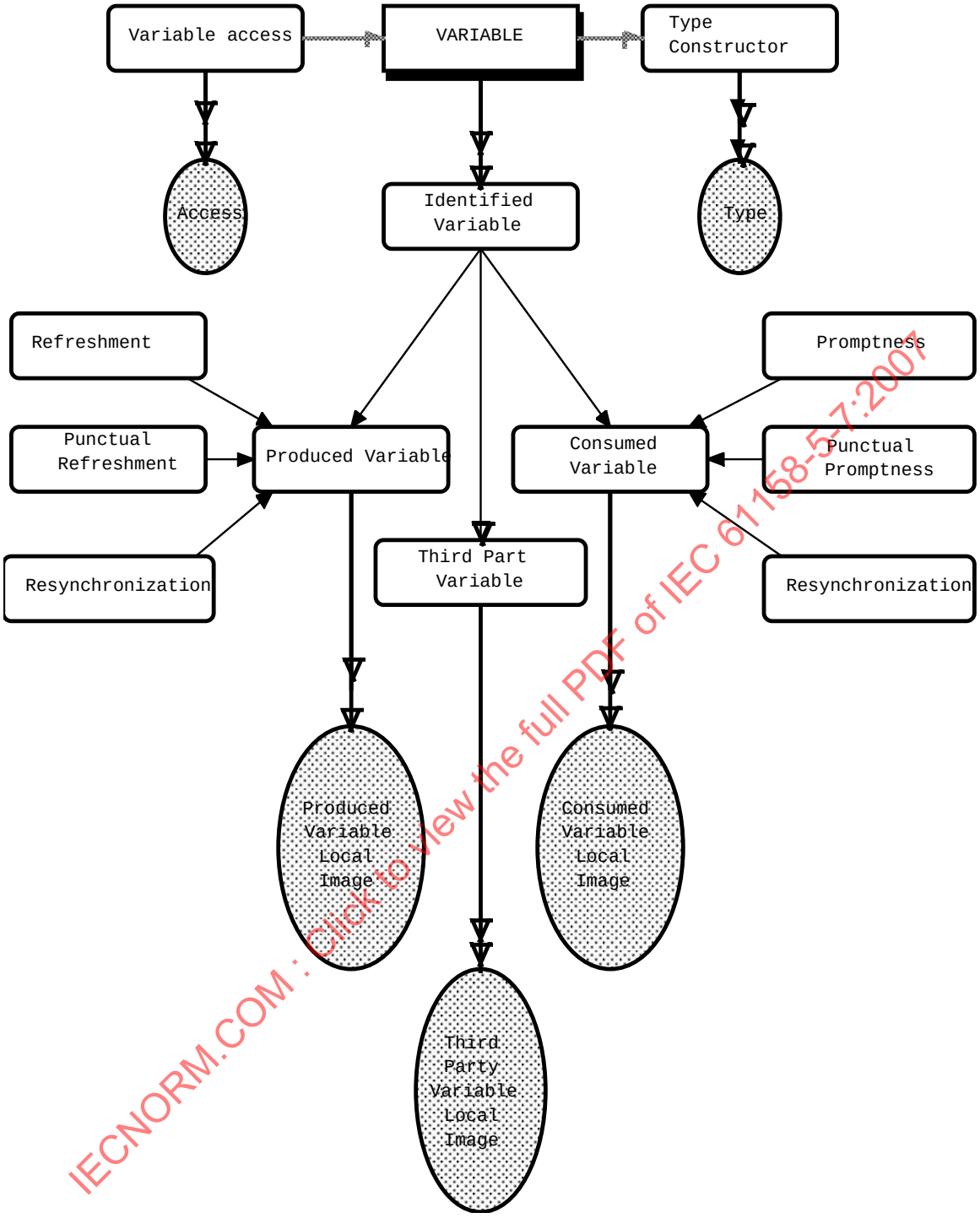
Chacune des classes *Produced Variable* et *Consumed Variable* hérite (en option) respectivement d'une classe *Production Resynchronization* et d'une classe *Consumption Resynchronization* qui incluent les attributs relatifs à la double mémorisation d'une variable afin de permettre la fourniture de la variable:

- au réseau par une entité productrice,
- aux entités consommatrices par le réseau, sur un ordre de synchronisation émis à partir du réseau.

Tous les attributs communs d'une variable sont décrits dans une classe *Variable* generic.

Le type de variable est décrit dans une classe *Type Constructor* qui est référencée par la variable et qui permet l'élaboration de tous les types qui peuvent exister dans l'environnement MPS.

En ce qui concerne l'accès à une variable, la classe *Variable Access* fait référence à une variable et indique le mode d'accès qui est utilisé.



Légende

Anglais	Français
VARIABLE	VARIABLE
Type Constructor	Type Constructeur
Variable access	Accès aux Variables
Identified Variable	Variable identifiée
Access	Accès
Type	Type
Produced Variable	Variable produite
Consumed Variable	Variable consommée

Refreshment	Rafraîchissement
Punctual Refreshment	Rafraîchissement ponctuel
Resynchronization	Resynchronisation
Punctual Promptness	Promptitude ponctuelle
Promptness	Promptitude
Produced Variable Local Image	Image locale de variable produite
Third Part Variable	Variable de tierce partie
Consumed Variable Local Image	Image locale de variable consommée
Third Party Variable Local Image	Image locale de variable de tierce partie

Figure 2 – Modèle d'objet du MPS ASE

6.2.1.3.2.2 Spécification de la classe *Variable* generic

6.2.1.3.2.2.1 Modèle de la classe Variable generic

FAL ASE: **MPS ASE**
CLASS: Variable
CLASS ID: non utilisé
PARENT CLASS: non utilisée
Generic Class: Variable
Class Attributes:

Key-Attribute	:	A_Name
Attribute	:	Reference Type Constructor
Attribute	:	Transmitted status [TRUE, FALSE]
Constraint	:	Transmitted status = TRUE
Attribute	:	Significant Status
Attribute	:	Identifier
Attribute	:	Transmission mode [PERIODIC,APERIODIC]]
Constraint	:	Transmission mode = PERIODIC
Attribute	:	Network period
Attribute	:	Class [NORMAL,SYNCHRONIZATION,DESCRIPTION]
Constraint	:	Class = SYNCHRONIZATION
Attribute	:	Consistency variable [TRUE, FALSE]
Constraint	:	Consistency variable = TRUE
Attribute	:	*Reference* Variable List

Instance Attributes:

Attribute	:	Public value
Constraint	:	Transmitted status = TRUE
Attribute	:	Transmitted status value
Attribute	:	Universal services requested [TRUE, FALSE]
Constraint	:	Universal services requested = TRUE
Attribute	:	Universal services scope [LOCAL,DISTANT]
Constraint	:	Universal services scope = DISTANT
Attribute	:	Priority [NORMAL,URGENT]

6.2.1.3.2.2.2 Attributs

A_Name:

Cet attribut identifie particulièrement une variable au niveau de l'interface entre la couche application et l'utilisateur. Le A_Name d'une variable appartient à l'espace d'adressage des services MPS.

En outre, pour une variable dont l'attribut "Class" a la valeur NORMAL ou SYNCHRONIZATION, la longueur du A_Name est limitée à 14 caractères.

***Reference* Type Constructor:**

Cette référence indique le A_name d'un type global associé à la variable.

Le type est décrit par un objet *Type Constructor* qui prend en charge tout type de variable qui peut exister dans l'environnement MPS.

Transmitted status:

Lorsque cet attribut boolean a la valeur TRUE, les statuts relatifs à la validité de la production de variable sont diffusés aux consommateurs respectifs.

Significant status:

Pour les variables ayant des informations relatives à leur validité de production, il est nécessaire de spécifier les statuts qui sont effectivement élaborés par le producteur.

Cet attribut est de type Boolean array (matrice de booléens), chaque élément TRUE indique que le statut de validité de production de la variable associée est significatif.

Le statut associé de la validité de production est défini pour l'élément de matrice *Transmitted status Value* avec le même indice

Les divers éléments sont

S1: Élaboration par le producteur d'un statut de rafraîchissement synchrone ou asynchrone.

S2: Elaboration par le producteur d'un statut de rafraîchissement ponctuel.

Identifier:

Cet attribut correspond à une adresse liaison de données qui se réfère à un objet de couche liaison de données.

Il existe une correspondance univoque (1:1) entre une variable A_name et l'identificateur associé. En outre, la relation entre un identifiant et un A_Name de variable est une question locale commise lors de l'étape de génération de la configuration réseau.

Transmission mode:

Cet attribut définit les échanges de la valeur de la variable à laquelle elle est associée.

Les échanges peuvent être PERIODIC ou APERIODIC, en fonction des exigences de l'utilisateur, et sont définis pendant l'étape de configuration du réseau.

Network period:

Quand le mode d'émission est PERIODIC, cet attribut spécifie la période de mise à jour de la variable par le réseau.

Cette période est exprimée en millisecondes.

Class:

Cet attribut permet d'associer une fonctionnalité à une variable. Il peut avoir l'une des valeurs suivantes:

NORMAL: Les variables de la classe NORMAL indiquent les informations de représentation des processus.

SYNCHRONIZATION: La variable de la classe SYNCHRONIZATION indique:

- Les informations de synchronisation de l'application distribuée,
- Les informations nécessaires au mécanisme de cohérence spatiale pour une liste de variables.

DESCRIPTION: Les variables de la classe DESCRIPTION caractérisent la description de tout ou partie des attributs relatifs à une variable, à un accès à une variable, à un type de variable, ou à une liste de variables.

Consistency variable:

Lorsque cet attribut a la valeur TRUE, la variable décrite correspond à une variable de cohérence impliquée avec le mécanisme de cohérence spatiale d'une liste de variables.

Reference Variable list:

Cette référence à la classe générique *Variable List* indique, pour une variable de la classe CONSISTENCY, les variables dont la réception au sein d'une entité d'application fait partie de l'élaboration de la valeur de la variable de cohérence lorsqu'un mécanisme de cohérence spatiale est impliqué

Public value:

Cet attribut correspond à la valeur de la variable fournie aux consommateurs par le producteur.

Après instanciation, cet attribut comprend:

- la valeur publique fournie au réseau,
- la dernière valeur publique reçue à partir du réseau

en ce qui concerne le fait que l'instance correspond à la valeur d'un producteur ou à la valeur d'un consommateur.

À un instant donné, doivent être considérées la valeur disponible au niveau du consommateur, la valeur émise lors du dernier échange et la dernière valeur reçue par chaque consommateur.

La dernière valeur reçue par un consommateur n'est pas toujours identique à la dernière valeur émise en raison de défauts d'émission possibles.

Transmitted status value:

Cet attribut correspond à la matrice de statut de validité de production, par rapport à une variable, et émis avec la valeur de cette variable. Chaque statut au sein de la matrice est un Boolean.

Tous les statuts de validité de production associés à la valeur de la variable ne sont pas nécessairement significatifs.

Quant à la valeur publique de la variable, l'instanciation de cet attribut inclut:

- les valeurs des statuts fournies au réseau,
- les dernières valeurs des statuts reçues à partir du réseau

en ce qui concerne le fait que l'instance correspond à la valeur d'un producteur ou à la valeur d'un consommateur.

Le traitement au niveau d'un producteur et des consommateurs de l'ensemble constitué de la valeur de variable et les valeurs de son statut associé est atomique.

Éléments de matrice des statuts:

S1 Statut de rafraîchissement synchrone/asynchrone

S2 Statut de rafraîchissement ponctuel

NOTE La syntaxe abstraite définit la matrice de statuts.

Requested universal services:

Lorsque cet attribut boolean a la valeur TRUE, les services universels peuvent être invoqués pour la variable en considération.

Universal services scope:

Cet attribut est utilisé par les services étendus d'accès aux variables, et définit la catégorie de service (local ou distant) à utiliser pour les services read/write universels.

Lorsque cet attribut a la valeur LOCAL, les services universels qui sont utilisés correspondent aux services locaux; lorsqu'il a la valeur DISTANT, ils correspondent à des services distants.

Priority:

Cet attribut est utilisé par les services étendus d'accès aux variables, et définit le niveau de priorité utilisé pour demander la mise à jour de variable lorsque les services universels correspondent à des services distants.

Lorsque cet attribut a la valeur URGENT, le contrôle de flux garantit qu'une demande de mise à jour urgente sera traitée avant une demande normale initiée après elle.

6.2.1.3.2.3 Spécification de la classe *Identified Variable*

La classe *Identified Variable* provient de l'instanciation de la classe générique *Variable*.

6.2.1.3.2.4 Spécification de la classe *Produced Variable*

6.2.1.3.2.4.1 Modèle de la classe Produced variable

Cette classe regroupe tous les attributs relatifs à une variable au sein d'une couche application productrice.

FAL ASE: **MPS ASE**
CLASS: Produced variable
CLASS ID: non utilisé
PARENT CLASS: non utilisée
Class: PRODUCED VARIABLE

Attribute	: *Inherit from* Identified Variable
Attribute	: Refreshment elaborated [TRUE, FALSE]
Constraint	: Refreshment elaborated = TRUE
Attribute	: *Inherit from* Refreshment
Attribute	: Punctual refreshment elaborated [TRUE, FALSE]
Constraint	: Punctual refreshment elaborated = TRUE,
Attribute	: *Inherit from* Punctual Refreshment
Attribute	: Resynchronized Variable [TRUE,FALSE]
Constraint	: Resynchronized Variable = TRUE
Attribute	: *Inherit from* Production Resynchronization
Attribute	: Emission Indication[TRUE,FALSE]

6.2.1.3.2.4.2 Attributs

***Inherit from* Identified Variable:**

La classe *Produced Variable* hérite de la classe *Identified Variable* qui lui fournit les attributs communs à toutes les copies locales de la même variable au sein de l'environnement MPS.

Refreshment elaborated:

Le rafraîchissement est une information relative à la validité de production d'une variable.

Cette information, qui est facultativement élaborée, indique aux utilisateurs consommateurs la période de production de l'utilisateur producteur.

Le statut associé à cette information est élaboré dans la couche application productrice et fourni sur le réseau avec la valeur du statut associée à la valeur de la variable.

Lorsque cet attribut Boolean a la valeur TRUE, un statut de rafraîchissement est élaboré au sein de l'entité d'application productrice.

***Inherit from* Refreshment:**

La classe *Produced Variable* hérite de la classe *Refreshment*.

La classe *Refreshment* inclut tous les attributs relatifs à l'élaboration des informations de rafraîchissement associées à une variable, afin de qualifier sa production.

Punctual Refreshment elaborated:

Le rafraîchissement ponctuel est une information relative à la validité de production d'une variable.

Cette information, qui est facultativement élaborée, indique aux utilisateurs consommateurs le respect par l'utilisateur producteur d'une opération d'écriture dans un intervalle de temps associé à un ordre de synchronisation émis à partir du réseau.

Le statut associé à cette information est élaboré dans la couche application productrice et fourni sur le réseau avec la valeur du statut associée à la valeur de la variable.

Lorsque cet attribut Boolean a la valeur TRUE, un statut de rafraîchissement ponctuel est élaboré au sein de l'entité d'application productrice.

Inherit from Punctual Refreshment:

La classe Produced Variable hérite de la classe Punctual Refreshment.

La classe Punctual Refreshment inclut tous les attributs relatifs à l'élaboration des informations de rafraîchissement ponctuel associées à la variable, afin de qualifier sa production.

Resynchronized Variable:

Certains utilisateurs produisent les variables de façon asynchrone par rapport au réseau.

Ces variables peuvent être resynchronisés au sein de la couche application par le biais d'une double mémoire tampon.

Le tampon privé est celui fourni à l'utilisateur producteur asynchrone. Le tampon public est celui fourni au réseau.

La resynchronisation d'une telle variable consiste au transfert du tampon privé au tampon public lors de la réception d'un ordre de synchronisation à partir du réseau.

Cet attribut prend en charge l'option choisie pour la variable en question.

Lorsque cet attribut boolean a la valeur TRUE, la variable concernée est resynchronisée en production.

***Inherit from* Production Resynchronization:**

La classe Produced Variable hérite de la classe Production Resynchronization.

La classe Production Resynchronization comprend tous les attributs relatifs à une resynchronisation de variable.

Emission Indication:

Lorsqu'il a la valeur TRUE, cet attribut boolean indique qu'une indication d'émission (A_SENT.ind) doit être retournée à l'utilisateur une fois que la variable a été envoyée sur le réseau.

6.2.1.3.2.5 Spécification de la classe Produced variable local image

Un objet *Produced Variable Local Image* est issu de l'instanciation à partir de la classe *Produced Variable.*

6.2.1.3.2.6 Spécification de la classe Consumed variable

6.2.1.3.2.6.1 Modèle de la classe Consumed variable

Cette classe regroupe tous les attributs relatifs à une variable au sein d'une couche application consommatrice.

FAL ASE: **MPS ASE**
CLASS: Consumed variable
CLASS ID: non utilisé
PARENT CLASS: non utilisée
Class: CONSUMED VARIABLE
Attribute : *Inherit from* Identified Variable
Attribute : Promptness elaborated [TRUE, FALSE]
Constraint : Promptness elaborated = TRUE
Attribute: *Inherit from* Promptness
Attribute : Punctual Promptness elaborated [TRUE, FALSE]
Constraint : Punctual Promptness elaborated = TRUE
Attribute: *Inherit from* Punctual Promptness
Attribute : Resynchronized Variable [TRUE, FALSE]
Constraint : Resynchronized Variable = TRUE
Attribute: *Inherit from* Consumption resynchronization
Attribute : Reception Indication [TRUE, FALSE]

6.2.1.3.2.6.2 Attributs

***Inherit from* Identified Variable:**

La classe *Consumed Variable* hérite de la classe *Identified* qui lui fournit tous les attributs communs à chaque copie locale de la même variable identifiée au sein de l'environnement MPS.

Promptness elaborated:

La promptitude est une information relative à la validité d'émission d'une variable.

Cette information, qui est facultativement élaborée, indique à l'utilisateur consommateur la réception d'une variable respectant une période de consommation

Le statut associé à cette information est élaboré dans la couche application consommatrice et fourni à l'utilisateur consommateur.

Lorsque cet attribut boolean a la valeur TRUE, un statut de promptitude est élaboré au sein de l'entité d'application consommatrice.

***Inherit from* Promptness:**

La classe Consumed Variable hérite de la classe Promptness.

La classe Promptness inclut tous les attributs relatifs à l'élaboration d'un statut de promptitude associé à une variable consommée.

Punctual Promptness elaborated:

La promptitude ponctuelle est une information relative à la validité d'émission d'une variable.

Cette information, qui est facultativement élaborée, indique à un utilisateur consommateur la réception d'une variable dans un intervalle de temps associé à un ordre de synchronisation émis à partir du réseau.

Le statut associé à cette information est élaboré dans la couche application productrice et fourni à l'utilisateur consommateur.

Lorsque cet attribut boolean a la valeur TRUE, un statut de promptitude ponctuelle est élaboré au sein de l'entité d'application consommatrice.

***Inherit from* Punctual Promptness:**

La classe Consumed Variable hérite de la classe Punctual Promptness.

La classe Punctual Promptness inclut tous les attributs relatifs à l'élaboration au niveau consommateur d'un statut de promptitude ponctuelle associé à une variable.

Resynchronized Variable:

Certains utilisateurs consomment les variables de façon asynchrone par rapport au réseau.

Ces variables peuvent être resynchronisés au sein de la couche application par le biais d'une double mémoire tampon.

Le tampon privé est celui fourni à l'utilisateur consommateur asynchrone. Le tampon public est celui qui reçoit la valeur à partir du réseau.

La resynchronisation d'une telle variable consiste au transfert du tampon public au tampon privé sur ordre de synchronisation du réseau.

Cet attribut prend en charge l'option choisie pour la variable concernée par la resynchronisation de la consommation.

Lorsque cet attribut a la valeur TRUE, la variable associée est resynchronisée en consommation.

***Inherit from* Consumption Resynchronization:**

La classe Consumed Variable hérite de Consumption resynchronization.

La classe Consumption Resynchronization comprend tous les attributs relatifs à une resynchronisation de variable en consommation.

Reception Indication:

Lorsqu'il a la valeur TRUE, cet attribut boolean indique qu'une indication de réception (A_RECEIVED.ind) doit être retournée à l'utilisateur une fois que la variable a été reçue à partir du réseau.

6.2.1.3.2.7 Spécification de la classe Consumed variable local image

Un objet *Consumed Variable Local Image* est issu de l'instanciation à partir de la classe *Consumed Variable*.

6.2.1.3.2.8 Spécification de la classe Third party variable

6.2.1.3.2.8.1 Modèle de la classe Third party variable

FAL ASE: **MPS ASE**
CLASS: Third party variable
CLASS ID: non utilisé
PARENT CLASS: non utilisée
Class: THIRD PARTY VARIABLE
Attribute : Inherit from IdentifiedVariable

6.2.1.3.2.8.2 Attributs

Inherit from IdentifiedVariable

La classe *Third Party Variable* hérite de la classe *Identified Variable* qui lui fournit les attributs nécessaires à connaître la variable.

Par conséquent, cette variable est fournie uniquement avec la paire (A_name, Identifier) de la variable identifiée qu'elle en hérite.

6.2.1.3.2.9 Spécification de la classe d'objet Third party variable local image

Un objet Third Party Variable Local Image est issu de l'instanciation à partir de la classe Third Party Variable.

6.2.1.3.2.10 Spécification de la classe *Type Constructor*

6.2.1.3.2.10.1 Modèle de la classe Type constructor

FAL ASE: **MPS ASE**
CLASS: Type constructor
CLASS ID: non utilisé
PARENT CLASS: non utilisée
Class : TYPE CONSTRUCTOR
Key–*Attribute* : A_Name
Attribute : Construction
[SIMPLE, ARRAY, STRUCTURE,PREDEFINED, EXPLICIT]
Constraint : Construction = SIMPLE
Attribute : Primitive type
Attribute : SIZE
Constraint : Construction = ARRAY
Attribute : Compressed [TRUE,FALSE]
Attribute : Dimension

Attribute : *Reference* Type Constructor
Constraint : Construction = STRUCTURE
Attribute : List of
Attribute : Named field [TRUE,FALSE]
Constraint : Named field = TRUE
Attribute : field name
Attribute : *Reference* Type Constructor

6.2.1.3.2.10.2 Attributs

A-Name:

Cet attribut identifie de façon unique un objet au sein de la couche application. Un A_name appartient à l'espace d'adressage MPS. En outre, il convient qu'un A_Name de la classe Type Constructor n'ait pas une longueur supérieure à 14 caractères.

Un A_name commençant par les caractères K_ indique un type réservé par la norme.

Construction:

Le type spécifié peut être de construction SIMPLE, et est alors directement dérivé des types de primitives.

Le type spécifié peut être de construction ARRAY, et représente un ensemble ordonné d'éléments du même type; le dernier type peut être lui-même de construction SIMPLE, ARRAY ou STRUCTURE. Les éléments de type de construction ARRAY sont numérotés dans un ordre croissant à partir de zéro (0) pour le premier élément.

Le type spécifié peut être de construction STRUCTURE et représente un type complexe composé d'un ensemble ordonné d'un ou plusieurs composants. Ces composants peuvent avoir différents types de construction SIMPLE, ARRAY ou STRUCTURE.

Le type spécifié peut être de construction PREDEFINED, et est ensuite défini dans la norme.

Les types de construction PREDEFINED ne peuvent pas être utilisés pour composer des types de ARRAY ou STRUCTURE.

Le type spécifié peut être de construction EXPLICIT afin de caractériser une variable dont le type peut changer dynamiquement.

Les types de construction EXPLICIT ne peuvent pas être utilisés pour composer des types de construction ARRAY ou STRUCTURE .

Caractéristiques des types de construction SIMPLE
Primitive Type:

Types de primitives utilisés.

Size:

Cet attribut a une valeur et un objectif qui dépend de la valeur de l'attribut Primitive Type défini ci-dessus. L'attribut size (taille), qui est décrit pour chacun des types de primitives, pour les instances de la classe Type Constructor, décrit une valeur maximale pour la variable.

Caractéristiques des types de construction ARRAY
Compressed:

La syntaxe de transfert de la couche application permet de définir, pour certaines variables de type ARRAY, les règles de codage compressées.

L'encodage compressé est spécifié en utilisant un attribut booléen.

Lorsque cet attribut a la valeur TRUE, le codage de la valeur de la variable associée est compressé, chaque fois que cela est possible en fonction du type des éléments de la matrice; lorsque cet attribut a la valeur FALSE, le codage de la valeur de la variable associée n'est pas compressé.

Dimension:

Cet attribut définit le nombre d'éléments de type de construction ARRAY.

***Reference* Type Constructor:**

Un type de construction ARRAY est composé d'éléments d'un type donné de construction SIMPLE ou ARRAY ou STRUCTURE.

Cet attribut fait référence à l'objet "Type", instancié à partir de la classe *Type Constructor* associée aux éléments de matrice

Caractéristiques des types de construction STRUCTURE
Named field:

Lorsqu'un type est STRUCTURE, chaque composant formant le type correspond à la représentation sémantique de la variable associée. Afin de permettre un accès partiel à ces composants, ils peuvent être désignés par un nom. Lorsque cet attribut a la valeur TRUE, un nom est associé au composant correspondant; lorsqu'il a la valeur FALSE, aucun nom n'est associé au composant.

Field name:

Cet attribut est une chaîne de caractères qui identifie de manière unique un composant de variable structurée.

NOTE Le domaine d'application des noms de composants est local à une structure.

***Reference* Type Constructor:**

Un type de construction STRUCTURE est composé de composants de tout type; c'est-à-dire SIMPLE, ARRAY ou STRUCTURE.

Cet attribut indique que l'instance de la classe *Type Constructor* associée à chaque composant est de type STRUCTURE

Caractéristiques des types de construction PREDEFINED

Les types de construction PREDEFINED sont utilisés généralement pour les variables de description et se rapportent aux types qui sont réservés par la norme.

Les types de construction PREDEFINED ne peuvent pas être définis explicitement parce qu'ils ont besoin d'une extension grammaticale afin d'utiliser des types facultatifs, des types alternatifs ou des types de longueur variable.

NOTE Les extensions grammaticales qui sont impliquées sont spécifiées dans la norme ISO 8824.

Caractéristiques des types de construction EXPLICIT

Les types de construction EXPLICIT permettent aux variables associées d'être fournies avec une syntaxe de transfert explicite, afin d'effectuer la sémantique de type avec les données.

6.2.1.3.2.10.3 Types primitifs

Les types de primitives qui sont sélectionnés et leurs caractéristiques sont principalement spécifiés dans la norme ISO 8824.

NOTE Le seul mode d'accès disponible pour une variable de type de primitive est l'accès global.

- **BOOLEAN**

La définition de ce type de primitive est la même que la définition de type boolean dans la norme ISO 8824. Pour ce type, il convient d'omettre l'attribut *Size*.

• BIT STRING

La définition de ce type de primitive est la même que la définition de la chaîne de bits dans l'ISO 8824. Pour ce type, l'attribut *Size* définit le nombre de bits dans la chaîne de bits.

• INTEGER

La définition de ce type de primitive est la même que la définition de type integer dans la norme ISO 8824. Pour ce type, l'attribut *Size* définit le nombre de bits qui sont nécessaires pour avoir une représentation en complément à deux de toutes les valeurs possibles.

• UNSIGNED

La définition de ce type de primitive est la même que la définition de type integer dans l'ISO 8824 à l'exclusion des nombres négatifs. Pour ce type, l'attribut *Size* définit le nombre de bits qui sont nécessaires pour avoir une représentation en complément à deux de toutes les valeurs possibles.

• OCTET STRING

La définition de ce type de primitive est la même que la définition de type chaîne de bits dans la norme ISO 8824. Pour ce type, l'attribut *Size* définit le nombre d'octets dans la chaîne.

• VISIBLE STRING

La définition de ce type de primitive est la même que la définition de type chaîne visible dans la norme ISO 8824. Pour ce type, l'attribut *Size* définit le nombre de caractères dans la chaîne.

• GENERALISED TIME

Ce type de primitive est défini par une chaîne visible à quatorze caractères (YYYYMMDDHHMMSS); il est équivalent à une forme particulière du type de temps généralisé, dans la norme ISO 8824. Pour ce type, il convient d'omettre *l'attribut Size*.

• FLOATING POINT

Ce type de primitive définit la représentation de la valeur pour les nombres réels positifs et négatifs, y compris, zéro, moins l'infini et plus l'infini.

Les valeurs aux bornes (0, +• -•), ainsi que les différents formats de représentation sont définis dans la CEI 60559.

Pour ce type, l'attribut *Size*, de type boolean, indique si le format de représentation est de simple précision (FALSE) ou de double précision (TRUE).

• BINARY TIME

Ce type de primitive est défini comme étant le type non signé de la présente norme, dont la valeur correspond à un nombre de temps élémentaires.

Pour ce type, chaque valeur de l'attribut *Size* définit la valeur d'un temps élémentaire ainsi qu'un nombre de bits utilisés pour représenter toute valeur temporelle binaire possible (voir Tableau 3).

Tableau 3 – Codage binaire temporel

Taille	10 ms	0,1 ms	1 ms	10 ms
16 Bits	0	1	2	3
32 Bits	4	5	6	7
48 Bits	8	9	—	—

• BCD

Ce type est utilisé pour représenter un ensemble de valeurs correspondant à la représentation numérique de caractères.

Pour ce type, il convient d'omettre *l'attribut Size*.

6.2.1.3.3 Mécanismes de base

6.2.1.3.3.1 Spécification de la classe Time-out

6.2.1.3.3.1.1 Modèle de la classe Time-out

L'élaboration, en produisant des entités d'application, d'informations par rapport à la validité de production implique des ressources de temporisation. De même pour les entités d'application consommatrices.

La description de cette ressource peut être modélisée avec un objet "attribute", dont le but est de définir les aspects visibles de temporisation utilisés par les services de l'environnement MPS.

FAL ASE: **MPS ASE**
CLASS: Time-out
CLASS ID: non utilisé
PARENT CLASS: non utilisée
Class: TIME-OUT
Attribute : Preset
Attribute : Current value
Attribute : State [RUNNING, IDLE]

6.2.1.3.3.1.2 Attributs

Preset:

"Preset" correspond à la durée associée à la temporisation quand elle est armée, lorsque la temporisation est en cours d'exécution ou a expiré.

Current value:

Current value (valeur courante) correspond à la valeur instantanée de la temporisation.

Cette valeur est égale à

- la période restante avant l'expiration, lorsque la temporisation s'exécute
- "0" lorsque le délai d'attente a expiré.

State:

L'état de la temporisation indique si la temporisation est active ou pas.

Les deux états sont comme suit:

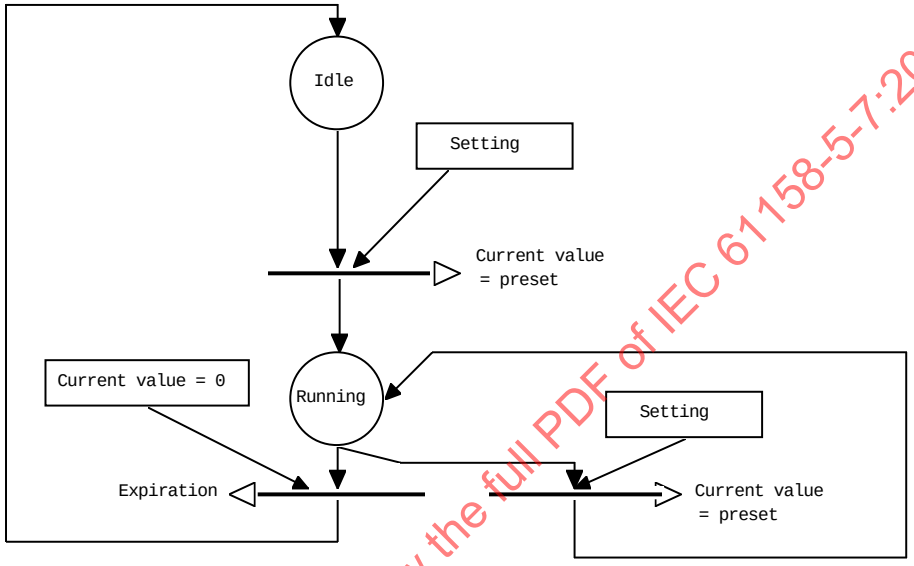
RUNNING:Correspond à une temporisation en cours d'exécution dans l'intervalle de temps entre une valeur prédéfinie et l'expiration.

IDLE: Correspond à une temporisation écoulée.

NOTE Dès que la valeur actuelle du délai d'attente est forcée à une valeur non nulle, le délai d'attente est automatiquement mis à l'état RUNNING.

6.2.1.3.3.1.3 Mécanisme de délai d'attente (time-out)

Le fonctionnement dynamique d'une temporisation est défini par le diagramme (réseau) d'évaluation représenté à la Figure 3.



Légende

Anglais	Français
Idle	Repos
Setting	Réglage
Running	En marche
Current value = 0	Valeur courante = 0
Expiration	Expiration
Current value = preset	Valeur courante = préétablie

Figure 3 – Diagramme d'évaluation de délai d'attente

6.2.1.3.3.2 Adressage de variable

6.2.1.3.3.2.1 Description de l'adressage de variable

L'accès de l'utilisateur à des variables d'application dans un environnement MPS peut être réalisé de diverses manières.

L'accès à une variable est réalisé à travers

- une variable A_Name, dans le cas d'un accès global de variable,
- un chemin d'accès dans le cas d'un accès partiel explicite à un champ nommé d'une variable structurée ou à un élément de matrice,
- un accès A_Name, dans le cas d'un accès partiel renommé à une variable ou à l'un de ses composants (champ de structure, élément de matrice...).

Les accès A_Name appartiennent tous au même espace d'adressage MPS fourni à l'utilisateur au niveau de l'interface avec la couche application.

6.2.1.3.3.2.2 Modes d'accès

6.2.1.3.3.2.2.1 Accès global

Ce mode d'accès permet à l'utilisateur d'accéder à une variable en tant qu'une variable globale.

L'accès à la variable est effectué avec le nom d'application de la variable qui est accessible. Dans ce cas, le nom d'application correspond à l'attribut *A_Name* de *Produced Variable Local Image,* ou *Consumed Variable Local Image,* ou *Third Party Variable Local Image*. Il appartient à l'espace d'adressage MPS fourni à l'utilisateur au niveau de l'interface entre la couche application et l'utilisateur.

6.2.1.3.3.2.2.2 Accès partiel explicite

Ce mode d'accès permet à l'utilisateur l'accès à un champ nommé d'une variable de type structuré ou à un élément d'une variable de type array.

L'accès partiel explicite est réalisé avec un chemin d'accès correspondant à l'énumération de tous les champs successifs dans l'arbre de structure ou à l'indice des éléments de matrice; Il convient que cette énumération commence à partir de la racine de la variable jusqu'au composant qui est accessible.

6.2.1.3.3.2.2.3 Accès renommé

Un accès renommé consiste en la projection d'un A_Name à partir de l'espace d'adressage MPS sur le A_Name d'une variable ou sur le chemin d'accès d'une variable de type structuré ou sur un élément de variable de type array.

Les noms qui sont utilisés dans l'accès partiel renommé appartiennent au même espace d'adressage que les noms de variables globales; ce qui signifie qu'ils appartiennent à l'espace d'adressage fourni à l'utilisateur au niveau de l'interface avec la couche application.

Par conséquent, l'accès renommé permet à l'utilisateur d'accéder à un champ de structure ou à un élément de matrice de la même manière que pour une variable globale.

NOTE Ce mode d'accès est fourni pour améliorer l'efficacité de l'émission, en regroupant plusieurs variables définies par l'utilisateur dans la même structure.

Cela signifie que l'amélioration de l'efficacité de l'émission peut conduire à grouper en un nombre minimal de DLPDU les variables échangées entre les AP dans une application distribuée élémentaire. Ceci est fait dans le but de minimiser les données de contrôle dans la couche liaison de données.

Les structures d'optimisation créées au niveau de la couche application sont transparentes pour l'utilisateur avec l'accès renommé, mais elles doivent être déclarées dans toutes les entités d'application qui doivent accéder à tout ou une partie des variables représentées définies par l'utilisateur.

NOTE pour l'élaboration de la structure d'optimisation: Seules les variables qui n'ont pas besoin de validité de production et pour lesquelles la validité d'émission est élaborée avec la même valeur de la période de consommation peuvent être regroupées dans la même structure d'optimisation.

6.2.1.3.3.2.3 Modélisation de l'accès renommé de variable

6.2.1.3.3.2.3.1 Spécification de la classe *Variable Access*

Class: VARIABLE ACCESS

Key-Attribute	: A_Name
Attribute	: Reference Variable
Attribute	: Access Mode [PARTIAL, GLOBAL]
Constraint	: Access Mode = PARTIAL
Attribute	: Access Path

A_Name:

Cet attribut permet une spécification unique d'un nom d'accès au niveau de l'interface entre la couche application et l'utilisateur. Le A_Name appartient à l'espace d'adressage des services MPS

En outre, il convient que le paramètre A_Name de la classe Variable Access ne soit pas plus long que 14 caractères.

***Reference* Variable:**

La classe *Variable Access* se réfère à la classe *Variable*.

La classe *Variable* indique la variable d'application avec laquelle l'accès A_Name est associé.

NOTE Un nom d'accès renommé d'une variable est global, ce qui signifie qu'il convient que le même nom d'accès au sein de plusieurs entités d'application différentes se réfère à la même variable de la couche application.

Access Mode:

Cet attribut indique quel mode d'accès doit être utilisé.

Lorsqu'il a la valeur TRUE, cet attribut indique un accès renommé global à une variable de la couche application.

Lorsqu'il a la valeur FALSE, il indique un accès partiel à un champ d'une variable structurée ou d'un élément de matrice.

Access Path:

Le chemin d'accès à un champ de variable structurée ou à un élément de matrice se réfère, par ordre croissant de profondeur, aux noms de champs de structure ou aux indices d'éléments de matrice.

NOTE Afin d'atteindre une spécification complète du chemin d'accès à un composant de variable, il est nécessaire que tous les attributs *Named Field* des objets *Type Constructor* à différents niveaux de la structure aient la valeur TRUE.

6.2.1.3.3.3 Spécification de la validité de l'émission de variables

6.2.1.3.3.3.1 Vue d'ensemble de l'émission de variables

Les utilisateurs consommateurs peuvent disposer d'informations relatives à la validité d'émission pour une variable.

Deux ensembles d'informations de validité d'émission sont définis:

- Promptitude; relative à la validité d'émission concernant le délai de validité d'un utilisateur pour la variable qui est reçue: ce délai de validité est appelé la période de consommation. La sémantique associée à cette information dépend de la nature synchrone ou asynchrone de la promptitude

- Promptitude ponctuelle; relative à la validité d'émission concernant un intervalle de temps de production associé à un ordre de synchronisation émis à partir du réseau.

6.2.1.3.3.3.2 Promptitude

6.2.1.3.3.3.2.1 Définitions de promptitude

Les attributs de promptitude associés à une variable sont facultatifs.

La promptitude est relative à la validité de disponibilité de la valeur d'une variable fournie à l'utilisateur par le réseau. Cette validité est élaborée avec une période de consommation maximale qui est fixée par l'utilisateur.

Dans le cas de la promptitude synchrone, la sémantique de cette information de validité prend en compte le respect de la disponibilité par le réseau d'un ordre de synchronisation associé à la variable.

Promptitude asynchrone:

Lorsqu'elle a la valeur TRUE, cette information booléenne indique dans une entité consommatrice que la valeur de la variable a été mise à jour par le réseau pendant un délai qui est inférieur à la période de consommation de la variable.

Promptitude synchrone:

Lorsqu'elle a la valeur TRUE, cette information booléenne indique dans une entité consommatrice que l'ordre de synchronisation a été reçu et a été suivi par (au moins) une mise à jour de la variable associée par le réseau et que la mise à jour se situait dans la période de consommation.

6.2.1.3.3.3.2.2 Spécification de la classe *Promptness*

Cette classe collecte tous les attributs relatifs à l'élaboration d'informations de promptitude associées à une variable au sein d'une entité d'application consommatrice.

Class:	PROMPTNESS	
	Attribute	: Consumption period
	Attribute	: *Inherit from* Time-out
	Attribute	: Promptness characteristic [SYNCHRONOUS, ASYNCHRONOUS]
	Constraint	: Promptness characteristic = SYNCHRONOUS
	Attribute	: *Reference* (Synchronization) Variable
	Attribute	: Promptness status [TRUE,FALSE]

Consumption period:

L'utilisateur consommateur déclare une période de consommation correspondant aux besoins de l'application distribuée.

Cette période correspond à la période minimale pour laquelle l'utilisateur consommateur accédera à la variable.

Inherit from Time-out:

La classe Promptness hérite de la classe Time-out.

La classe Time-out comprend tous les attributs relatifs à une temporisation. Cette ressource est obligatoire pour élaborer une information relative à une promptitude de variable.

Promptness characteristics:

La promptitude peut avoir une caractéristique qui est propre au consommateur, et qui influe sur la sémantique du statut de promptitude.

Lorsque la caractéristique est ASYNCHRONOUS, la promptitude est élaborée exclusivement avec des attributs propres à la variable.

Lorsque la caractéristique est SYNCHRONOUS, la promptitude prend en compte non seulement les attributs de variables, mais aussi un événement lié à la réception d'un ordre de synchronisation.

***Reference* (Synchronization) Variable:**

Cet attribut fait référence à une classe *Variable* pour laquelle l'attribut *Class* a la valeur SYNCHRONIZATION. Cela permet une référence à l'ordre de synchronisation qui définit la temporisation dans le cas d'une promptitude synchrone.

Promptness status:

La promptitude de variable est caractérisée par un statut.

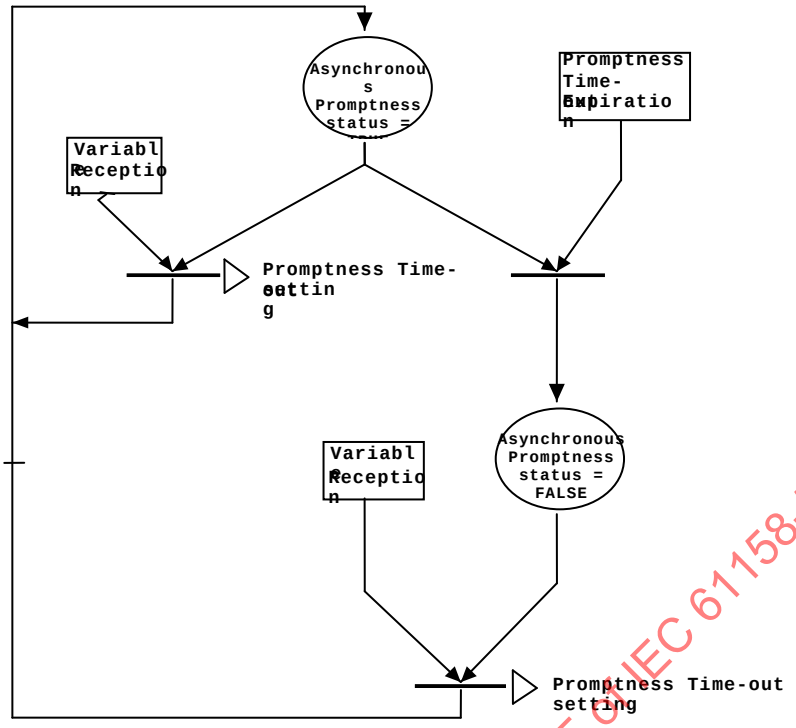
Ce statut est de type booléen et est localement élaboré au sein de chaque entité ayant une *Consumed Variable Local Image* (image locale de variable consommée) associée à la variable.

Les conditions de transition de statut de promptitude d'un état à un autre sont élaborées à travers

- des événements correspondants à la réception de variable, ou à l'occurrence de l'ordre de synchronisation associé à la variable,
- l'expiration du délai d'attente associé à la variable,
- l'état du délai d'attente associé à la variable pour le traitement de la promptitude synchrone.

6.2.1.3.3.3.2.3 Mécanisme

Le statut de promptitude asynchrone associé à une variable au sein d'une entité consommatrice est dynamiquement traité par le mécanisme décrit à la Figure 4 et dans le Tableau 4.



Légende

Anglais	Français
Variable Reception	Réception de variable
Promptness Timeout setting	Réglage de temporisation de promptitude
PromptnessTimeout Expiration	Expiration de délai de promptitude
Asynchronous Promptness status = TRUE	Statut de promptitude asynchrone = VRAI
Asynchronous Promptness status = FALSE	Statut de promptitude asynchrone = FAUX

Figure 4 – Diagramme d'évaluation de statut de promptitude asynchrone

Tableau 4 – Événements et actions de promptitude asynchrone

État initial: **Statut de promptitude asynchrone = FALSE**

État du délai d'attente de promptitude = IDLE

Demandes: **Réception de Variable:**

Variable disponible à partir du réseau

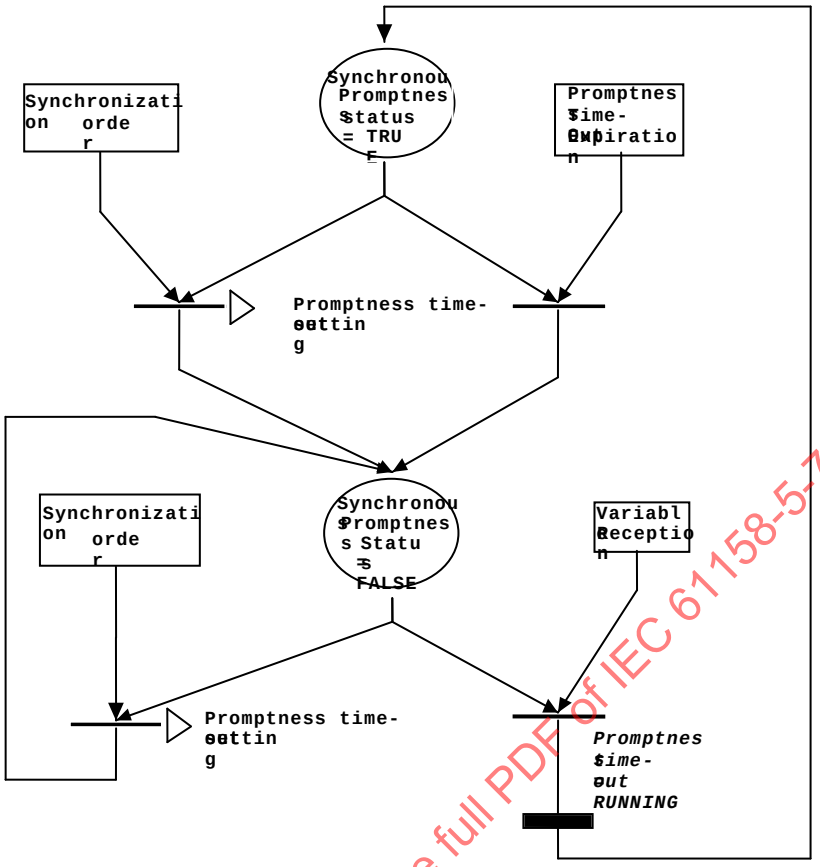
Expiration du délai d'attente de promptitude:

Transition de la temporisation associée au mécanisme élaborant le statut de promptitude asynchrone de l'état RUNNING à l'état IDLE.

Actions: **Paramétrage du délai d'attente de promptitude**

Transition de la temporisation associée au mécanisme élaborant le statut de promptitude asynchrone de l'état IDLE à l'état RUNNING avec un préréglage égal à la valeur de l'attribut *Consumption period*.

Le statut de promptitude synchrone associé à une variable au sein d'une entité d'application consommatrice est traité par le mécanisme décrit à la Figure 5 et dans le Tableau 5.



Légende

Anglais	Français
Promptness Timeout setting	Réglage de temporisation de promptitude
PromptnessTimeout Expiration	Expiration de délai de promptitude
Synchronous Promptness status = TRUE	Statut de promptitude synchrone = VRAI
Synchronous Promptness status = FALSE	Statut de promptitude synchrone = FAUX
Synchronisation order	Ordre de synchronisation
Promptness Timeout = RUNNING	Temporisation de promptitude =EN MARCHE

Figure 5 – Diagramme d'évaluation de statut de promptitude synchrone

Tableau 5 – Événements et actions de promptitude synchrone

ÉTAT INITIAL:	**Statut de promptitude synchrone = FALSE**
	État du délai d'attente de promptitude = IDLE
Demandes:	**Réception de variable:**
	Variable disponible à partir du réseau
	Ordre de synchronisation:
	Occurrence d'un ordre de synchronisation associé au mécanisme d'élaboration du statut de promptitude synchrone.
	Expiration du délai d'attente de promptitude:
	Transition de la temporisation associée au mécanisme élaborant le statut de promptitude synchrone de l'état RUNNING à l'état IDLE.
Actions:	**Paramétrage du délai d'attente de promptitude**
	Transition de la temporisation associée au mécanisme élaborant le statut de promptitude synchrone de l'état IDLE à l'état RUNNING avec un préréglage égal à la valeur de l'attribut *Consumption period*.

6.2.1.3.3.3 Promptitude ponctuelle

6.2.1.3.3.3.1 Définition de promptitude ponctuelle

L'information de promptitude ponctuelle associée à une variable est facultative.

La promptitude ponctuelle est relative à une validité de la disponibilité de la valeur de variable jusqu'à l'utilisateur consommateur à partir du réseau. Cette validité est élaborée par le biais d'un intervalle de temps de consommation qui est défini pendant l'étape de configuration. Cet intervalle de temps est activé à la réception d'un ordre de synchronisation à partir du réseau.

Promptitude ponctuelle:

Lorsqu'il a la valeur TRUE, ce booléen informe l'entité d'application consommatrice que la valeur de la variable a été mise à jour par le réseau (au moins) une fois dans l'intervalle de temps qui suit la réception d'un ordre de synchronisation associé à la variable. Il indique également que la valeur de variable n'a pas été mise à jour hors de l'intervalle de temps.

6.2.1.3.3.3.2 Spécification de la classe *Punctual Promptness*

Cette classe collecte tous les attributs relatifs à l'élaboration d'informations de promptitude ponctuelle associées à une variable au sein d'une entité d'application consommatrice.

Class: PUNCTUAL PROMPTNESS

Attribute	: Consumption time slot
Attribute	: *Inherit from* Time-out
Attribute	: *Reference* (Synchronization) Variable
Attribute	: Punctual Promptness Status [TRUE,FALSE]

Consumption time-slot:

La promptitude ponctuelle est élaborée par le biais d'un intervalle de temps de consommation, associé à une variable par un utilisateur consommateur.

La promptitude ponctuelle informe un consommateur de variable qu'il a reçu une nouvelle valeur (au moins) une fois durant l'intervalle de temps associé au dernier ordre de synchronisation qui a été reçu.

Inherit from Time-out:

La classe Punctual Promptness hérite de la classe Time-out.

La classe Time-out comprend tous les attributs relatifs à une temporisation (prédéfini, état).

***Reference* (Synchronization) Variable:**

Cet attribut fait référence à une classe *Consumed Variable* dont l'attribut *Class* a la valeur SYNCHRONIZATION, et qui définit l'ordre de synchronisation qui définit le délai d'attente dans le cas d'une promptitude ponctuelle.

Punctual promptness status:

La promptitude ponctuelle d'une variable est caractérisée par un statut.

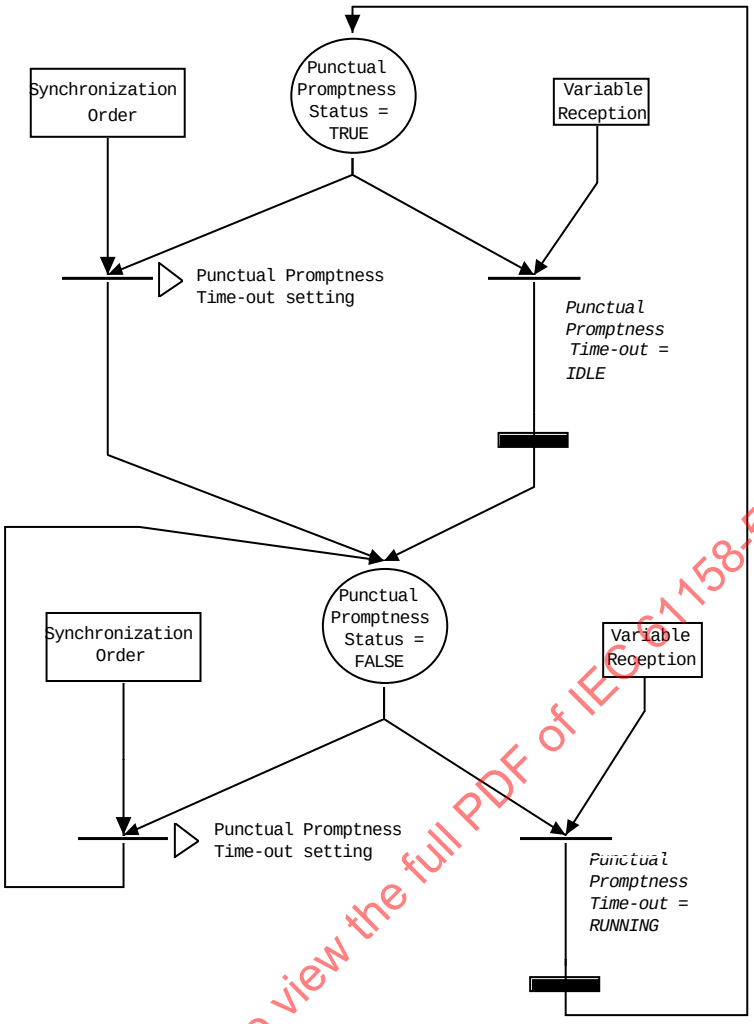
Ce statut est de type booléen et est localement élaboré au sein de chaque entité d'application où un objet *Consumed Variable Local Image* (image locale de variable consommée) est déclaré pour la variable.

Les conditions de transition de statut de promptitude d'un état à un autre sont élaborées à travers

- des événements correspondants à la réception de variable, ou à l'occurrence d'un ordre de synchronisation associé à cette variable,
- l'état du délai d'attente associé à la variable pour le traitement de la promptitude ponctuelle.

6.2.1.3.3.3.3 Mécanisme

Le statut de promptitude ponctuelle associé à une variable au sein d'une entité d'application consommatrice est traité par le mécanisme décrit à la Figure 6 et dans le Tableau 6.



Légende

Anglais	Français
Punctual Promptness Timeout setting	Réglage de temporisation de promptitude ponctuelle
Punctual Promptness status = TRUE	Statut de promptitude ponctuelle = VRAI
Punctual Promptness status = IDLE	Statut de promptitude ponctuelle = REPOS
Synchronisation order	Ordre/commande de synchronisation
Punctual Promptness Timeout =RUNNING	Temporisation de promptitude ponctuelle =EN MARCHE
Variable reception	Réception de variable
Punctual Promptness status = FALSE	Statut de promptitude ponctuelle = FAUX

Figure 6 – Diagramme d'évaluation de statut de promptitude ponctuelle

Tableau 6 – Événements et actions de promptitude ponctuelle

État initial:	**Statut de promptitude ponctuelle = FALSE**
	État du délai d'attente de promptitude = IDLE
Demandes:	**Réception de Variable:**
	Variable disponible à partir du réseau
	Ordre de synchronisation:
	Occurrence de l'ordre de synchronisation associé au mécanisme d'élaboration de la promptitude ponctuelle.
Actions:	**Paramétrage du délai d'attente de promptitude ponctuelle**
	Transition de la temporisation associée au mécanisme élaborant le statut de promptitude ponctuelle de l'état IDLE à l'état RUNNING avec un préréglage égal à la valeur de l'attribut *Consumption Time slot*.

6.2.1.3.3.4 Spécification de la validité de production de variable

6.2.1.3.3.4.1 Définition de la validité de production de variable

La validité de production peut être associée à n'importe quelle variable. Les informations de validité sont élaborées par une entité d'application productrice et sont fournies aux entités consommatrices. Ces entités consommatrices ne voient qu'une valeur échantillonnée de l'information.

Deux catégories d'informations de validité de production sont définies:

- Rafraîchissement, par rapport à une validité de production en ce qui concerne un délai maximal fixé par l'utilisateur. La sémantique de rafraîchissement dépend de sa caractéristique, synchrone ou asynchrone, à l'intérieur d'une entité d'application de production.
- Rafraîchissement ponctuel, par rapport à une validité de production en ce qui concerne un intervalle de temps de production associé à un ordre de synchronisation émis à partir du réseau.

6.2.1.3.3.4.2 Rafraîchissement

6.2.1.3.3.4.2.1 Définition du rafraîchissement

Cette information de validité de production est élaborée en option et est caractérisée par un statut.

Le rafraîchissement est relatif à la validité de la disponibilité du réseau par un utilisateur producteur d'une valeur de variable. Cette validité est élaborée par le biais d'une période de production qui est fixée par l'utilisateur.

Dans le cas d'un rafraîchissement synchrone, la sémantique de cette information de validité indique le respect de la disponibilité par le réseau de l'ordre de synchronisation associé à la variable.

Cette information relative au rafraîchissement, transportée avec la valeur de variable par le réseau, est fournie aux utilisateurs consommateurs.

Rafraîchissement asynchrone:

Lorsqu'il a la valeur TRUE, ce booléen indique dans une entité d'application productrice que la valeur de la variable a été fournie au réseau par l'utilisateur producteur, avec un délai inférieur à la période de production.

Rafraîchissement synchrone:

Lorsqu'il a la valeur TRUE, ce booléen indique au sein d'une entité d'application productrice qu'un ordre de synchronisation a été reçu et suivi par (au moins) un approvisionnement d'une nouvelle valeur de la variable du réseau à partir de l'utilisateur producteur avec un délai inférieur à la période de production.

6.2.1.3.3.4.2.2 Spécification de la classe *Refreshment*

Cette classe collecte tous les attributs relatifs à l'élaboration d'informations de rafraîchissement associées à une variable au sein d'une entité d'application productrice.

Class:	REFRESHMENT	
	Attribute	: Production period
	Attribute	: *Inherit from* Time-out
	Attribute	: Refreshment characteristic [SYNCHRONOUS,ASYNCHRONOUS]
	Constraint	: Refreshment characteristic = SYNCHRONOUS
	Attribute	: *Reference* (Synchronization) Variable
	Attribute	: Refreshment Status [TRUE, FALSE]

Production period:

L'utilisateur producteur peut déclarer une période de production correspondant aux besoins de son application distribuée.

Cette période correspond à l'intervalle maximal au cours duquel l'utilisateur producteur fournira la variable au réseau.

Inherit from Time-out:

La classe *Refreshment* hérite de la classe *Time-out*.

La classe *Time-out* comprend tous les attributs relatifs à une temporisation. Cette ressource est obligatoire à l'élaboration de l'information relative au rafraîchissement de la variable.

Refreshment characteristic:

Le rafraîchissement peut être fourni avec une caractéristique propre à un producteur et qui influe sur la sémantique du statut de rafraîchissement.

Lorsque la caractéristique est ASYNCHRONOUS, le rafraîchissement est élaboré exclusivement avec des attributs propres à la variable.

Lorsque la caractéristique est SYNCHRONOUS, le rafraîchissement prend en compte non seulement les attributs de variables, mais un événement lié à la réception d'un ordre de synchronisation.

***Reference* (Synchronization) Variable:**

L'attribut fait référence à une classe *Variable* dont l'attribut *Class* a la valeur SYNCHRONIZATION. Cela définit l'ordre de synchronisation qui règle la temporisation dans le cas d'un rafraîchissement synchrone.

Refreshment Status:

Le rafraîchissement de variable fourni à un utilisateur est caractérisé par un statut.

Ce statut est de type booléen et est localement élaboré au sein de chaque entité d'application où une *Produced Variable Local Image* (image locale de variable produite) est déclarée.

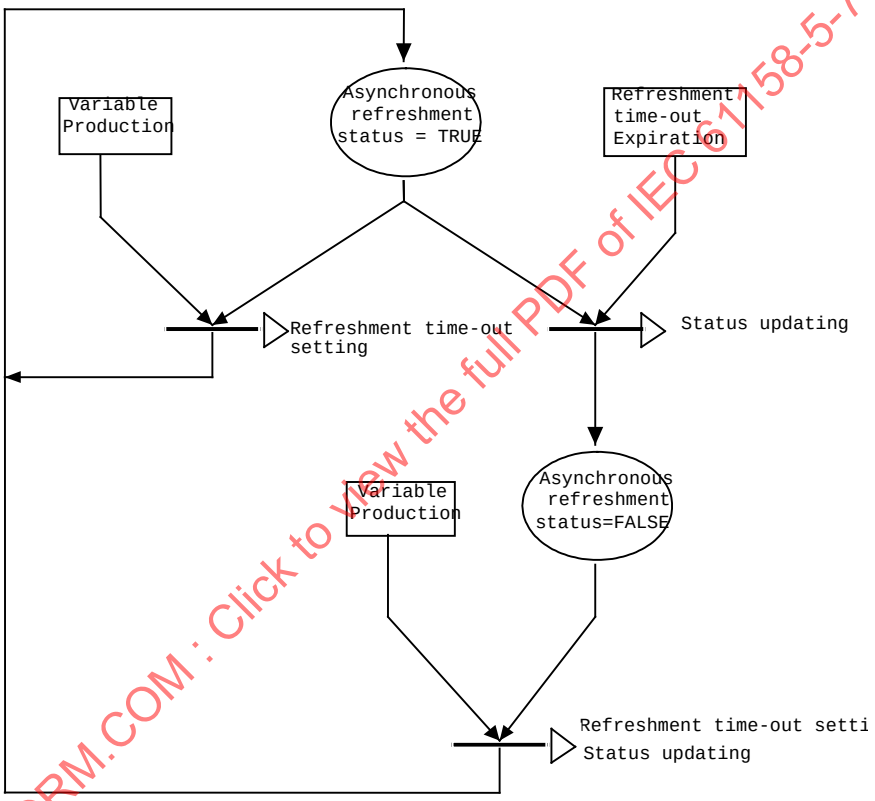
La valeur du statut est associée à la valeur de la variable pendant l'échange sur le bus.

Le statut de rafraîchissement est défini par deux états, et les conditions de transition d'un état à un autre sont élaborées à travers

- des événements correspondants à la production de la variable,
- des événements liés à l'occurrence de l'ordre de synchronisation associé à la variable,
- l'expiration du délai d'attente associé à la variable,
- l'état du délai d'attente associé à la variable pour le traitement du rafraîchissement synchrone.

6.2.1.3.3.4.2.3 Mécanisme

Le statut de rafraîchissement asynchrone associé à une variable au sein d'une entité d'application productrice est élaboré par le mécanisme décrit à la Figure 7 et dans le Tableau 7.



Légende

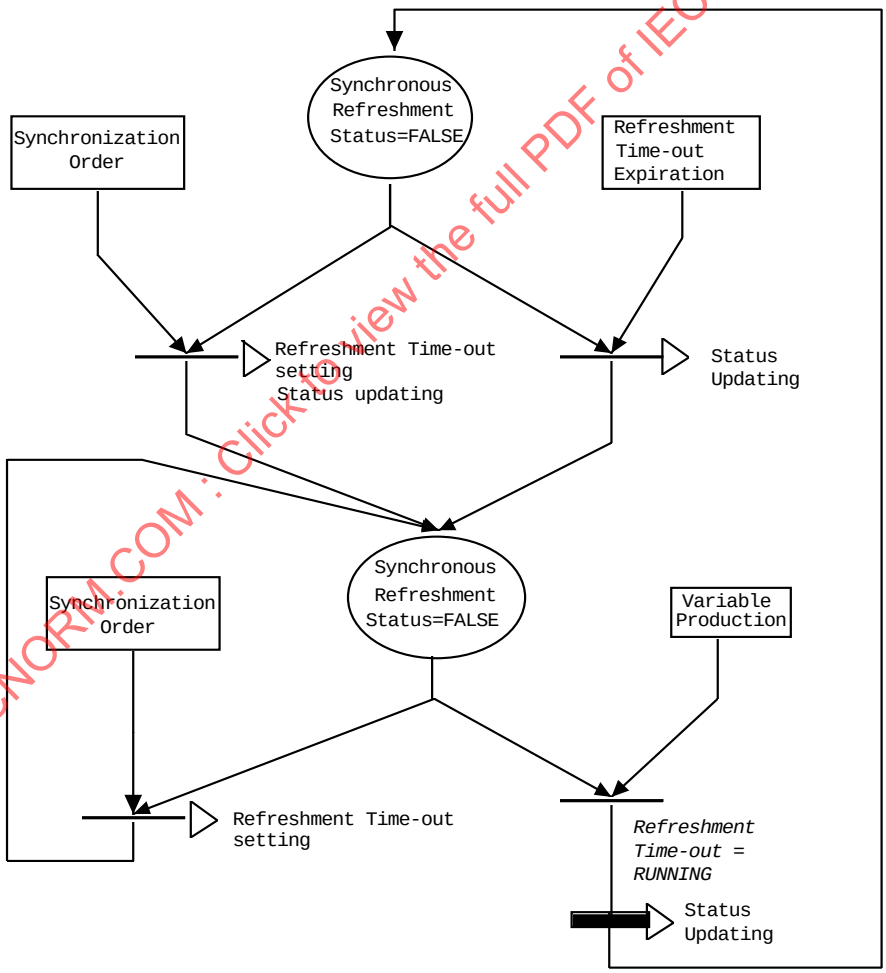
Anglais	Français
Variable Production	Production de variable
Refreshment time-out Expiration	Expiration de temporisation de rafraîchissement
Refreshment time-out setting	Réglage de temporisation de rafraîchissement
Asynchronous refreshment status = TRUE	Statut de rafraîchissement asynchrone = VRAI
Status updating	Mise à jour de statut
Asynchronous refreshment status=FALSE	Statut de rafraîchissement asynchrone = FAUX

Figure 7 – Diagramme d'évaluation du statut de rafraîchissement asynchrone

Tableau 7 – Événements et actions de rafraîchissement asynchrone

État initial:	**Statut de rafraîchissement asynchrone = FALSE**
	État du délai d'attente de rafraîchissement = IDLE
Demandes:	**Production de variable:**
	Invocation par l'utilisateur du service A_WRITELOC, A_WRITEFAR ou A_WRITE
	Expiration du délai d'attente de rafraîchissement:
	Transition de la temporisation associée au mécanisme d'évaluation de rafraîchissement asynchrone de l'état *RUNNING* à l'état *IDLE*
Action:	**Paramétrage du délai d'attente de rafraîchissement:**
	Transition de la temporisation associée au mécanisme de rafraîchissement asynchrone de l'état *IDLE* à l'état *RUNNING*, avec un préréglage égal à la valeur de l'attribut *Production Period*.
	Mise à jour du statut:
	Octroi de la valeur du statut de rafraîchissement asynchrone au réseau.

Le statut de rafraîchissement synchrone associé à une variable au sein d'une entité d'application productrice est élaboré par le mécanisme décrit à la Figure 8 et dans le Tableau 8.



Légende

Anglais	Français
Variable Production	Production de variable
Refreshment time-out Expiration	Expiration de temporisation de rafraîchissement
Refreshment time-out setting	Réglage de temporisation de rafraîchissement

Synchronous refreshment status = FALSE	Statut de rafraîchissement synchrone = FAUX
Status updating	Mise à jour de statut
Refreshment time-out =RUNNING	Temporisation de rafraîchissement =EN MARCHE
Synchronisation order	Ordre/commande de synchronisation

Figure 8 – Diagramme d'évaluation de statut de rafraîchissement synchrone

Tableau 8 – Événements et actions de rafraîchissement synchrone

État initial: **Statut de rafraîchissement synchrone = FALSE**

État du délai d'attente de rafraîchissement = IDLE

Demandes: **Production de variable:**

Invocation par l'utilisateur du service A_WRITELOC, A_WRITEFAR ou A_WRITE

Expiration du délai d'attente de rafraîchissement:

Transition de la temporisation associée au mécanisme d'évaluation de rafraîchissement synchrone de l'état *RUNNING* à l'état *IDLE*

Ordre de synchronisation:

Occurrence de l'ordre de synchronisation associé au mécanisme d'élaboration du rafraîchissement synchrone.

Action: **Paramétrage du délai d'attente de rafraîchissement:**

Transition de la temporisation associée au mécanisme de rafraîchissement synchrone de l'état *IDLE* à l'état *RUNNING*, avec un préréglage égal à la valeur de l'attribut *Production Period*.

Mise à jour du statut:

Octroi de la valeur du statut de rafraîchissement asynchrone au réseau.

6.2.1.3.3.4.3 Rafraîchissement ponctuel

6.2.1.3.3.4.3.1 Définition du rafraîchissement ponctuel

L'information relative à la validité de production est facultative.

Le rafraîchissement ponctuel est relatif à la validité de la disponibilité du réseau par l'utilisateur producteur d'une valeur de variable. Cette validité est élaborée par le biais d'un intervalle de temps fixé par l'utilisateur. Cet intervalle de temps est activé à la réception d'un ordre de synchronisation à partir du réseau.

Rafraîchissement ponctuel:

Lorsqu'il a la valeur TRUE, ce booléen indique au sein d'une entité d'application productrice qu'une valeur de la variable a été fournie au réseau par l'utilisateur producteur (au moins) une fois pendant l'intervalle de temps qui suit la réception d'un ordre de synchronisation. Il convient que la valeur de la variable ne soit pas fournie hors de l'intervalle de temps.

Le statut de rafraîchissement ponctuel est associé à la valeur de variable et est fourni au réseau.

6.2.1.3.3.4.3.2 Spécification de la classe *Punctual Refreshment*

Cette classe collecte tous les attributs relatifs à l'élaboration d'informations de rafraîchissement ponctuel associées à une variable au sein d'une entité d'application productrice.

Class: PUNCTUAL REFRESHMENT

Attribute	: Production time slot
Attribute	: *Inherit from* Time-out
Attribute	: *Reference* (Synchronization) Variable

Attribute : Punctual Refreshment Status [TRUE,FALSE]

Production Time slot:

Le rafraîchissement ponctuel est élaboré par le biais d'un intervalle de temps de production associé à une variable par l'utilisateur producteur. Le rafraîchissement ponctuel permet à un utilisateur consommateur de vérifier si le producteur de variable garantit sa production à l'intérieur de l'intervalle de temps activé par le dernier ordre de synchronisation émis à partir du réseau, et qui lui est associé.

Inherit from Time-out:

La classe *Punctual Refreshment* hérite de la classe *time-out*.

La classe *Time-out* comprend tous les attributs relatifs à une temporisation (prédéfini, état) qui caractérisent l'intervalle de temps de production.

***Reference* (Synchronization) Variable:**

Cet attribut fait référence à une classe *Variable* dont l'attribut *Class* a la valeur SYNCHRONIZATION, et qui permet de définir l'ordre de synchronisation qui définit le délai d'attente dans le cas d'un rafraîchissement ponctuel.

Punctual Refreshment Status:

Le rafraîchissement ponctuel d'une variable est caractérisé par un statut.

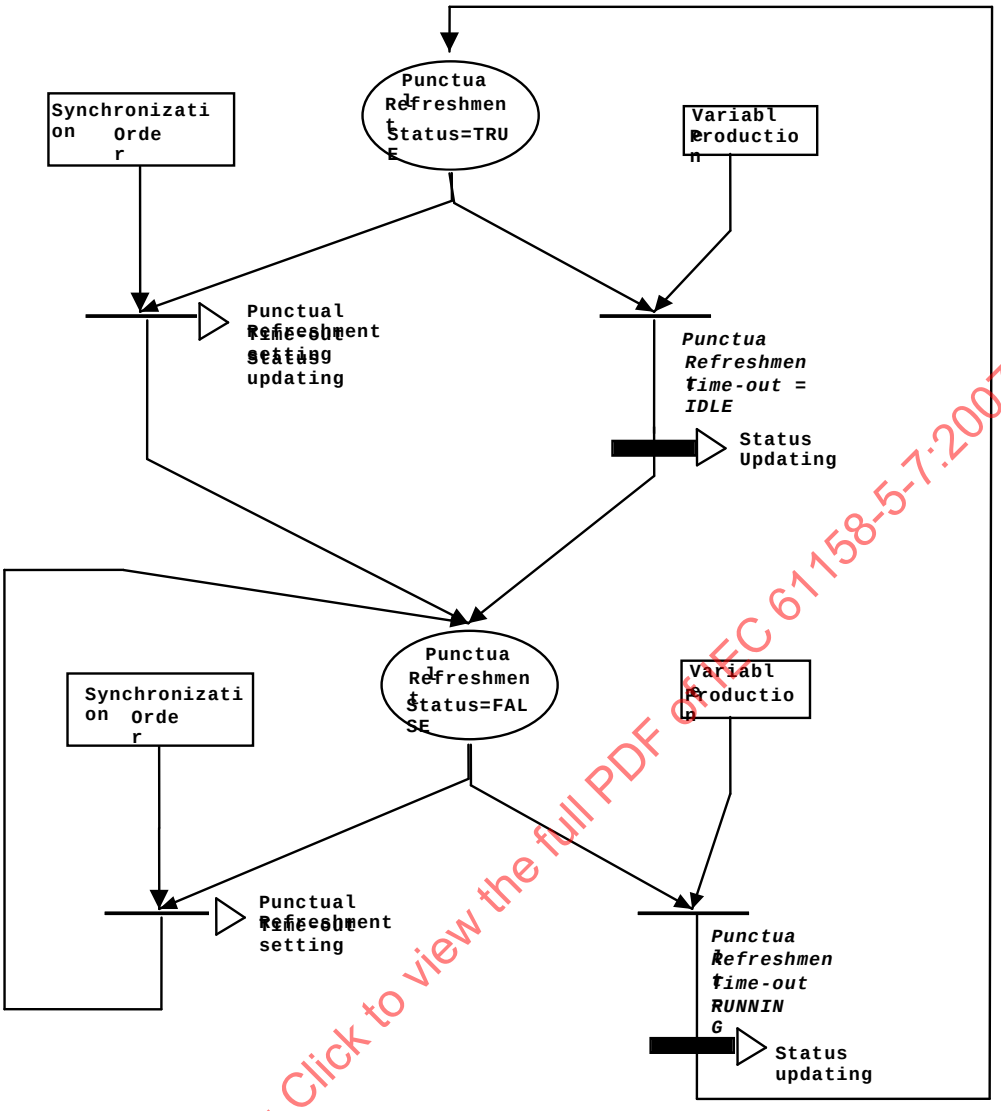
Ce statut est de type booléen et est localement élaboré au sein de chaque entité possédant un objet *Produced Variable Local Image* de la variable.

Le statut de rafraîchissement ponctuel est caractérisé par deux états; les conditions de transition entre les deux états sont élaborées à travers

- des événements correspondants à la production de variable, ou à l'occurrence d'un ordre de synchronisation associé à la variable,
- l'état du délai d'attente associé à la variable pour le traitement du rafraîchissement ponctuel.

6.2.1.3.3.4.3.3 Mécanisme

Le statut de rafraîchissement ponctuel associé à une variable au sein d'une entité d'application productrice est élaboré par le mécanisme décrit à la Figure 9 et dans le Tableau 9.



Légende

Anglais	Français
Synchronization Order	Ordre de synchronisation
Punctual Refreshment Time-out setting	Réglage de la temporisation de rafraîchissement ponctuel
Status updating	Mise à jour de statut
Punctual Refreshment Time-out = IDLE	Temporisation de rafraîchissement ponctuel = REPOS
Variable Production	Production de variable
Punctual Refreshment Status=TRUE	Statut de rafraîchissement ponctuel =VRAI
Punctual Refreshment Status=FALSE	Statut de rafraîchissement ponctuel =FAUX
Punctual Refreshment Time-out = RUNNING	Temporisation de rafraîchissement ponctuel = EN MARCHE

Figure 9 – Diagramme d'évaluation de statut de rafraîchissement ponctuel

Tableau 9 – Événements et actions de rafraîchissement ponctuel

État initial:	**Statut de rafraîchissement ponctuel = FALSE**
	État du délai d'attente de rafraîchissement = IDLE
Demandes:	**Production de variable:**
	Invocation par l'utilisateur du service A_WRITELOC, A_WRITEFAR ou A_WRITE
	Expiration du délai d'attente de rafraîchissement:
	Transition de la temporisation associée au mécanisme d'évaluation de rafraîchissement ponctuel de l'état *RUNNING* à l'état *IDLE*
	Ordre de synchronisation:
	Occurrence de l'ordre de synchronisation associé au mécanisme d'élaboration du rafraîchissement ponctuel.
Action:	**Paramétrage du délai d'attente de rafraîchissement:**
	Transition de la temporisation associée au mécanisme de rafraîchissement ponctuel de l'état *IDLE* à l'état *RUNNING,* avec un préréglage égal à la valeur de l'attribut *Production Time Slot*.
	Mise à jour du statut:
	Octroi de la valeur du statut de rafraîchissement asynchrone au réseau.

6.2.1.3.4 Services de base

6.2.1.3.4.1 Vue d'ensemble des services de base

Les services de base d'accès de variables dans l'environnement MPS permettent à l'utilisateur d'accéder aux objets d'application associés identifiés par leur spécification.

La spécification de variable correspond:

- au nom de l'application pour les variables d'accès global. Dans ce cas, le nom d'application correspond au paramètre *A_Name* des objets *Consumed Variable Local Image* ou *Produced Variable Local Image* ou *Third part Variable Local Image*.
- ou au nom d'accès explicite partiel, soit pour les variables structurées soit pour les variables de type array accessibles partiellement. Ce nom d'accès est construit avec la variable d'application *A_Name* suivie par le chemin d'accès qui correspond à l'énumération successive des noms de champs de l'arbre de la structure et/ou des éléments d'indices d'une matrice.
- ou au chemin d'accès dans le cas de l'accès renommé partiel ou global à une variable. Ce nom correspond à l'attribut *A_Name* de l'objet *Variable Access.*

6.2.1.3.4.2 Service Local read

6.2.1.3.4.2.1 Fonctionnalité

Avec ce service, l'utilisateur peut lire une valeur de variable ou un champ d'une valeur de variable structurée ou l'élément d'une valeur de variable de type array qui est disponible dans l'entité de communication locale.

6.2.1.3.4.2.2 Primitives de service

Le service Local read est effectué en utilisant les primitives de service A_READLOC (voir Tableau 10):

A_READLOC.Rq (Argument): Local read request

A_READLOC.Cnf (Result): Local read confirmation

Tableau 10 – Paramètres du service A_Readloc

Nom de paramètre	Req	Cnf
Argument		
Variable specification	M	M (=)
Result(+)		S
Valeur		M
Refreshment status		C
Refreshment status		C
Promptness status		C
Punctual Promptness status		C
Result(-)		S
Type of error		M
NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. Voir 1.2.		

Argument:

Information associée à une demande de lecture locale.

Variable specification:

Correspond à l'identification d'un objet *Produced Variable Local Image* dans l'espace d'adressage des services MPS.

Result (+):

Réponse valide associée à une demande de lecture locale

Value:

C'est la valeur de la variable ou du champ ou de l'élément de matrice spécifié(e) dans la demande. Ce paramètre correspond à l'attribut *Public value* de l'objet *Consumed Variable Local Image* pour les variables non resynchronisées. Pour les variables resynchronisées, ce paramètre correspond à l'attribut *Private value* de l'objet *Consumed Variable Local Image*.

Refreshment status:

Punctual Refreshment status:

Information relative à la validité de production.

Cette information provient de l'attribut *Transmitted status value* de l'objet *Consumed Variable Local Image*.

Promptness status:

Punctual Promptness status:

Information relative à la validité de consommation.

Cette information provenait des attributs ayant le même nom au sein de l'objet *Consumed Variable Local Image*.

Result (-):

Cela indique que le service de lecture locale a échoué.

Type of error:

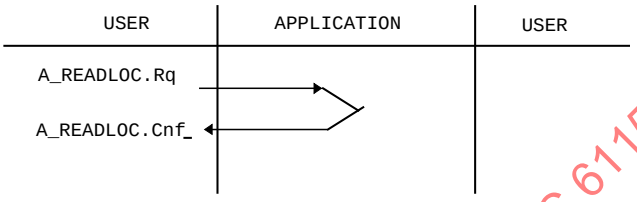
Divers types d'erreurs peuvent être retournés dans la confirmation d'une demande de lecture.

Error_1: La valeur de la variable est temporairement inaccessible.

Error_2: La valeur de la variable n'est pas accessible au moyen de ce service. Ce service correspond à une variable qui n'est pas déclarée localement, ou qui n'est accessible que par les services de production.

Error_3: La variable est déclarée non valide au niveau de la gestion de réseau en référence à la vérification de cohérence de la base de données.

6.2.1.3.4.2.3 Procédure de service



Légende

Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

Figure 10 – Procédure du service A_Readloc

Ce service, représenté à la Figure 10, garantit l'intégrité de valeur de la variable ou du champ ou de l'élément traité(e).

L'intégrité de la valeur d'une variable est assurée dans une opération de lecture lorsque tous les éléments binaires des données associées à cette valeur proviennent de la même mise à jour.

En outre, les demandes d'une variable sont traitées séquentiellement; une nouvelle demande ne peut être effectuée qu'après réception de la confirmation à la demande précédente.

6.2.1.3.4.3 Service Local write

6.2.1.3.4.3.1 Fonctionnalité

Avec ce service, l'utilisateur peut écrire une valeur de variable ou un champ d'une valeur de variable structurée ou un élément d'une valeur de variable de type array dans l'entité de communication locale.

6.2.1.3.4.3.2 Primitives de service

Le service Local write est effectué en utilisant la primitive de service A_WRITELOC (voir Tableau 11).

A_WRITELOC.Rq (Argument) : Demande d'écriture locale.

A_WRITELOC.Cnf (Result) : Confirmation d'écriture locale.

Tableau 11 – Paramètres du service A_Writeloc

Nom de paramètre	Req	Cnf
Argument		
Variable specification	M	M (=)
Valeur	M	
Result(+)		S
Result(-)		S
Type of error		M
NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. Voir 1.2.		

Argument:

Information associée à une demande d'écriture locale

Variable Specification:

Correspond à l'identification d'un objet *Consumed Variable Local Image* dans l'espace d'adressage des services MPS. La définition de ce paramètre correspond à celle indiquée dans le paragraphe 3.8.

Value:

Valeur qui doit être écrite dans la spécification de variable spécifiée, ou le champ spécifié d'une variable structurée ou l'élément spécifié d'une variable de type array.

Ce paramètre correspond à l'attribut *Public value* de l'objet *Produced Variable Local Image* pour les variables non resynchronisées. Pour les variables resynchronisées, ce paramètre correspond à l'attribut *Private value* de l'objet *Produced Variable Local Image*.

Result (+):

Pas de paramètres

Result (-):

Cela indique que le service d'écriture locale a échoué.

Type of error:

Divers types d'erreurs peuvent être retournés dans la confirmation d'une demande de lecture.

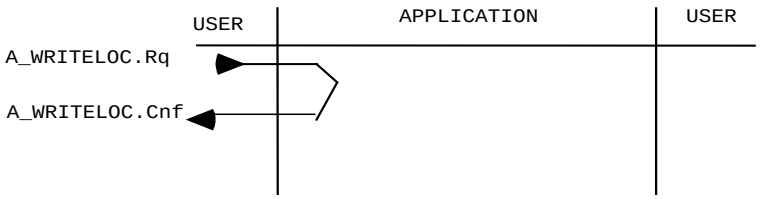
Différents types d'erreurs peuvent être retournés avec la confirmation d'écriture locale.

Error_1: La valeur de la variable est temporairement inaccessible.

Error_2: La variable n'est pas accessible en utilisant ce service. Ce service correspond à une variable qui n'est pas déclarée localement, ou qui n'est accessible que par les services consommateurs.

Error_3: La variable est déclarée non valide au niveau de la gestion de réseau en référence à la vérification de cohérence de la base de données.

6.2.1.3.4.3.3 Procédure de service



Légende

Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

Figure 11 – Procédure du service A_Writeloc

Ce service, représenté à la Figure 11, garantit l'intégrité de la valeur de la variable traitée.

L'intégrité de la variable écrite correspond au fait de fournir au réseau une valeur cohérente provenant de la même opération d'écriture.

Il convient que tous les statuts élaborés par l'entité productrice et fournis au réseau soient cohérents avec la valeur de la variable émise.

En outre, ce service déclenche le mécanisme associé aux éléments de service qui élaborent le statut refreshment (rafraîchissement) et le statut punctual refreshment (rafraîchissement ponctuel) de l'objet *Produced Variable Local Image*.

L'accès partiel sur une variable lors de l'écriture d'une valeur déclenche le mécanisme d'élaboration du statut de validité de production associé à cette variable.

En outre, les demandes de la variable sont traitées séquentiellement; une nouvelle demande ne peut être effectuée qu'après réception de la confirmation à la demande précédente.

6.2.1.3.4.4 Service Updating

6.2.1.3.4.4.1 Fonctionnalité

Ce service prend en charge la demande (au bus administrateur) pour la mise à jour d'une valeur de variable au niveau des entités consommatrices. La valeur de la variable est disponible localement au niveau de l'entité de communication productrice.

Ce service peut être utilisé par l'entité d'application qui produit la variable par une entité d'application qui la consomme, ou par une entité d'application qui n'est ni productrice ni consommatrice (c'est-à-dire: une entité tierce).

6.2.1.3.4.4.2 Primitives de service

Une demande de mise à jour est effectuée avec les primitives de service suivantes, comme indiqué dans le Tableau 12.

A_UPDATE.Rq (Argument) : Updating request.

A_UPDATE.Cnf (Result) : Updating confirmation.

Tableau 12 – Paramètres du service A_Update

Nom de paramètre	Req	Cnf
Argument		
Variable specification	M	M (=)
Priority	M	
Result(+)		S
Result(-)		S
Type of error		M
NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. Voir 1.2.		

Arguments:

Informations associées à une demande de mise à jour.

Variable Specification:

Correspond à l'identification d'un objet *Produced Variable Local Image* ou d'un objet *Produced Variable Local Image* ou d'un objet *Third party Variable Local Image* dans l'espace d'adressage des services MPS

Priority:

Ce paramètre permet à l'utilisateur de choisir entre deux possibilités de contrôle de flux **Urgent** et **Normal** pour demander la mise à jour d'une variable dont le nom a été spécifié dans la demande.

Le contrôle de flux garantit qu'une demande urgente déclenchera une mise à jour avant une demande normale ultérieure.

Result (+):

Pas de paramètres.

Result (-):

Cela indique que le service updating a échoué.

Type of error:

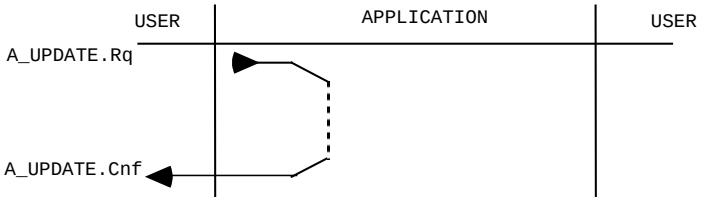
Divers types d'erreurs peuvent être retournés dans la confirmation de service:

Error_1: La variable n'est pas déclarée au niveau de l'AP de l'initiateur de la demande.

Error_2: La demande ne peut pas être acceptée compte tenu des limites locales relatives au nombre de demandes en attente.

Error_3: La variable est déclarée non valide au niveau de la gestion de réseau en référence à la vérification de cohérence de la base de données.

6.2.1.3.4.4.3 Procédure de service



Légende

Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

Figure 12 – Procédure du service A_Update

Ce service, représenté à la Figure 12, garantit que l'utilisateur recevra une confirmation après une demande.

La confirmation de service est positive lorsque la demande est confirmée positive au niveau de la liaison de données.

Les demandes peuvent être enchaînées ensemble avant même de recevoir toutes les confirmations. Cependant, les confirmations ne sont pas toujours émises dans l'ordre des demandes précédemment initiées.

Pour une variable de la classe SYNCHRONIZATION, ce service ne peut être initié que par le producteur de variable.

NOTE Le nombre maximal de demandes en attente dépend de la mise en œuvre, et la saturation est spécifiée dans des codes d'erreur associés à la confirmation.

6.2.1.3.4.5 Service Remote read

6.2.1.3.4.5.1 Fonctionnalité

Avec ce service, l'utilisateur consommateur d'une variable peut prendre l'initiative d'un échange d'informations sur le bus, avant de procéder à une lecture sur la nouvelle valeur de la variable disponible localement.

6.2.1.3.4.5.2 Primitives de service

Le service Remote read est effectué par la primitive de service A_READFAR (voir Tableau 13).

A_READFAR.Rq (Argument): Demande de lecture distante

A_READFAR.Cnf (Result): Confirmation de lecture distante avec retour de la valeur lue

Tableau 13 – Paramètres du service A_Readfar

Nom de paramètre	Req	Cnf
Argument		
Variable specification	M	M (=)
Priority	M	
Result(+)		S
Valeur		M
Refreshment Status		C
Punctual Refreshment Status		C
Promptness Status		C
Punctual Promptness status		C
Result(-)		S
Type of error		M

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument:

Information associée à une demande de lecture distante.

Variable specification:

Correspond à l'identification d'un objet *Consumed Variable Local Image* dans l'espace d'adressage des services MPS.

Priority:

Ce paramètre permet à l'utilisateur de choisir entre deux possibilités de contrôle de flux **Urgent** et **Normal** pour demander la mise à jour d'une variable dont le nom a été spécifié dans la demande.

Le contrôle de flux garantit qu'une demande urgente déclenchera une mise à jour avant une demande normale ultérieure.

Result (+):

Ce résultat indique que le service demandé a réussi.

Value:

Valeur de la variable ou du champ ou d'un élément de matrice spécifié(e) dans la demande.

Ce paramètre correspond à l'attribut *Public value* de l'objet *Consumed Variable Local Image* pour les variables non resynchronisées. Pour les variables resynchronisées, ce paramètre correspond à l'attribut *Private value* de l'objet *Consumed Variable Local Image*.

Refreshment status:

Punctual Refreshment status:

Information relative à la validité de production. Cette information provient de l'attribut *Transmitted status value* de l'objet *Consumed Variable Local Image*.

Promptness status:

Punctual Promptness status:

Information relative à la validité de consommation

Cette information provient des attributs ayant le même nom au sein de l'objet *Consumed variable Local Image*.

Result (-):

Cela indique que le service demandé a échoué.

Type of error:

Divers types d'erreurs peuvent être retournés dans la confirmation de service:

Error_1: La valeur de la variable est temporairement inaccessible.

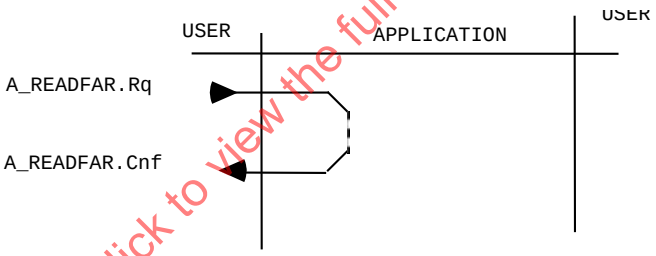
Error_2: La valeur de la variable n'est pas accessible au moyen de ce service. Ce service correspond à une variable qui n'est pas déclarée localement, ou qui n'est accessible que par les services de production.

Error_3: La variable est déclarée non valide au niveau de la gestion de réseau en référence à la vérification de cohérence de la base de données.

Error_4: La demande ne peut pas être acceptée compte tenu des limites locales relatives au nombre de demandes en attente.

Error_5: La demande a été interrompue en raison de l'expiration de la temporisation associée à l'attente de la réception de la valeur de variable.

6.2.1.3.4.5.3 Procédure de service



Légende

Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

Figure 13 – Procédure du service A_Readfar

Ce service, représenté à la Figure 13, garantit l'intégrité de la valeur de la variable ou du champ ou de la valeur de l'élément de matrice traité(e).

Les demandes peuvent être enchaînées ensemble. Une nouvelle demande peut être émise sans attendre les confirmations correspondantes. Toutefois, les confirmations ne sont pas toujours émises dans le même ordre que les demandes.

Ce service garantit que l'utilisateur recevra une confirmation après une demande.

L'intégrité de la valeur d'une valeur de variable est assurée dans une opération de lecture lorsque tous les éléments binaires des données associées à cette valeur proviennent de la même mise à jour.

6.2.1.3.4.6 Service Remote write

6.2.1.3.4.6.1 Fonctionnalité

Avec ce service, l'utilisateur producteur de la variable peut initier un échange d'informations sur le bus après avoir fourni la valeur de la variable au réseau.

6.2.1.3.4.6.2 Primitives de service

Le service Remote write est effectué en utilisant la primitive de service A_WRITEFAR (voir Tableau 14).

A_WRITEFAR.Rq (Argument): Demande Remote write

A-WRITEFAR.Cnf (Result): Confirmation Remote write

Tableau 14 – Paramètres du service A_Writefar

Nom de paramètre	Req	Cnf
Argument		
Variable specification	M	M (=)
Priority	M	M (=)
Valeur	M	
Result(+)		S
Result(-)		S
Type of error		M

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. Voir 1.2.

Arguments:

Information associée à une demande distante

Variable specification:

Correspond à l'identification d'un objet *Produced Variable Local Image* dans l'espace des services MPS.

Priority:

Ce paramètre permet à l'utilisateur de choisir entre deux possibilités de contrôle de flux **Urgent** et **Normal** pour demander la mise à jour d'une variable dont le nom a été spécifié dans la demande.

Le contrôle de flux garantit qu'une demande urgente déclenchera une mise à jour avant une demande normale ultérieure.

Value:

Lors d'une écriture distante, l'utilisateur associe la valeur qu'il entend affecter à la spécification de la variable.

Ce paramètre correspond à l'attribut *Public value* de l'objet *Produced Variable Local Image* pour les variables non resynchronisées. Pour les variables resynchronisées, ce paramètre correspond à l'attribut *Private value* de l'objet *Produced Variable Local Image*.

Result (+):

Pas de paramètres.

Result (-):

Cela indique que le service remote write a échoué.

Type of error:

Différents types d'erreurs peuvent être retournés dans la confirmation de service:

Error_1: La valeur de la variable est temporairement inaccessible.

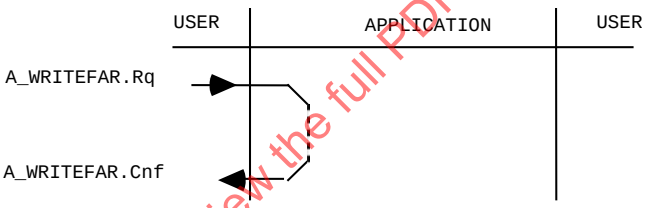
Error_2: La variable n'est pas accessible au moyen de ce service. Ce service correspond à une variable qui n'est pas déclarée localement, ou qui n'est accessible que par les services consommateurs.

Error_3: La variable est déclarée non valide au niveau de la gestion de réseau en référence à la vérification de cohérence de la base de données.

Error_4: La demande du service "remote write" ne peut pas être acceptée en raison des limites locales relatives au nombre de demandes en attente. Toutefois, la valeur de la variable locale a été mise à jour.

Error_5: La demande a été supprimée

6.2.1.3.4.6.3 Procédure de service



Légende

Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

Figure 14 – Procédure du service A_Writefar

Ce service, représenté à la Figure 14, garantit l'intégrité de valeur de la variable ou du champ ou de l'élément traité(e).

Les demandes peuvent être enchaînées ensemble. Une nouvelle demande peut être émise sans attendre les confirmations correspondantes. Toutefois, les confirmations ne sont pas toujours émises dans le même ordre que les demandes.

Ce service assure que l'utilisateur recevra une confirmation après une demande.

L'intégrité d'une valeur de variable écrite correspond à une mise à jour du réseau avec une valeur cohérente provenant d'une seule opération d'écriture. Ce service assure également la cohérence entre la valeur de la variable produite et les statuts liés à la validité de la production, qui sont facultativement associés.

En outre, ce service déclenche les mécanismes associés aux éléments de service qui élaborent la valeur du statut refreshment (rafraîchissement) et du statut punctual refreshment (rafraîchissement ponctuel) de l'objet *Produced Variable Local Image*.

NOTE La valeur fournie aux entités consommatrices sera celle qui était disponible au niveau du producteur pendant la mise à jour du réseau.

6.2.1.3.4.7 Service d'indication d'émission

6.2.1.3.4.7.1 Fonctionnalité

Ce service est facultatif et indique à un utilisateur producteur qu'une variable a été émise sur le réseau.

6.2.1.3.4.7.2 Primitives de service

Une indication d'émission est effectuée en utilisant la primitive de service A_SENT, comme montrée dans le Tableau 15.

A_SENT.Ind (Result): Indication d'émission d'une valeur de variable

Tableau 15 – Paramètres du service A_Sent

Nom de paramètre	Ind
Result	
Variable specification	M
Result(+)	S
Result(-)	S
Type of error	M

Result:

Variable specification:

Correspond à l'identification d'un objet *Produced Variable Local Image* dans l'espace d'adressage des services MPS.

Result(+):

Pas de paramètres

Result(-):

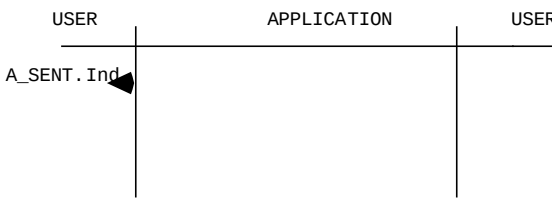
Les résultat indiquent que le service a échoué.

Type of error:

Divers types d'erreurs peuvent être retournés par le service d'indication d'émission:

Error_1: La valeur de la variable émise est déclarée non valide.

6.2.1.3.4.7.3 Procédure de service



Légende

Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

Figure 15 – Procédure du service A_Sent

Ce service, représenté à la Figure 15, informe l'utilisateur producteur de la diffusion d'une variable sur le réseau.

Lorsque cette variable ou certains de ses champs sont connus localement au niveau de l'utilisateur producteur par le biais de noms provenant d'un accès partiel renommé, ce service fournit une indication à l'utilisateur pour chacun des noms associés à la même variable.

6.2.1.3.4.8 Service d'indication de réception

6.2.1.3.4.8.1 Fonctionnalité

Ce service informe un utilisateur consommateur qu'une variable a été mise à disposition par le réseau.

6.2.1.3.4.8.2 Primitives de service

L'indication de réception est accomplie en utilisant la primitive de service A_RECEIVED, montrée dans le Tableau 16.

A_RECEIVED.Ind(Result): Indication de réception de variable

Tableau 16 – Paramètres du service A_Received

Nom de paramètre	Ind
Result	
Variable specification	M
Result(+)	S
Result(-)	S
Type of error	M

Result:

Variable specification:

Correspond à l'identification d'un objet *Consumed Variable Local Image* dans l'espace d'adressage des services MPS.

Result(+):

Pas de paramètres

Result (-):

Cela indique que le service indication a échoué.

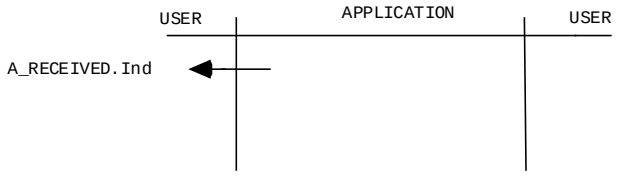
Type of error:

Différents types d'erreurs peuvent être retournés par le service d'indication de réception:

Error_1: La valeur de la variable reçue est déclarée non valide par l'entité d'application productrice.

Error_2: L'émission de cette variable a échoué.

6.2.1.3.4.8.3 Procédure de service



Légende

Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

Figure 16 – Procédure du service A_Received

Ce service, représenté à la Figure 16, informe l'utilisateur concernant la transmission d'une variable par le réseau.

6.2.1.3.5 Services améliorés

6.2.1.3.5.1 Définition des services améliorés

Les services améliorés constituent un autre niveau d'abstraction et s'appuient sur les services de base pour permettre à l'utilisateur d'accéder aux variables sans avoir besoin de spécifier ni le type de service (local ou distant) ni la priorité utilisée.

6.2.1.3.5.2 Service Universal read

6.2.1.3.5.2.1 Fonctionnalité

Ce service permet à l'utilisateur de lire une valeur de variable sans connaître le type de service "read" (local ou distant) qui doit être utilisé; le service est déterminé par la valeur de l'attribut *Universal services range* de la classe *Identified Variable.*

De plus, lorsqu'il s'agit d'une lecture à distance, la priorité utilisée par le contrôle de flux est déterminée par la valeur de l'attribut *Priority* de la classe *Identified Variable.*

6.2.1.3.5.2.2 Primitives de service

Le service universal read est effectué en utilisant les primitives de service A_READ, montrées dans le Tableau 17.

A_READ.Rq (Argument): Universal read request

A_READ.Cnf (Result): Confirmation de lecture universelle avec la valeur lue

Tableau 17 – Paramètres du service A_Read

Nom de paramètre	Req	Cnf
Argument		
Variable specification	M	M (=)
Result(+)		S
Value		M
Refreshment Status		C
Punctual Refreshment Status		C
Promptness Status		C
Punctual Promptness status		C
Result(-)		S
Type of error		M
NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. Voir 1.2.		

Argument:

Information associée à une demande de lecture universelle.

Variable specification:

Correspond à l'identification d'un objet *Consumed Variable Local Image* dans l'espace d'adressage des services MPS.

Result (+):

Réponse valide associée à la demande de lecture universelle.

Value:

C'est la valeur de la variable ou du champ ou de l'élément de matrice spécifié(e) dans la demande. Ce paramètre correspond à l'attribut Public value de l'objet *Consumed Variable Local Image* pour les variables non resynchronisées. Pour les variables resynchronisées, ce paramètre correspond à l'attribut *Private value* de l'objet *Consumed Variable Local Image*.

Refreshment status:

Punctual Refreshment status:

Information relative à la validité de production.

Cette information est dérivée de l'attribut *Transmitted status value* de l'objet *Consumed Variable Local Image*.

Promptness status:

Punctual promptness status:

Information relative à la validité de consommation.

Cette information est dérivée des attributs ayant le même nom au sein de l'objet *Consumed Variable Local Image*.

Result(-):

Cela indique que le service de lecture universelle a échoué.

Type of error:

Divers types d'erreurs peuvent être retournés dans la confirmation de service:

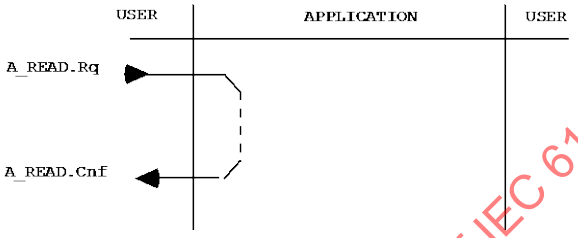
Error_1: La valeur de la variable est temporairement inaccessible.

Error_2: La valeur de la variable n'est pas accessible en utilisant ce service. Ce service correspond à une variable qui n'est pas déclarée localement, ou qui n'est accessible que par les services de production.

Error_3: La variable est déclarée non valide au niveau de la gestion de réseau en référence à la vérification de cohérence de la base de données.

Error_4: La demande ne peut pas être acceptée compte tenu des limites locales relatives au nombre de demandes en attente.

6.2.1.3.5.2.3 Procédure de service



Légende

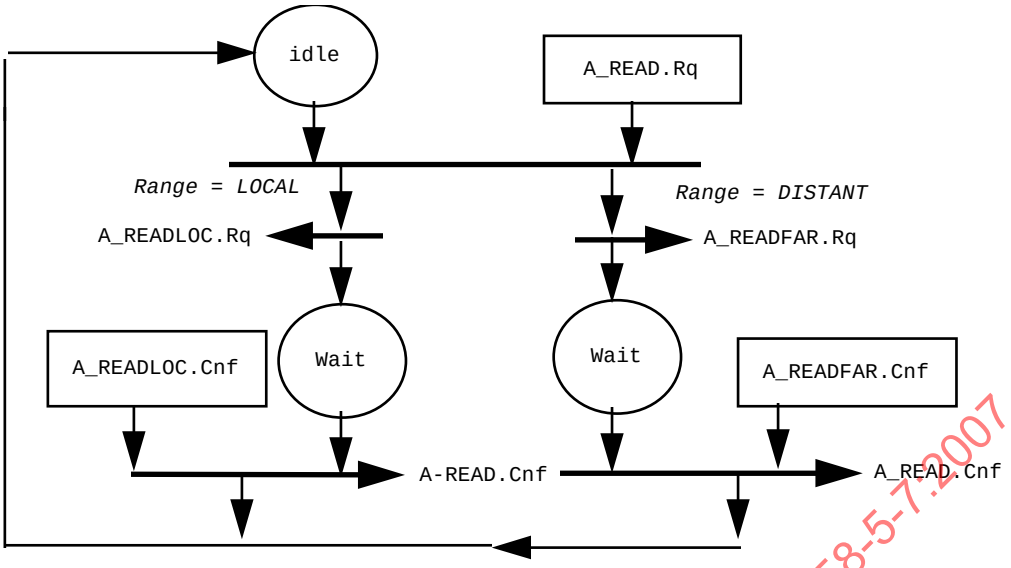
Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

Figure 17 – Procédure du service A_Read

Selon la valeur de l'attribut *Universal services range* de l'objet *Consumed Variable Local Image*, l'invocation de ce service implique soit l'appel du service local "read", soit l'appel du service distant "read". La procédure de service de lecture universelle, Figure 17, est celle qui est associée au service invoqué (local ou distant).

La priorité utilisée dans le cas d'une invocation de service distant dépend de la valeur de l'attribut *Priority* de la classe *Identified variable*.

La Figure 18 montre le mécanisme relatif au chaînage des services de base:



Légende

Anglais	Français
Range = LOCAL	Portée = locale
idle	Repos
Wait	Attente
Range = DISTANT	Portée = distante

Figure 18 – Diagramme d'états du service A_Read

6.2.1.3.5.3 Service Universal write

6.2.1.3.5.3.1 Fonctionnalité du service Universal write

Ce service permet à l'utilisateur d'effectuer une écriture de la valeur de la variable sans avoir à connaître le type de service "write" (local ou distant) qui doit être utilisé. Le service est déterminé par la valeur de l'attribut *Universal service range* de l'objet *Produced Variable Local Image*.

De plus, lorsqu'il s'agit d'une écriture à distance, la priorité utilisée par le contrôle de flux est déterminée par la valeur de l'attribut *Priority* de la classe *Identified Variable.*

6.2.1.3.5.3.2 Primitives de service

Le service universal write est effectué en utilisant les primitives de service A_WRITE, montrées dans le Tableau 18.

A_WRITE.Rq (Argument) : Demande Universal write

A_WRITE.Cnf (Result) : Confirmation Universal write

Tableau 18 – Paramètres du service A_Write

Nom de paramètre	Req	Cnf
Argument		
Variable specification	M	M (=)
Priority	M	
Result(+)		S
Result(-)		S
Type of error		M

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. Voir 1.2.

Arguments:

Information associée à une demande d'écriture universelle.

Variable specification:

Correspond à l'identification d'un objet Produced Variable Local Image dans l'espace d'adressage des services MPS.

Value:

C'est la valeur de la variable ou du champ ou de l'élément de matrice spécifié(e) dans la demande. Ce paramètre correspond à l'attribut *Public Value* de l'objet *Produced Variable Local Image* pour les variables non resynchronisées. Pour les variables synchronisées, ce paramètre correspond à l'attribut *Private Value* de l'objet *Produced Variable Local Image*.

Result(+):

Pas de paramètres.

Result(-):

Cela indique que le service universal write a échoué.

Type of error:

Différents types d'erreurs peuvent être retournés dans la confirmation d'une écriture universelle:

Error_1: La valeur de la variable est temporairement inaccessible.

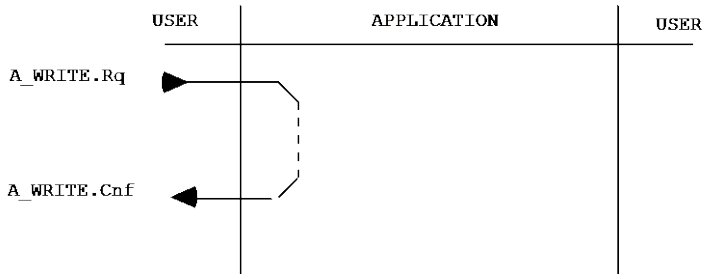
Error_2: La valeur de la variable n'est pas accessible au moyen de ce service. Ce service correspond à une variable qui n'est pas déclarée localement, ou qui n'est accessible que par les services consommateurs.

Error_3: La variable est déclarée non valide au niveau de la gestion de réseau en référence à la vérification de cohérence de la base de données.

Error_4: La demande ne peut pas être acceptée en raison des limites locales relatives au nombre de demandes en attente. Toutefois, la valeur de la variable locale a été mise à jour.

Error_5: La demande a été supprimée

6.2.1.3.5.3.3 Procédure de service



Légende

Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

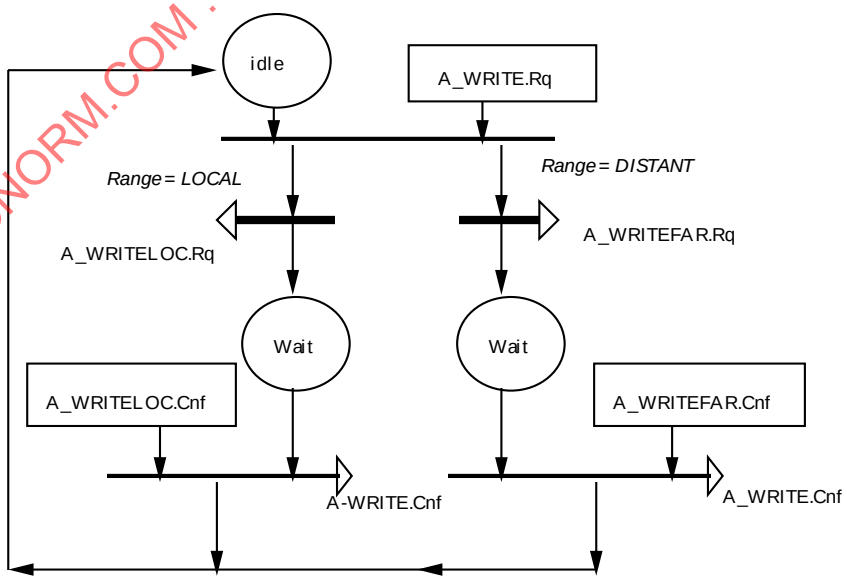
Figure 19 – Procédure du service A_Write

Selon la valeur de l'attribut *Universal services range* de l'objet *Produced Variable Local Image*, l'invocation de ce service implique soit l'appel du service local "write", soit l'appel du service distant "write". La procédure de service d'écriture universelle, Figure 19, est celle qui est associée au service invoqué (local ou distant).

La priorité utilisée dans le cas d'une invocation de service distant dépend de la valeur de l'attribut *Priority* de la classe *Identified variable*.

Ce service déclenche le mécanisme associé aux éléments de service qui élaborent la valeur du statut refreshment (rafraîchissement) et la valeur du statut punctual refreshment (rafraîchissement ponctuel) de l'objet *Produced Variable Local Image*.

La Figure 20 montre le mécanisme relatif au chaînage des services de base utilisés par le service d'écriture universelle:



Légende

Anglais	Français
Range = LOCAL	Portée = locale

idle	Repos
Wait	Attente
Range = DISTANT	Portée = distante

Figure 20 – Diagramme d'états du service A_Write

6.2.1.3.6 Mécanismes avancés

6.2.1.3.6.1 Synchronisation

6.2.1.3.6.1.1 Concept d'application synchrone

Les services de synchronisation fournis par les éléments de service MPS ont été définis afin de permettre l'exécution des applications distribuées.

Une application distribuée est composée de plusieurs Processus d'Application (AP, Application Processes) dans plusieurs appareils du système de commande.

Entre les AP qui participent à une application distribuée, il est nécessaire de distinguer:

– ceux qui sont appelés ASYNCHRONOUS, parce que leur exécution est indépendante du réseau,

– ceux qui sont appelés SYNCHRONOUS, parce que leur exécution est associée aux indications émises par le réseau; ces indications peuvent être périodiques ou non périodiques.

Plusieurs AP synchrones, dont l'exécution est associée aux mêmes indications, forment une application distribuée synchrone.

6.2.1.3.6.1.2 Besoins de resynchronisation

Certains AP ne peuvent prendre en charge que le mode de fonctionnement asynchrone, c'est-à-dire qu'ils sont gérés indépendamment de l'environnement de communication.

NOTE Ce mode de fonctionnement exclusivement asynchrone est généralement la caractéristique des AP situés sur des appareils dont l'âge ou la simplicité n'offre pas la possibilité de synchronisation.

Le mécanisme de resynchronisation permet la fonction d'échantillonnage / de maintien des valeurs de variables produites et consommées par un AP de manière asynchrone par rapport au réseau. Cette fonction permet aux AP asynchrones de participer à une application distribuée synchrone.

6.2.1.3.6.1.3 Variables de synchronisation

Les variables de l'environnement MPS décrites dans le chapitre précédent correspondent à des éléments passifs qui sont échangés sur le réseau.

Les variables de synchronisation sont des éléments associés à des actions concernant:

– traitement de synchronisation au niveau de l'entité d'application productrice et de l'entité d'application consommatrice de cette variable de synchronisation,

– ordres d'exécution de procédure pour différents AP dans une application distribuée.

Au niveau de l'entité d'application, certaines variables de l'environnement MPS tiennent compte de l'événement associé à l'émission d'une variable de synchronisation pour élaborer les informations de validité synchrones, telles que la promptitude synchrone et le rafraîchissement synchrone, ou la promptitude ponctuelle et le rafraîchissement ponctuel.

6.2.1.3.6.1.4 Synchronisation à l'émission et à la réception

Certains services fournis à l'utilisateur peuvent être utilisés pour synchroniser les procédures d'applications distribuées utilisant le réseau.

Parmi ces services, il est nécessaire de distinguer ceux qui permettent:

– la synchronisation au moment de l'émission pour assurer l'activation de différentes tâches locales à l'intérieur d'un seul AP,
– la synchronisation au moment de la réception pour assurer la gestion de l'activation de différentes tâches de plusieurs AP situés dans plusieurs appareils du système de commande.

Ce type de synchronisation est utilisé par une application distribuée synchrone.

6.2.1.3.6.1.5 Définition d'une variable de synchronisation

6.2.1.3.6.1.5.1 Description du modèle de synchronisation

Une variable de synchronisation dans l'environnement MPS est modélisée de la même manière que les variables en utilisant les objets suivants:

– un objet *Produced Variable Local Image* comprenant tous les attributs requis par une variable de synchronisation dans une entité d'application productrice,
– un objet *Consumed Variable Local Image* comprenant tous les attributs requis par une variable de synchronisation dans une entité d'application consommatrice.

Ces deux types d'objet correspondent à des instanciations particulières de la classe *Identified variable* pour laquelle l'attribut *Class* a la valeur SYNCHRONIZATION.

Un objet *Produced Variable Local Image* ou *Consumed Variable Local Image* associé à une variable de synchronisation diffère de celui associé à une variable normale puisque les valeurs de certains de ses attributs sont prédéterminées.

6.2.1.3.6.1.5.2 Valeurs d'attribut prédéterminées de la classe *Identified variable*

La classe *Identified variable* associée à une variable de synchronisation est caractérisée par des valeurs prédéterminées pour certains de ses attributs:

Class:

Cet attribut prend la valeur: *SYNCHRONIZATION*

Toutes les autres valeurs d'attributs ne sont pas prédéterminées.

6.2.1.3.6.1.5.3 Valeurs prédéterminées des attributs de l'objet *Produced Variable Local Image*

L'objet *Produced Variable Local Image* associé à une variable de synchronisation est caractérisé par des valeurs prédéterminées pour certains de ses attributs:

Resynchronized variable:

Cet attribut prend la valeur: *FALSE*

Toutes les autres valeurs d'attributs ne sont pas prédéterminées.

6.2.1.3.6.1.5.4 Valeurs prédéterminées des attributs de l'objet *Consumed Variable Local Image*

L'objet *Consumed Variable Local Image* associé à une variable de synchronisation est caractérisé par des valeurs prédéterminées pour certains de ses attributs:

Resynchronized variable:

Cet attribut prend la valeur: *FALSE*

Toutes les autres valeurs d'attributs ne sont pas prédéterminées.

6.2.1.3.6.1.5.5 Accès aux variables de synchronisation

Les variables de synchronisation peuvent être traitées par un sous-ensemble de services fournis à l'utilisateur pour des variables normales.

Les services qui fournissent l'accès aux variables de synchronisation sont:

- A_WRITELOC / A_WRITEFAR / A_WRITE
- A_READLOC
- A_SENT
- A_RECEIVED
- A_UPDATE

Tous les autres services offerts à l'utilisateur pour une variable normale, ne sont pas autorisés pour une variable de synchronisation.

6.2.1.3.6.1.5.6 Type de variable de synchronisation

Les variables de synchronisation peuvent avoir une valeur de tout type provenant de l'instanciation de la classe *Type constructor*.

Cependant, il convient qu'une variable de synchronisation utilisée pour satisfaire aux exigences de l'élément de service d'application qui élaborent un statut de cohérence spatiale, contienne dans sa valeur, la valeur d'échange réseau de la liste de variables. Dans ce cas, la variable a un type prédéterminé.

La valeur d'échange est une donnée globale dans l'environnement MPS, produite par une entité d'application spécifique. Cette valeur d'échange qualifie la valeur d'une variable échangée de façon périodique ou non périodique, par rapport à l'échange précédent.

Cette valeur d'échange est accessible par l'utilisateur de la même manière que la valeur d'une variable normale, et elle est également utilisée par un élément de service d'application pour l'élaboration du statut de cohérence spatiale.

Ce type prédéterminé est décrit par l'instance suivante de la classe *Type constructor*.

A_type:	K_UNS16
Construction:	SIMPLE
Classe de type:	UNSIGNED
Size:	16

6.2.1.3.6.2 Resynchronisation

6.2.1.3.6.2.1 Principe du mécanisme de resynchronisation

Le mécanisme de resynchronisation s'applique aux variables de la classe NORMAL afin de fournir la fonction d'échantillonnage / de maintien sur des valeurs de variables asynchrones produites ou consommées. Ce mécanisme de resynchronisation exige que le réseau émette des ordres de synchronisation afin de:

– déclencher l'échantillonnage/le maintien des variables produites par l'utilisateur,

– déclencher l'échantillonnage/le maintien des variables consommées par l'utilisateur.

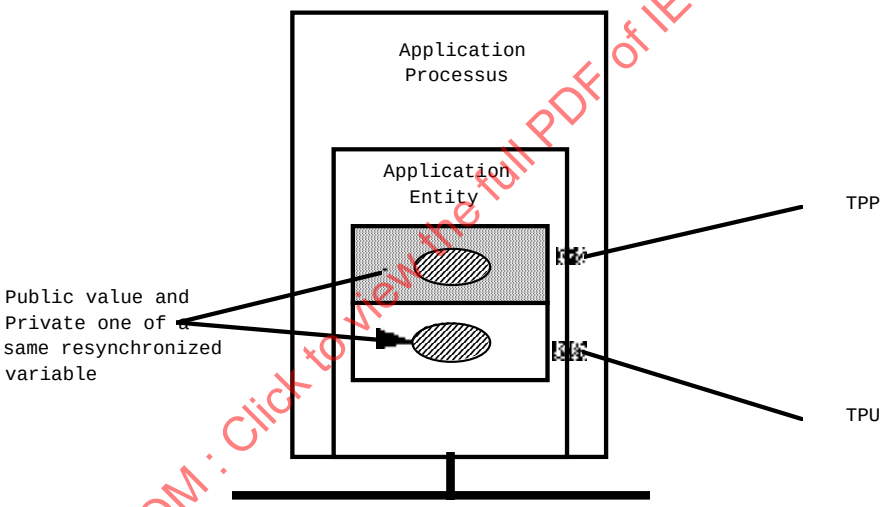
Définition:

Un ordre de resynchronisation correspond à un ordre de synchronisation utilisé par le mécanisme de resynchronisation.

Le mécanisme de resynchronisation associe un deuxième tampon avec une variable au niveau de l'entité d'application (voir Figure 21). Une variable resynchronisée nécessite:

– Un tampon privé (TPP) exclusivement accessible par l'utilisateur.

– Un tampon public (TPU) accessible par le réseau.



Légende

Anglais	Français
Application Processus	Processus d'application
Application Entity	Entité d'application
TPP	TPP (Tampon privé)
TPU	TPU (Tampon public)
Public value and Private one of a same resynchronized variable	Valeur publique et valeur privée de la même variable resynchronisée

Figure 21 – Modèle d'une variable resynchronisée

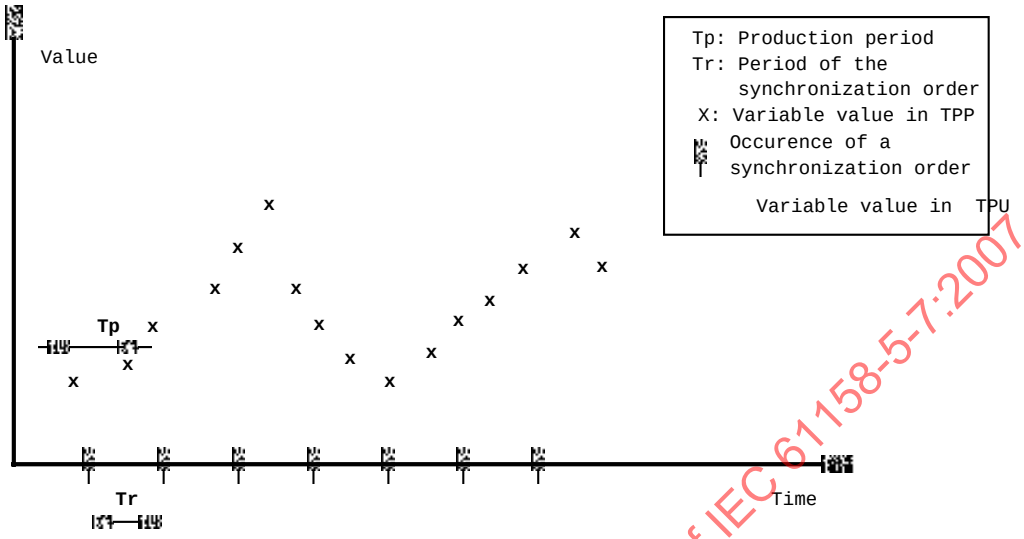
6.2.1.3.6.2.2 Resynchronisation d'une variable produite

6.2.1.3.6.2.2.1 Principe

Supposons qu'un AP produise une variable dans le tampon privé (TPP) avec une période de production Tp. La disponibilité de la valeur de cette variable dans le tampon public (TPU) est synchronisée par le réseau au moyen d'un ordre de resynchronisation. Cet ordre de synchronisation est diffusé avec une période Tr.

Le mécanisme de resynchronisation en production contient la valeur de la variable produite durant l'intervalle de temps entre deux occurrences du même ordre de resynchronisation.

La Figure 22 montre ces principes.



Légende

Anglais	Français
Value	Valeur
Time	Temps
Tp: Production period	Tp: période de production
Tr: Period of the synchronization order	Tr: Période d'ordre/commande de synchronisation
X: Variable value in TPP	X: Valeur de variable dans le TPP
Occurence of a synchronization order	Occurrence d'un ordre/d'une commande de synchronisation
Variable value in TPU	Valeur de variable dans le TPU

Figure 22 – Principes de resynchronisation d'une variable produite

NOTE Le mécanisme de resynchronisation d'une variable produite est utile seulement si $T_r >> T_p$.

Le service de resynchronisation implique l'utilisation de ressources supplémentaires telles que des tampons supplémentaires pour les valeurs privées associées aux valeurs publiques des variables resynchronisées.

Une temporisation de l'élaboration du statut de rafraîchissement et une temporisation de l'élaboration du rafraîchissement ponctuel sont éventuellement associées à chaque tampon public d'une variable resynchronisée.

Le service de resynchronisation nécessite l'utilisation d'une temporisation privée associée au tampon privé afin de permettre au statut de production de préserver sa sémantique lorsque les variables sont resynchronisées.

L'élaboration du statut de validité de production pour une variable resynchronisée au niveau de l'entité productrice implique l'utilisation de deux mécanismes associés aux valeurs publiques et privées de la variable resynchronisation.

6.2.1.3.6.2.2.2 Spécification de la resynchronisation dans la classe de production

Cette classe décrit les ressources supplémentaires utilisées pour resynchroniser une variable produite au niveau de l'entité d'application.

La classe utilisée pour décrire une variable resynchronisée au niveau de l'entité productrice hérite des attributs de la classe *Resynchronization in production*.

Class: RESYNCHRONIZATION IN PRODUCTION
Attribute : Private value
Constraint : Refreshment elaborated = TRUE
Attribute : Private Refreshment status
Attribute : *Inherit from* Time-out
Constraint : Punctual Refreshment elaborated = TRUE
Attribute : Private Punctual Refreshment status
Attribute : *Reference* Synchronization variable

Private value:

Le principe du mécanisme de resynchronisation est basé sur des données qui sont à double mémoire tampon. Il est nécessaire d'ajouter une valeur privée à la valeur publique de la variable. La valeur privée peut maintenant être écrite de façon asynchrone.

Refreshment elaborated:

Cet attribut, tant dans le contexte privé que dans le contexte public conditionne l'existence d'un statut de rafraîchissement associé à une variable.

Lorsque cet attribut a la valeur TRUE, une variable resynchronisée est fournie avec un statut de rafraîchissement et un statut de rafraîchissement privé.

L'attribut conditionnel est déclaré dans la classe *Produced variable*.

Private Refreshment status

Ce statut privé correspond aux informations internes de l'entité de production de la variable resynchronisée à partir de laquelle est construite la valeur du statut de rafraîchissement qui sera émise avec la valeur publique de la variable.

La caractéristique de ce statut privé est SYNCHRONOUS ou ASYNCHRONOUS selon la déclaration globale associée à la variable resynchronisée en production qui est effectuée dans la classe *Refreshment*.

Dans le cas d'un statut synchrone privé, la variable de synchronisation utilisée par le mécanisme privé associé à ce statut est déclarée comme étant une variable globale dans la classe *Refreshment.*

Inherit from Time-out:

Cet ensemble de ressources permet à l'utilisateur de maintenir la sémantique du statut de rafraîchissement associé à une variable resynchronisée.

En fait, l'utilisation d'un statut de rafraîchissement synchrone ou asynchrone nécessite l'utilisation d'un délai d'attente supplémentaire pour préserver la sémantique entière de ce statut.

Punctual Refreshment elaborated:

Cet attribut, tant dans le contexte privé que dans le contexte public conditionne l'existence d'un statut de rafraîchissement ponctuel associé à une variable.

Lorsque cet attribut a la valeur TRUE, une variable resynchronisée est fournie avec un statut de rafraîchissement ponctuel et un statut de rafraîchissement ponctuel privé.

L'attribut conditionnel est déclaré dans la classe *Produced variable*.

Private Punctual Refreshment status:

Ce statut privé correspond aux informations internes de l'entité de production de la variable resynchronisée à partir de laquelle est construite la valeur du statut de rafraîchissement qui sera émise avec la valeur publique de la variable.

Dans le cas d'un statut ponctuel privé, la variable de synchronisation utilisée par le mécanisme privé associé à ce statut est déclarée comme étant une variable globale dans la classe *Punctual Refreshment*.

***Reference* (Synchronization) variable:**

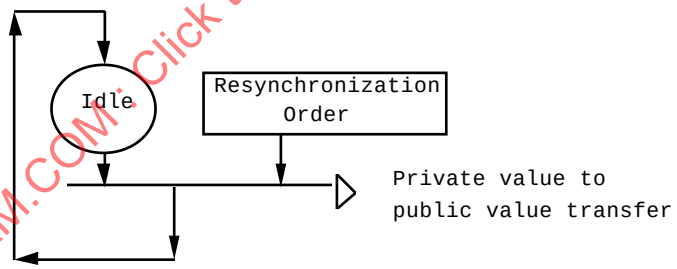
La référence à une variable de synchronisation indique l'ordre de resynchronisation associé à une variable resynchronisée.

Cet ordre de resynchronisation conditionne le transfert des valeurs de variables, les valeurs des statuts de production et les valeurs courantes des temporisations du domaine privé au domaine public. Cette référence indique le *A_Name* de l'objet *Produced Variable Local Image*.

6.2.1.3.6.2.2.3 Mécanisme associé à la valeur produite

Le mécanisme de resynchronisation en production de la valeur de la variable locale est effectué par le transfert du tampon privé vers le tampon public, à la réception de l'ordre de resynchronisation.

Le diagramme (réseau) d'évaluation de cet élément de service qui gère les actions de resynchronisation à l'intérieur d'une entité d'application productrice pour une variable resynchronisée est représenté à la Figure 23.



Légende

Anglais	Français
Idle	Repos
Resynchronization Order	Ordre/commande de resynchronisation
Private value to public value transfer	Transfert de valeur privée vers valeur publique

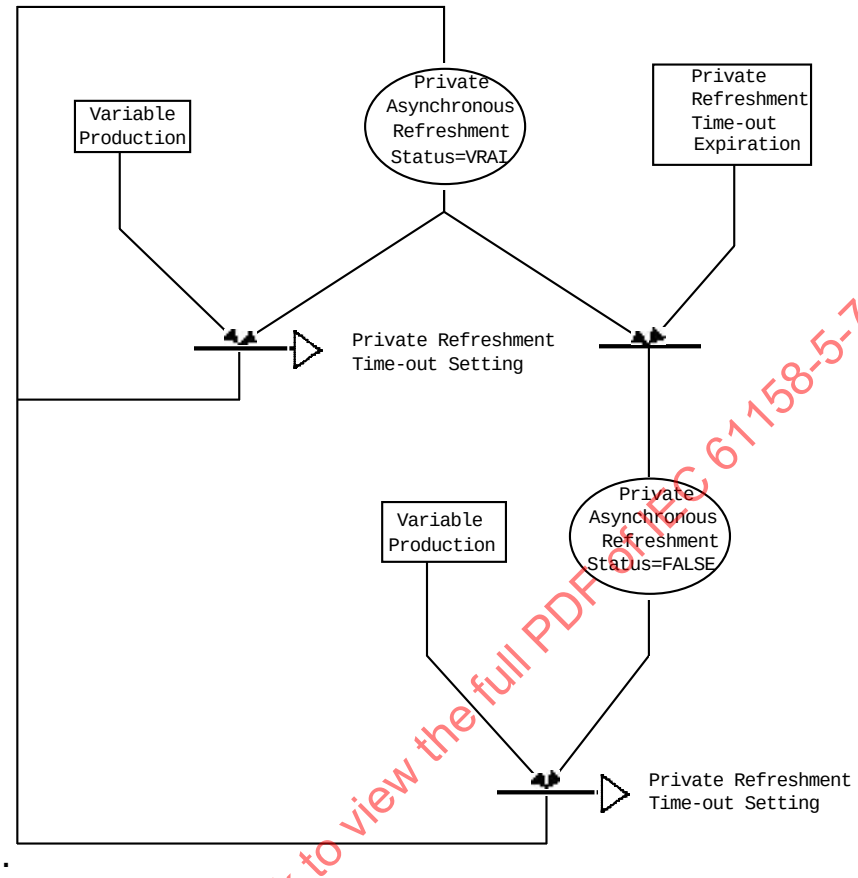
Figure 23 – Diagramme d'états du mécanisme de resynchronisation pour une variable produite

6.2.1.3.6.2.2.4 Mécanisme associé à un rafraîchissement asynchrone

Le mécanisme qui élabore le statut rafraîchissement asynchrone pour une variable resynchronisée peut être décrit avec deux diagrammes d'évaluation, l'un décrit le statut de rafraîchissement asynchrone privé et l'autre le statut de rafraîchissement asynchrone associé à la valeur publique de la variable.

Mécanisme privé:

La temporisation privée associée à un rafraîchissement est prédéfinie avec la période de production de la variable et définie par l'utilisateur avec l'acte d'écriture d'une nouvelle valeur privée (voir, Figure 24 et Tableau 19).



Légende

Anglais	Français
Variable Production	Production de variable
Private Refreshment Time-out Expiration	Expiration de temporisation de rafraîchissement privé
Private Refreshment Time-out Setting	Réglage de temporisation de rafraîchissement privé
Private Asynchronous Refreshment Status=VRAI	Statut de rafraîchissement asynchrone privé =VRAI
Private Asynchronous Refreshment Status=FALSE	Statut de rafraîchissement asynchrone privé =FAUX

Figure 24 – Diagramme d'évaluation de mécanisme privé de rafraîchissement asynchrone

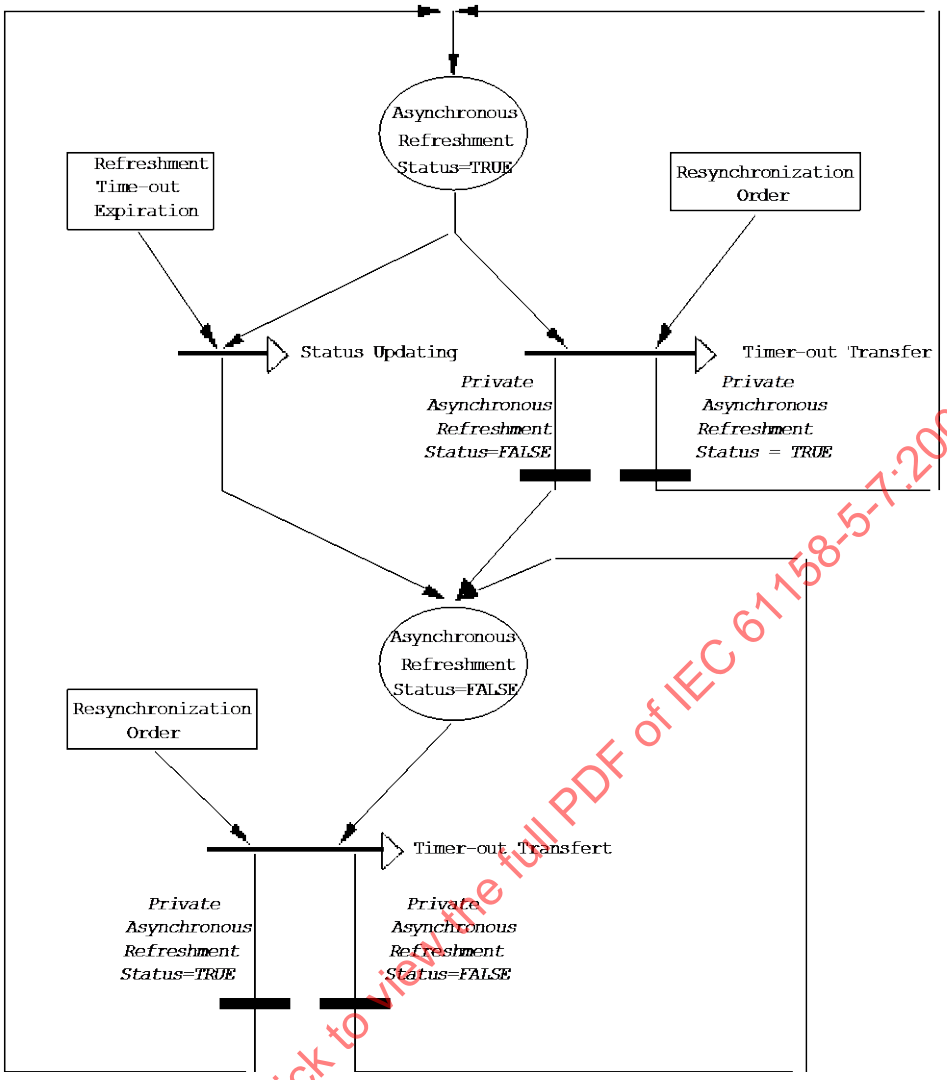
Tableau 19 – Événements et actions de mécanisme privé de rafraîchissement asynchrone

État initial:	**Statut de rafraîchissement asynchrone privé = *FALSE***
	État du délai d'attente de rafraîchissement privé = *IDLE*
Demandes:	**Production de variable:**
	Invocation par l'utilisateur des services A_WRITELOC, A_WRITEFAR ou A_WRITE
	Expiration du délai d'attente de rafraîchissement privé:
	Basculement de la temporisation privée associée au statut de rafraîchissement asynchrone privé de l'état *RUNNING* à l'état *IDLE*.
Action:	**Paramètrage du délai d'attente de rafraîchissement privé**
	Basculement de la temporisation privée associée au statut asynchrone privé de l'état *IDLE* à l'état *RUNNING*, avec la valeur "preset" (préréglage) égale à la valeur de l'attribut Production *period*.

Mécanisme public:

La temporisation publique est prédéfinie avec la valeur actuelle de la temporisation privée et définie sur l'ordre de resynchronisation associé à ce statut.

Le mécanisme public d'élaboration du statut de rafraîchissement asynchrone d'une variable resynchronisée est représenté par le diagramme d'évaluation (voir Figure 25 et Tableau 20).



Légende

Anglais	Français
Asynchronous Refreshment Status=TRUE	Statut de rafraîchissement asynchrone =VRAI
Resynchronization Order	Ordre/commande de resynchronisation
Refreshment Time-out Expiration	Expiration de temporisation de rafraîchissement
Timer-out Transfer	Transfert temporisateur
Private Asynchronous Refreshment Status=FALSE	Statut de rafraîchissement asynchrone privé=FAUX
Status Updating	Mise à jour de statut
Asynchronous Refreshment Status=FALSE	Statut de rafraîchissement asynchrone =FAUX
Private Asynchronous Refreshment Status = TRUE	Statut de rafraîchissement asynchrone privé = VRAI

Figure 25 – Diagramme d'évaluation de mécanisme public de rafraîchissement asynchrone

Tableau 20 – Événements et actions de mécanisme public de rafraîchissement asynchrone

État initial: **Statut de rafraîchissement asynchrone = *FALSE***

État du délai d'attente de rafraîchissement = *IDLE*

Demande: **Expiration du délai d'attente de rafraîchissement:**

Évolution de la temporisation publique associée au statut de rafraîchissement asynchrone de l'état *RUNNING* à l'état *IDLE*.

Ordre de Resynchronisation

Occurrence de l'ordre de non-synchronisation associé au mécanisme de resynchronisation publique.

Action: **Transfert des temporisations**

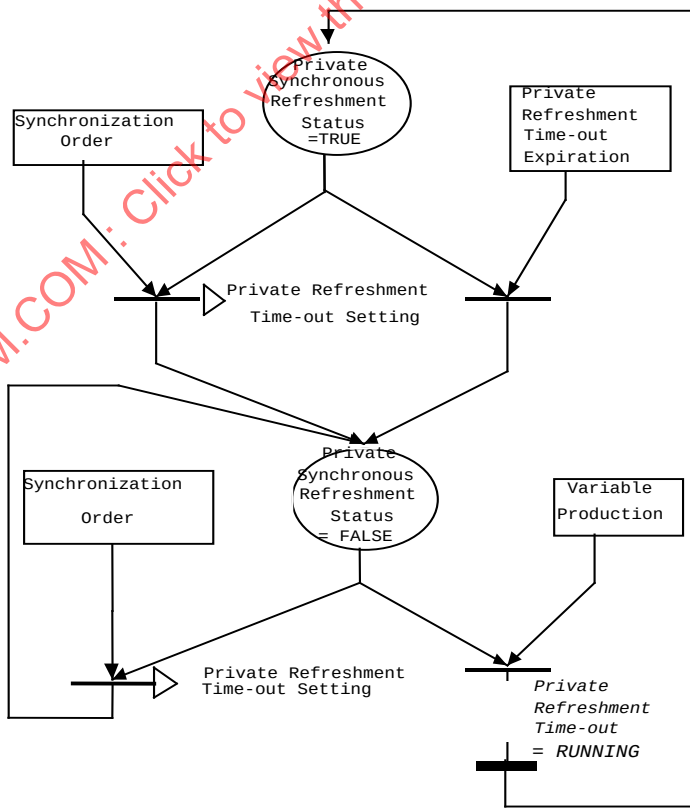
Transfert dans la valeur d'attribut *State* et la *valeur d'attribut Current* de la temporisation privée du statut de rafraîchissement asynchrone privé, des attributs *State* et *Preset* appartenant à la temporisation publique associée au statut de rafraîchissement asynchrone.

6.2.1.3.6.2.2.5 Mécanisme associé au rafraîchissement synchrone

Le mécanisme qui élabore le statut rafraîchissement synchrone pour une variable resynchronisée peut être décrit avec deux diagrammes d'évaluation, l'un décrit le statut de rafraîchissement synchrone privé et l'autre le statut de rafraîchissement synchrone associé à la valeur publique de la variable.

Mécanisme privé:

La temporisation privée associée au rafraîchissement est prédéfinie avec la période de production de la variable et définie au moment de la réception d'un nouvel ordre de synchronisation associé à ce statut (voir Figure 26 et Tableau 21).



Légende

Anglais	Français
Synchronization Order	Ordre/commande de synchronisation

Private Refreshment Time-out Expiration	Expiration de temporisation de rafraîchissement privé
Private synchronous Refreshment Status=FALSE	Statut de rafraîchissement synchrone privé=FAUX
Private Synchronous Refreshment Status = TRUE	Statut de rafraîchissement synchrone privé = VRAI
Private Refreshment Time-out Setting	Réglage de temporisation de rafraîchissement privé
Private Refreshment Time-out = RUNNING	Temporisation de rafraîchissement privé = EN MARCHE
Variable production	Production de variable

Figure 26 – Diagramme d'évaluation de mécanisme privé de rafraîchissement synchrone

Tableau 21– Événements et actions de mécanisme privé de rafraîchissement synchrone

État initial: **Statut de rafraîchissement synchrone privé = FALSE**

État du délai d'attente de rafraîchissement privé = IDLE

Demandes: **Production de variable:**

Invocation par l'utilisateur des services A_WRITELOC, A_WRITEFAR ou A_WRITE

Expiration du délai d'attente de rafraîchissement privé:

Basculement de la temporisation privée associée au statut synchrone privé de l'état RUNNING à l'état IDLE.

Ordre de synchronisation:

Occurrence d'un ordre de synchronisation associé au statut de rafraîchissement synchrone public et privé.

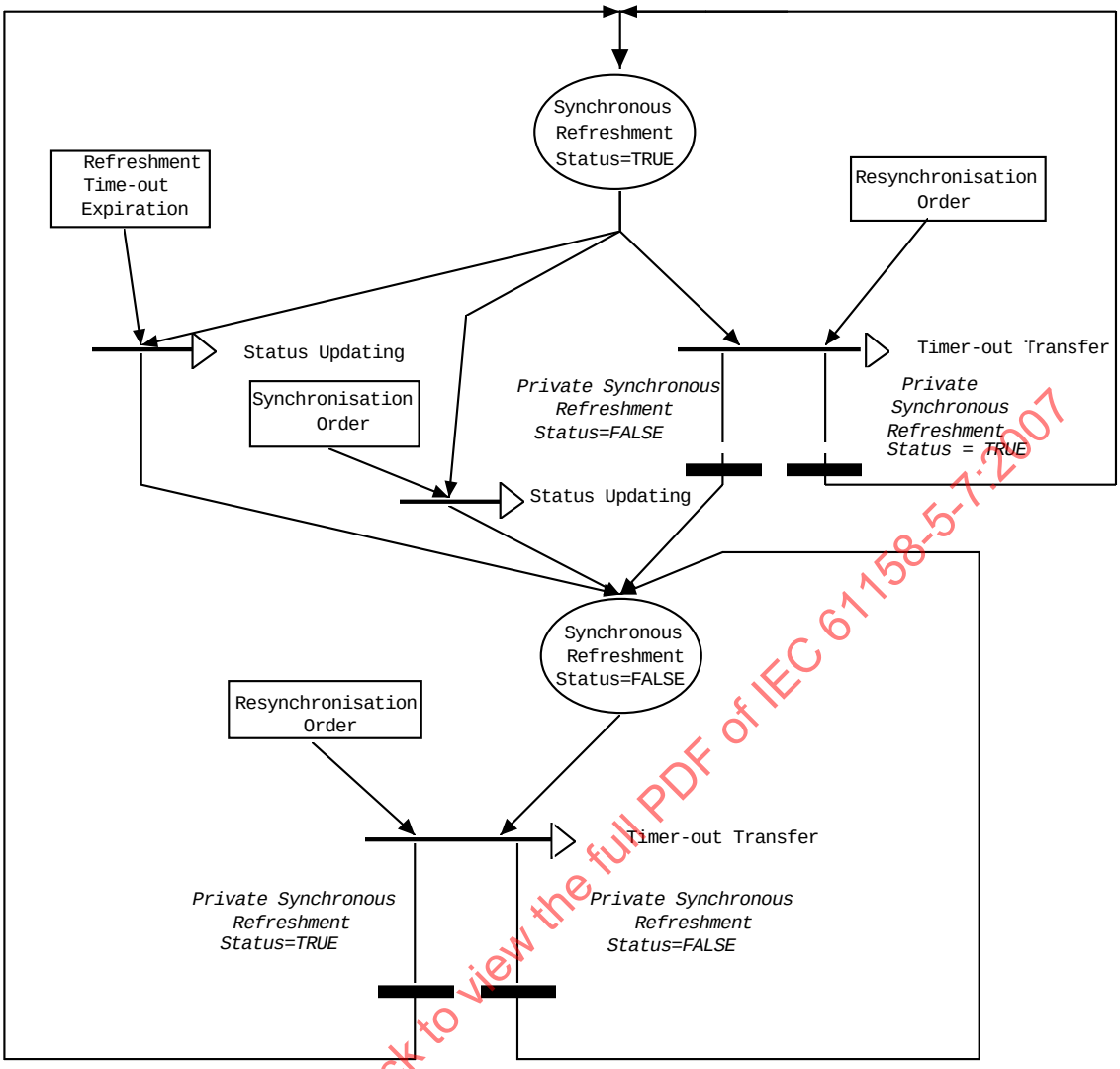
Action: **Paramétrage du délai d'attente de rafraîchissement privé**

Basculement de la temporisation privée associée au statut asynchrone privé de l'état *OUT* à l'état *SET*, avec la valeur "preset" (préréglage) égale à la valeur de l'attribut Production *period*.

Mécanisme public:

La temporisation associée au mécanisme public est prédéfinie avec la valeur actuelle de la temporisation privée et définie sur l'occurrence de l'ordre de resynchronisation.

Le mécanisme public d'élaboration du statut de rafraîchissement synchrone d'une variable resynchronisée est représenté par le diagramme d'évaluation (voir Figure 27 et Tableau 22).



Légende

Anglais	Français
Synchronous Refreshment Status=TRUE	Statut de rafraîchissement synchrone =VRAI
Resynchronization Order	Ordre/commande de resynchronisation
Refreshment Time-out Expiration	Expiration de temporisation de rafraîchissement
Timer-out Transfer	Transfert temporisateur
Status Updating	Mise à jour de statut
Synchronous Refreshment Status=FALSE	Statut de rafraîchissement synchrone =FAUX
Private synchronous Refreshment Status = TRUE	Statut de rafraîchissement asynchrone privé = VRAI
Synchronization Order	Ordre/commande de synchronisation
Private synchronous Refreshment Status = FALSE	Statut de rafraîchissement asynchrone privé = FAUX

Figure 27 – Diagramme d'évaluation de mécanisme public de rafraîchissement synchrone

Tableau 22 – Événements et actions de mécanisme public de rafraîchissement synchrone

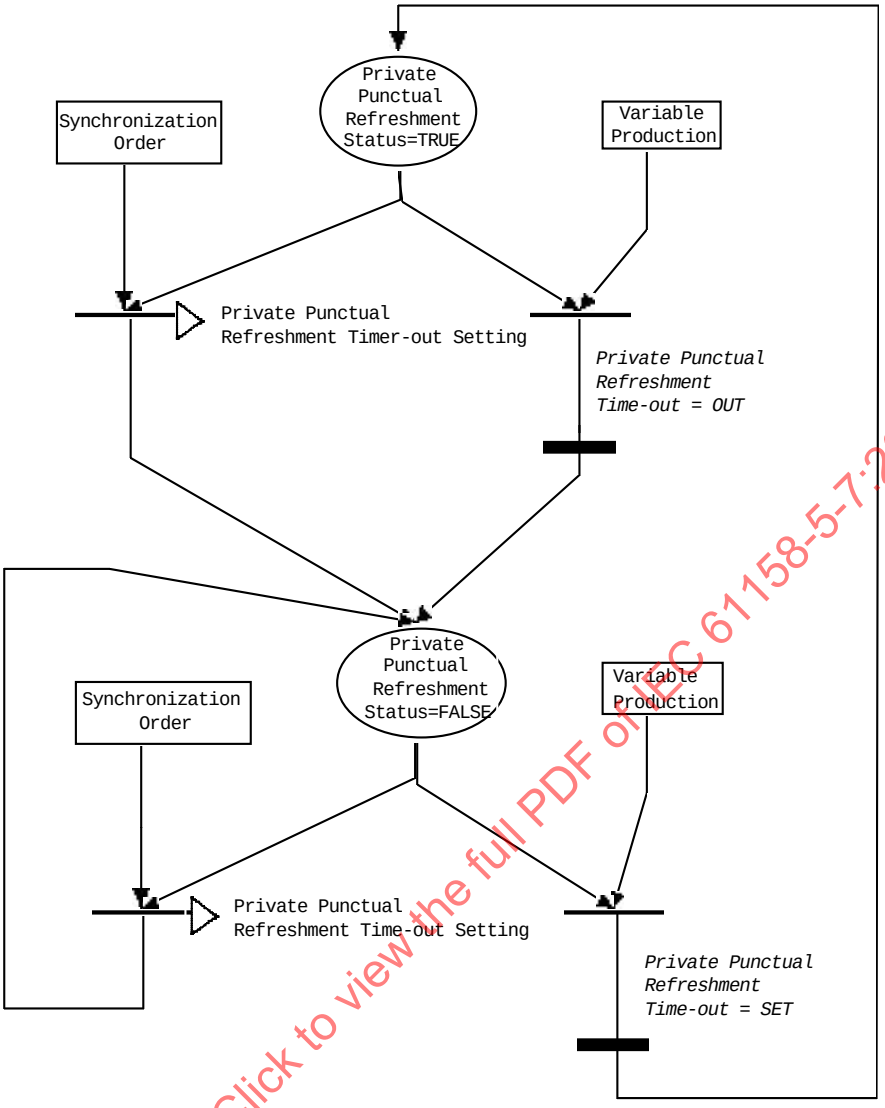
État initial:	**Statut de rafraîchissement synchrone = FALSE**
	État du délai d'attente de rafraîchissement = IDLE
Demandes:	**Expiration du délai d'attente de rafraîchissement:**
	Basculement de la temporisation publique associée au statut de rafraîchissement synchrone de l'état RUNNING à l'état IDLE.
	Ordre de Resynchronisation:
	Occurrence de l'ordre de synchronisation associé au mécanisme de resynchronisation publique.
	Ordre de Synchronisation:
	Occurrence de l'ordre de synchronisation associé au statut de rafraîchissement et à état de rafraîchissement synchrone privé.
Action:	**Transfert des temporisations:**
	Transfert dans la valeur d'attribut *State* et la valeur d'attribut *Current* de la temporisation privée du statut de rafraîchissement asynchrone privé, des attributs State et Preset appartenant à la temporisation publique associée au statut de rafraîchissement.

6.2.1.3.6.2.2.6 Mécanisme associé au rafraîchissement ponctuel

Le mécanisme qui élabore le rafraîchissement ponctuel pour une variable resynchronisée peut être décrit avec deux diagrammes d'évaluation, l'un décrit le statut de rafraîchissement ponctuel privé associé à la valeur privée de la variable, et l'autre le statut de rafraîchissement ponctuel associé à la valeur publique de la variable.

Mécanisme privé:

La temporisation privée associée au rafraîchissement ponctuel est prédéfinie avec l'intervalle de temps de production de la variable et définie à l'occurrence de l'ordre de synchronisation associé à ce statut (voir Figure 28 et Tableau 23).



Légende

Anglais	Français
Private punctual Refreshment Status = FALSE	Statut de rafraîchissement ponctuel privé = FAUX
Synchronization Order	Ordre/commande de synchronisation
Private punctual Refreshment Status = TRUE	Statut de rafraîchissement ponctuel privé = VRAI
Variable Production	Production de variable
Private punctual Refreshment Timer-out setting	Réglage de temporisateur de rafraîchissement ponctuel privé
Private punctual Refreshment Time-out =OUT	Temporisation de rafraîchissement ponctuel privé = hors limite
Private punctual Refreshment Time-out =SET	temporisation de rafraîchissement ponctuel privé = mise
Private punctual Refreshment Time-out setting	Réglage de temporisation de rafraîchissement ponctuel privé

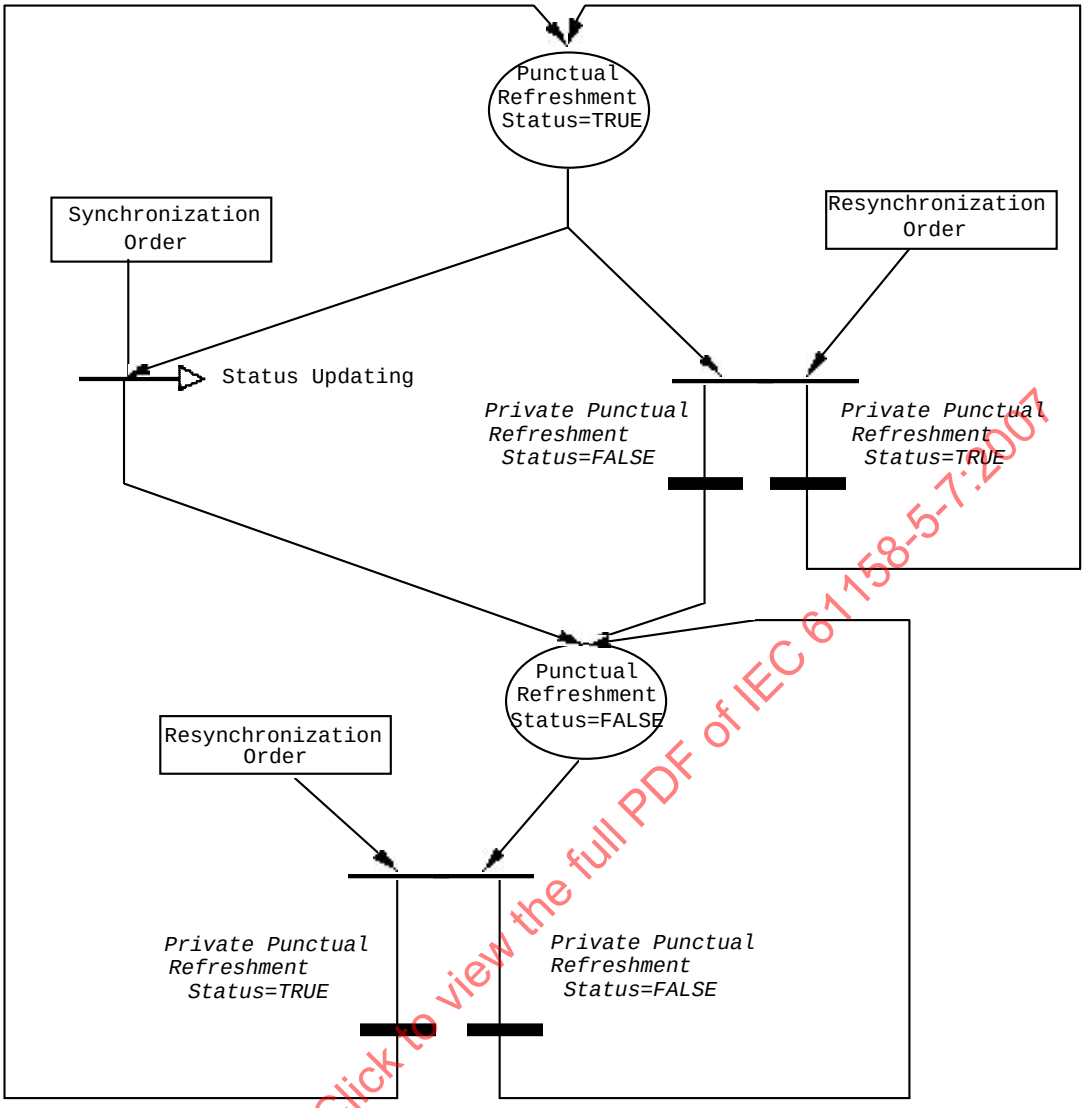
Figure 28 – Diagramme d'évaluation de mécanisme privé de rafraîchissement ponctuel

Tableau 23 – Événements et actions de mécanisme privé de rafraîchissement ponctuel

État initial:	**Statut de rafraîchissement ponctuel privé = FALSE**
	État du délai d'attente de rafraîchissement ponctuel privé = IDLE
Demandes	**Production de variable:**
	Invocation par l'utilisateur des services A_WRITELOC, A_WRITEFAR ou A_WRITE
	Ordre de Synchronisation:
	Occurrence de l'ordre de synchronisation associé au statut de rafraîchissement ponctuel et au rafraîchissement ponctuel privé.
Action:	**Paramétrage du délai d'attente de rafraîchissement ponctuel privé**
	Basculement de la temporisation privée associée au statut de rafraîchissement ponctuel privé de l'état IDLE à l'état RUNNING, avec la valeur "preset" (préréglage) égale à la valeur de l'attribut *Production time slot*.

Mécanisme public:

Le mécanisme public qui élabore le statut de rafraîchissement ponctuel d'une variable resynchronisée est représenté par le diagramme d'évaluation suivant (voir Figure 29 et Tableau 24).



Légende

Anglais	Français
Private punctual Refreshment Status = FALSE	Statut de rafraîchissement ponctuel privé = FAUX
Synchronization Order	Ordre/commande de synchronisation
Punctual Refreshment Status = TRUE	Statut de rafraîchissement ponctuel = VRAI
Resynchronization Order	Ordre/commande de resynchronisation
Status updating	Mise à jour de statut
Private punctual Refreshment Status = TRUE	Statut de rafraîchissement ponctuel privé = VRAI
Punctual Refreshment Status = FALSE	Statut de rafraîchissement ponctuel = FAUX

Figure 29 – Diagramme d'évaluation de mécanisme public de rafraîchissement ponctuel

Tableau 24 – Événements et actions de mécanisme public de rafraîchissement ponctuel

État initial	**Statut de rafraîchissement ponctuel = FALSE**
Demandes :	**Ordre de Resynchronisation**
	Occurrence de l'ordre de synchronisation associé au mécanisme de resynchronisation publique.
	Ordre de Synchronisation:
	Occurrence de l'ordre de synchronisation associé au statut de rafraîchissement ponctuel et au statut de rafraîchissement ponctuel privé.

L'élaboration du statut de rafraîchissement synchrone et du statut de rafraîchissement ponctuel nécessite des précautions particulières en ce qui concerne le choix respectif des variables de synchronisation associées à un statut synchrone et celles associées à la resynchronisation.

NOTE Le choix spécifique consistant à utiliser la même variable de synchronisation donne lieu au statut qui sera FALSE chaque fois que l'utilisateur exerce une écriture.

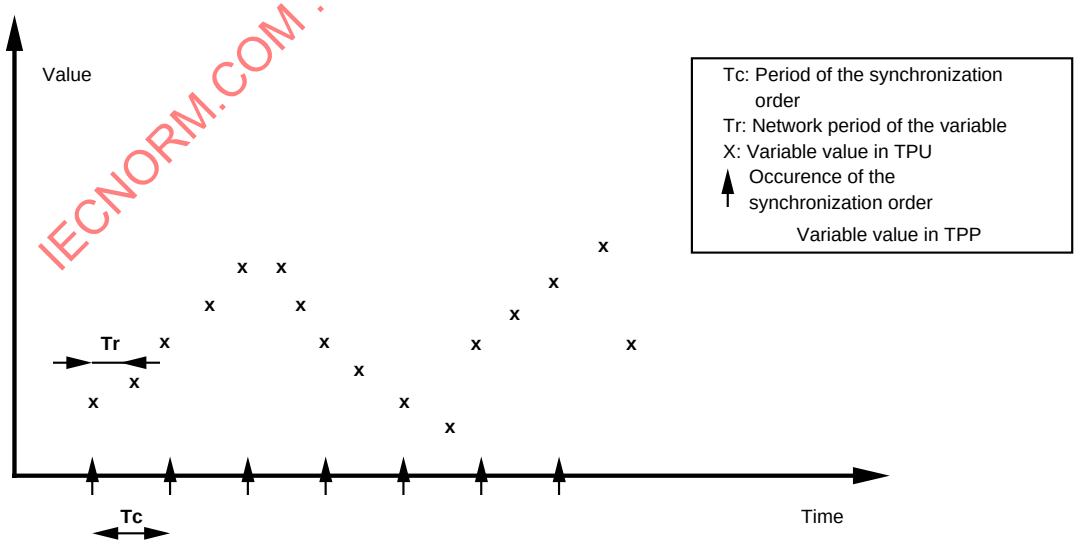
Le choix de deux variables de synchronisation distinctes, dont l'émission sur le bus est séparée par un intervalle de temps inférieur aux périodes de production ou aux intervalles de temps de production, peut conduire au statut FALSE si le temps d'arrivée de l'ordre de resynchronisation précède l'instant de l'écriture de l'utilisateur.

6.2.1.3.6.2.3 Resynchronisation d'une variable consommée

6.2.1.3.6.2.3.1 Principe

Si le réseau fournit une valeur de variable dans le tampon public (TPU) avec une période d'émission Tr, et en même temps un AP consomme cette valeur dans le tampon privé (TPP), alors le transfert de la valeur de variable du tampon public au tampon privé est effectué lors de la réception d'un ordre de resynchronisation avec une période Tc.

Le mécanisme de resynchronisation pour le consommateur contient dans le tampon privé la même valeur de la variable consommée durant l'intervalle de temps entre deux occurrences du même ordre de resynchronisation. La Figure 30 montre ces principes.



Légende

Anglais	Français
Value	Valeur

Time	Temps
Tc: Period of the synchronization order	Tc: Période d'ordre/commande de synchronisation
Tr: Network period of the variable	Tp: Période réseau de la variable
X: Variable value in TPU	X: Valeur de variable dans le TPU
Occurence of a synchronization order	Occurrence d'un ordre/d'une commande de synchronisation
Variable value in TPP	Valeur de variable dans le TPP

Figure 30 – Principes pour la resynchronisation d'une variable consommée

NOTE Le mécanisme de resynchronisation d'une variable consommée variable est utile seulement si $T_r << T_c$

Le service de resynchronisation implique l'utilisation de ressources supplémentaires telles que des tampons supplémentaires pour les valeurs privées associées aux valeurs publiques des variables resynchronisées.

Avec chaque tampon public d'une variable resynchronisée, une temporisation de l'élaboration du statut de promptitude et une temporisation de l'élaboration de la promptitude ponctuelle sont éventuellement associées.

Le service de resynchronisation nécessite l'utilisation d'une temporisation privée associée au tampon privé afin de permettre au statut d'émission de préserver sa sémantique lorsque les variables sont resynchronisées.

L'élaboration du statut de validité de l'émission pour une variable resynchronisée au niveau de l'entité consommatrice implique l'utilisation de deux mécanismes associés respectivement aux valeurs publiques et aux valeurs privées de la variable resynchronisée.

6.2.1.3.6.2.3.2 Spécification de la resynchronisation dans la classe de consommation

Cette classe décrit les ressources supplémentaires utilisées pour resynchroniser une variable consommée à un niveau de l'entité d'application.

La classe utilisée pour décrire une variable resynchronisée au sein d'une entité consommatrice hérite des attributs de la classe *Resynchronization in consumption*.

Class: RESYNCHRONIZATION IN CONSUMPTION

Attribute	:	Private value
Constraint	:	Promptness elaborated = TRUE
Attribute	:	Private Promptness status
Attribute	:	*Inherit from* Time-out
Constraint	:	Punctual Promptness elaborated = TRUE
Attribute	:	Private Punctual Promptness status
Attribute	:	*Reference* Synchronization variable

Private value:

Le principe de resynchronisation est basé sur des données qui sont à double mémoire tampon. Il est nécessaire d'ajouter une valeur privée à la valeur publique de la variable. La valeur privée peut maintenant être lue de façon asynchrone.

Promptness elaborated:

Cet attribut, tant dans le contexte privé que dans le contexte public conditionne l'existence d'un statut de promptitude associé à une variable.

Lorsque cet attribut a la valeur TRUE, une variable resynchronisée dispose du statut de promptitude et du statut de promptitude privée.

L'attribut conditionnel est déclaré dans la classe *Consumed variable*.

Private promptness status

Ce statut privé correspond aux informations internes à partir de l'entité consommatrice de la variable resynchronisée, à partir de laquelle est construite la valeur du statut de promptitude.

La caractéristique de ce statut est SYNCHRONOUS ou ASYNCHRONOUS selon la déclaration globale associée à la variable resynchronisée en consommation qui est effectuée dans la classe *Promptness*.

Dans le cas d'un statut synchrone privé, la variable de synchronisation utilisée par le mécanisme privé associé à ce statut est déclarée comme étant une variable globale dans la classe *Promptness*.

Inherit from Time-out:

Cet ensemble de ressources permet à l'utilisateur de maintenir la sémantique du statut de promptitude associé à une variable resynchronisée.

En fait, l'utilisation d'un statut de promptitude synchrone ou asynchrone nécessite l'utilisation d'un délai d'attente supplémentaire pour préserver la sémantique entière de ce statut.

Punctual Promptness elaborated

Cet attribut traite généralement dans le contexte privé et dans le contexte public l'existence d'un statut de promptitude ponctuelle associé à une variable.

Lorsque cet attribut a la valeur TRUE, une variable resynchronisée dispose du statut de promptitude ponctuelle et du statut de promptitude ponctuelle privée.

L'attribut conditionnel est déclaré dans la classe *Consumed variable*.

Private Punctual Promptness status

Ce statut privé correspond aux informations internes à partir de l'entité consommatrice de la variable resynchronisée, à partir de laquelle est construite la valeur du statut de promptitude ponctuelle.

Dans le cas d'un statut ponctuel privé, la variable de synchronisation utilisée par le mécanisme privé associé à ce statut est déclarée comme étant une variable globale dans la classe *Punctual Promptness*.

***Reference* (Synchronization) variable:**

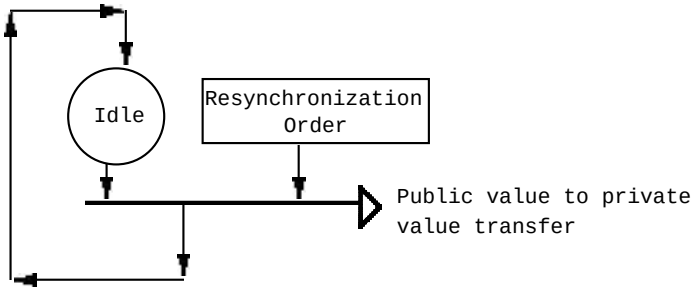
La référence à une variable de synchronisation indique l'ordre de resynchronisation associé à une variable resynchronisée.

Cet ordre de resynchronisation conditionne le transfert des valeurs de variables, les statuts d'émission et les valeurs courantes des temporisations du domaine privé au domaine public. Cette référence indique le *A_Name* de l'objet *Consumed Variable Local Image*.

6.2.1.3.6.2.3.3 Mécanisme associé à la valeur consommée

Le mécanisme de resynchronisation en consommation de la valeur de la variable locale est le transfert du tampon privé vers le tampon public, sur ordre de resynchronisation.

Le diagramme d'évaluation de cet élément de service qui gère les actions de resynchronisation à l'intérieur d'une entité d'application consommatrice pour une variable resynchronisée est représenté à la Figure 31.



Légende

Anglais	Français
Idle	Repos
Resynchronization Order	Ordre/commande de resynchronisation
Public value to Private value transfer	Transfert de valeur publique vers valeur privée

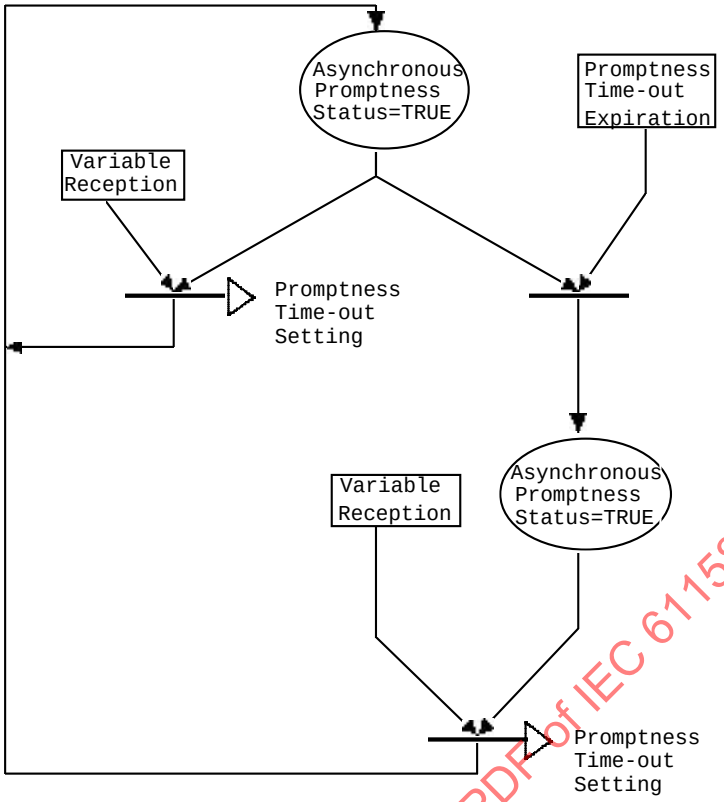
Figure 31 – Diagramme d'états du mécanisme de resynchronisation pour une variable consommée

6.2.1.3.6.2.3.4 Mécanisme associé à la promptitude asynchrone

Le mécanisme qui élabore le statut de promptitude asynchrone pour une variable resynchronisée peut être décrit avec deux diagrammes d'évaluation, l'un d'eux décrit la promptitude asynchrone privée et l'autre le statut de promptitude asynchrone associé à la valeur publique de la variable.

Mécanisme public:

La temporisation publique associée à la promptitude est prédéfinie avec la période de consommation de la variable et définie à l'instant de réception d'une nouvelle valeur de variable publique (voir Figure 32 et Tableau 25).



Légende

Anglais	Français
Variable Reception	Réception de variable
Promptness Time-out Expiration	Expiration de temporisation de promptitude
Promptness Time-out Setting	Réglage de temporisation de promptitude
Asynchronous promptness Status=TRUE	Statut de promptitude asynchrone =VRAI

Figure 32 – Diagramme d'évaluation de mécanisme public de promptitude asynchrone

Tableau 25 – Événements et actions de mécanisme public de promptitude asynchrone

État initial: **Statut de promptitude asynchrone = IDLE**

État du délai d'attente de promptitude = OUT

Demandes: **Réception de Variable:**

Mise à jour de la valeur de variable par le réseau

Expiration du délai d'attente de promptitude:

Basculement de la temporisation publique associée au statut public de promptitude asynchrone de l'état RUNNING à l'état IDLE.

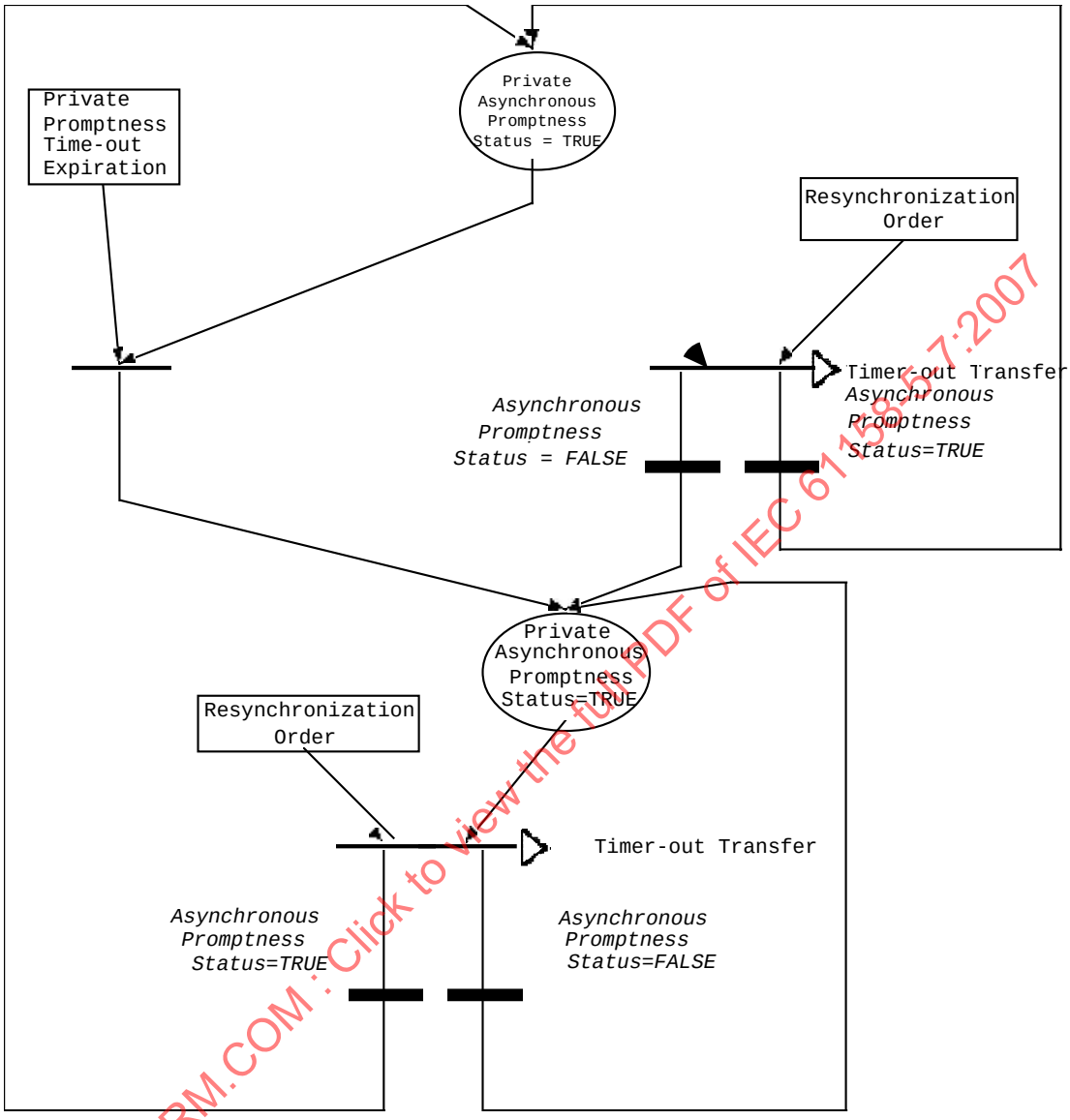
Actions: **Paramétrage du délai d'attente de promptitude:**

Basculement de la temporisation publique associée au statut asynchrone public de l'état IDLE à l'état RUNNING, avec la valeur "preset" (préréglage) égale à la valeur de l'attribut *Consumption period*.

Mécanisme privé:

La temporisation privée est prédéfinie avec la valeur actuelle de la temporisation publique et définie sur l'ordre de resynchronisation associé à ce statut.

Le mécanisme d'élaboration du statut de promptitude asynchrone privé d'une variable resynchronisée est représenté par le diagramme d'évaluation suivant (voir Figure 33 et Tableau 26).



Légende

Anglais	Français
Asynchronous promptness Status=TRUE	Statut de promptitude asynchrone =VRAI
Resynchronization Order	Ordre/commande de resynchronisation
Timer-out Transfer	Transfert temporisateur
Private Asynchronous Promptness Status=TRUE	Statut de promptitude asynchrone privée=VRAI
Private Promptness Time-out Expiration	Expiration de temporisation de promptitude privée
Asynchronous promptness Status=FALSE	Statut de promptitude asynchrone =FAUX

Figure 33 – Diagramme d'évaluation de mécanisme privé de promptitude asynchrone

Tableau 26 – Événements et actions de mécanisme privé de promptitude asynchrone

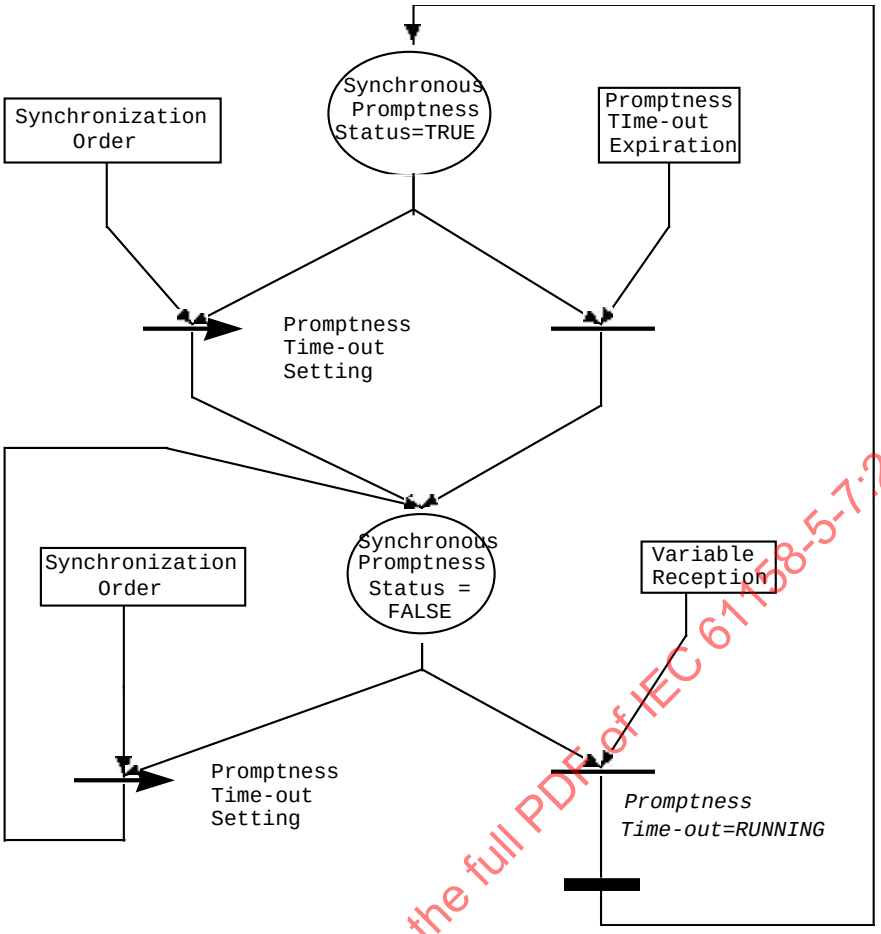
État initial:	**Statut de promptitude asynchrone privée = FALSE**
	État du délai d'attente de promptitude privée = IDLE
Demandes:	**Expiration du délai d'attente de promptitude privée:**
	Basculement de la temporisation privée associée au statut de promptitude asynchrone privé de l'état RUNNING à l'état IDLE.
	Ordre de Resynchronisation
	Occurrence d'un ordre de synchronisation associé au mécanisme de resynchronisation.
Action:	**Transfert des temporisations**
	Transfert dans la valeur d'attribut State et la valeur d'attribut Current de la temporisation publique du statut public de promptitude asynchrone, des attributs State et Preset appartenant à la temporisation privée associée au statut de promptitude asynchrone.

6.2.1.3.6.2.3.5 Mécanisme associé à la promptitude synchrone

Le mécanisme qui élabore le statut de promptitude synchrone pour une variable resynchronisée peut être décrit avec deux diagrammes d'évaluation, l'un d'eux décrit le statut de promptitude synchrone privé et l'autre le statut de promptitude synchrone associé à la valeur publique de la variable.

Mécanisme public:

La temporisation publique associée à la promptitude est prédéfinie avec la période de consommation de la variable et définie à l'instant de réception d'un nouvel ordre de synchronisation (voir Figure 34 et Tableau 27).



Légende

Anglais	Français
Synchronization Order	Ordre de synchronisation
Promptness Time-out setting	Réglage de la temporisation de promptitude
Synchronus promptness Status = FALSE	Statut de promptitude synchrone = FAUX
Variable Reception	Réception de variable
Synchronus promptness Status = TRUE	Statut de promptitude synchrone =VRAI
Promptness Time-out = RUNNING	Temporisation de promptitude = EN MARCHE
Promptness time-out expiration	Expiration de temporisation de promptitude

Figure 34 – Diagramme d'évaluation de mécanisme public de promptitude synchrone

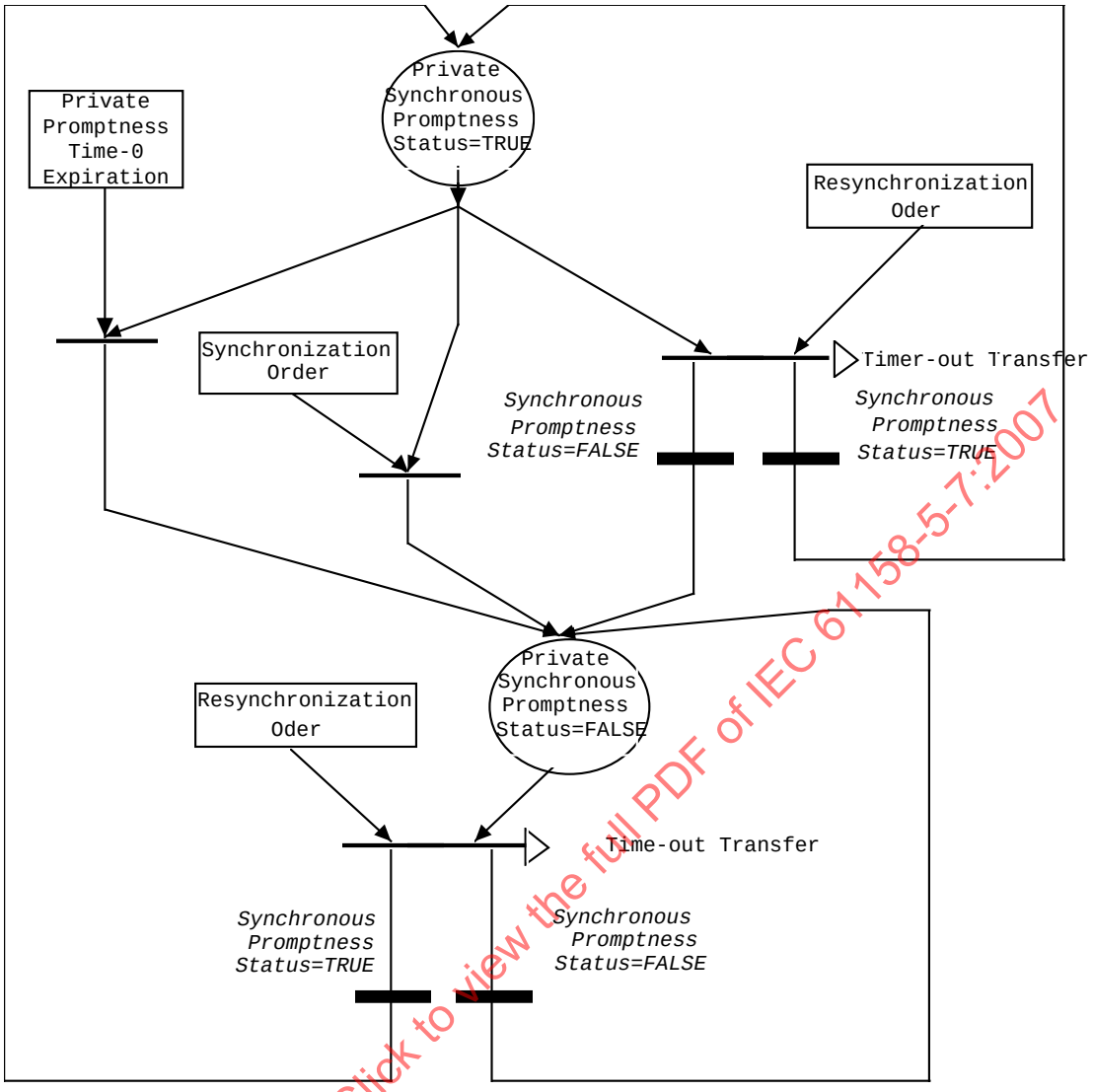
Tableau 27 – Événements et actions de mécanisme public de promptitude synchrone

État initial:	**Statut de promptitude synchrone = FALSE**
	État du délai d'attente de promptitude = IDLE
Demandes:	**Réception de Variable:**
	Mise à jour de la valeur de variable par le réseau
	Ordre de Synchronisation:
	Occurrence d'un ordre de synchronisation associé au statut de promptitude et au statut de promptitude synchrone privée.
	Expiration du délai d'attente de promptitude:
	Basculement de la temporisation publique associée au statut de promptitude synchrone de l'état RUNNING à l'état IDLE.
Action:	**Paramétrage du délai d'attente de promptitude:**
	Basculement de la temporisation publique associée au statut synchrone public de l'état IDLE à l'état RUNNING, avec la valeur "preset" (préréglage) égale à la valeur de l'attribut Consumption period.

Mécanisme privé:

La temporisation privée est prédéfinie avec la valeur actuelle de la temporisation publique et définie sur l'Occurrence de l'ordre de resynchronisation.

Le mécanisme privé d'élaboration du statut de promptitude synchrone privé d'une variable resynchronisée est représenté par le diagramme d'évaluation suivant (voir Figure 35 et Tableau 28).



Légende

Anglais	Français
Private Synchronous Promptness Status=TRUE	Statut de promptitude synchrone privée =VRAI
Private Synchronous Promptness Status=FALSE	Statut de promptitude synchrone privée =FAUX
Private Promptness Time-out Expiration	Expiration de temporisation de promptitude privée
Resynchronization Order	Ordre/commande de resynchronisation
Synchronization Order	Ordre/commande de synchronisation
Timer-out Transfer	Transfert temporisateur
Synchronous Promptness Status=FALSE	Statut de promptitude synchrone =FAUX
Synchronous Promptness Status=TRUE	Statut de promptitude synchrone =VRAI
Time-out Transfer	Transfert temporisation

Figure 35 – Diagramme d'évaluation de mécanisme privé de promptitude synchrone

Tableau 28 – Événements et actions de mécanisme privé de promptitude synchrone

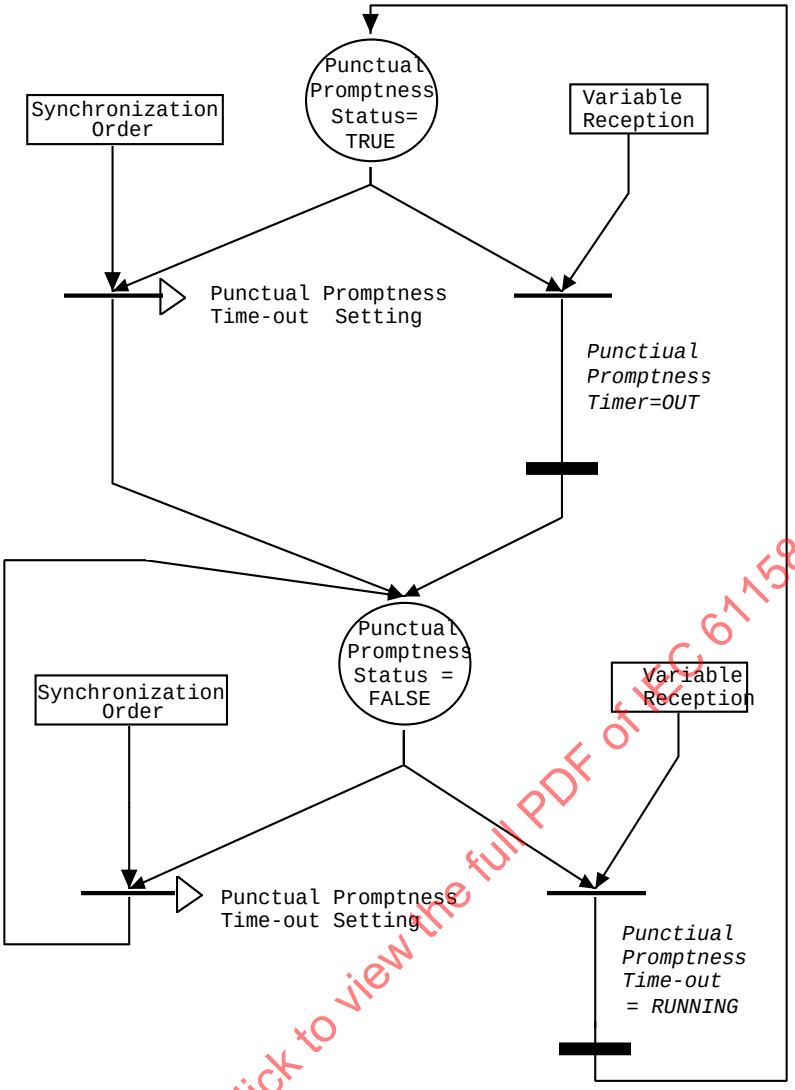
État initial:	**Statut de promptitude synchrone privée = FALSE**
	État du délai d'attente de promptitude privée = IDLE.
Demandes:	**Expiration du délai d'attente de promptitude privée:**
	Basculement de la temporisation privée associée à la promptitude synchrone de l'état RUNNING à l'état IDLE.
	Ordre de Resynchronisation
	Occurrence d'un ordre de synchronisation associé au mécanisme de resynchronisation
	Ordre de Synchronisation
	Occurrence d'un ordre de synchronisation associé au statut de promptitude et à la promptitude synchrone privée
Action:	**Transfert des temporisations:**
	Affectation de l'état de la valeur des attributs et la valeur courante du statut public de promptitude synchrone, aux attributs State et Preset du temporisateur privé associé au statut de promptitude synchrone.

6.2.1.3.6.2.3.6 Mécanisme associé à la promptitude ponctuelle

Le mécanisme qui élabore la promptitude ponctuelle pour une variable resynchronisée peut être décrit avec deux diagrammes d'évaluation, l'un décrivait le statut de promptitude ponctuelle privé associé à la valeur privée et l'autre le statut de promptitude ponctuelle associé à la valeur publique de la variable.

Mécanisme public:

La temporisation publique associée à la promptitude ponctuelle est prédéfinie avec l'intervalle de temps de consommation de la variable et définie à l'occurrence de l'ordre de synchronisation associé à ce statut (voir Figure 36 et Tableau 29).



Légende

Anglais	Français
Punctual Promptness Status=TRUE	Statut de promptitude ponctuelle =VRAI
Synchronization Order	Ordre/commande de synchronisation
Variable reception	Réception de variable
Punctual Promptness Time-out setting	Réglage de temporisation de promptitude ponctuelle
Punctual Promptness Timer = OUT	Temporisateur de promptitude ponctuelle= HORS limite
Punctual Promptness Status=FALSE	Statut de promptitude ponctuelle =FAUX
Punctual Promptness Time-out = RUNNING	Temporisation de promptitude ponctuelle = EN MARCHE

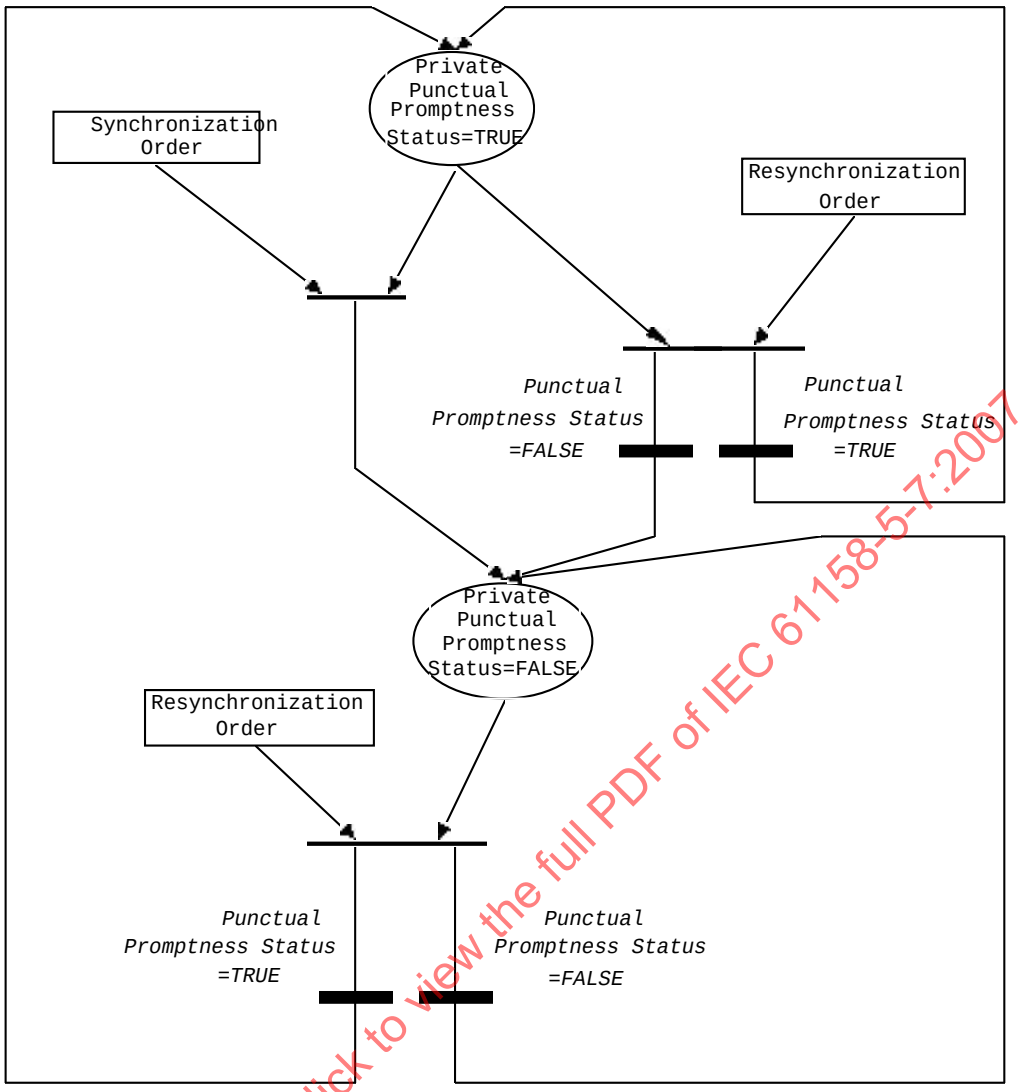
Figure 36 – Diagramme d'évaluation de mécanisme public de promptitude ponctuelle

Tableau 29 – Événements et actions de mécanisme public de promptitude ponctuelle

État initial	**Statut de promptitude ponctuelle = FALSE**
	État du délai d'attente de promptitude = OUT
Demandes :	**Réception de Variable:**
	Mise à jour de la valeur de variable par le réseau.
	Ordre de Synchronisation:
	Occurrence d'un ordre de synchronisation associé au statut de promptitude ponctuelle et au statut de promptitude ponctuelle privée.
Action:	**Paramétrage du délai d'attente de promptitude ponctuelle**
	Basculement de la temporisation publique associée au statut public de promptitude ponctuelle de l'état IDLE à l'état RUNNING, avec la valeur "preset" (préréglage) égale à la valeur de l'attribut *Consumption time slot*.

Mécanisme privé:

Le mécanisme privé qui élabore le statut de promptitude ponctuelle d'une variable resynchronisée est représenté par le diagramme d'évaluation suivant (voir Figure 37 et Tableau 30).



Légende

Anglais	Français
Private Punctual Promptness Status=TRUE	Statut de promptitude ponctuelle privée =VRAI
Synchronization Order	Ordre/commande de synchronisation
Resynchronization Order	Ordre/commande de resynchronisation
Private Punctual Promptness Status=FALSE	Statut de promptitude ponctuelle privée =FAUX
Variable reception	Réception de variable
Punctual Promptness Status=TRUE	Statut de promptitude ponctuelle =VRAI
Punctual Promptness Timer = OUT	Temporisateur de promptitude ponctuelle= HORS LIMITE
Punctual Promptness Status=FALSE	Statut de promptitude ponctuelle =FAUX

Figure 37 – Diagramme d'évaluation de mécanisme privé de promptitude ponctuelle

Tableau 30 – Événements et actions de mécanisme privé de promptitude ponctuelle

État initial	**Statut de promptitude ponctuelle privée = FALSE**
Demandes	**Ordre de Resynchronisation:**
	Occurrence d'un ordre de non-synchronisation associé au mécanisme de resynchronisation.
	Ordre de Synchronisation:
	Occurrence de l'ordre de synchronisation associé au statut de promptitude ponctuelle et au statut de promptitude ponctuelle privée.

L'élaboration d'une promptitude synchrone et d'un statut de promptitude ponctuelle nécessite des précautions particulières pour des choix respectifs des variables de synchronisation associées à un statut synchrone et celles associées à la resynchronisation.

NOTE Le choix spécifique consistant à utiliser la même variable de synchronisation donne lieu au statut qui sera FALSE quel que soit le moment de la mise à jour du réseau.

Le choix de deux variables de synchronisation distinctes, dont l'émission sur le bus est séparée par un intervalle de temps inférieur à la période de consommation ou à l'intervalle de temps de consommation, peut conduire au statut FALSE si le temps d'arrivée de l'ordre de resynchronisation précède l'instant de mise à jour par le réseau.

6.2.1.4 Sous-ASE d'accès de listes de variables

6.2.1.4.1 Concepts

6.2.1.4.1.1 Définition de liste

La couche application prend en charge la définition et la gestion des listes de variables.

Une liste est un ensemble ordonné de variables de classe NORMAL. Une liste de variables ne peut pas inclure de variables des classes SYNCHRONIZATION ou DESCRIPTION.

Une liste de variables est définie globale et est instanciée au niveau de l'entité d'application. Elle se compose exclusivement de variables consommées.

Plusieurs listes peuvent être définies dans un système unique, mais il convient qu'elles soient toutes séparées deux par deux:

$$\forall_{i,j} \quad \mathbf{list}_i \cap \mathbf{list}_j = \varnothing$$

Cependant, plusieurs instances d'une même liste peuvent être définies, au sein d'un système, parce que plusieurs AP peuvent être consommateurs de la même liste.

Une liste de variables définie de cette manière ne peut être traitée que par un service de lecture approprié permettant un accès global à l'ensemble de la liste. Cependant, toutes les variables appartenant à une liste peuvent également être traitées individuellement par les services de lecture de variable.

NOTE Il n'y a pas de prise en charge du concept d'une liste récursive au niveau des services d'application fournis à l'utilisateur. (Par conséquent, il n'existe pas de liste de la liste.)

Les variables déclarées par l'utilisateur comme appartenant à une liste de variables ne peuvent pas être regroupées dans une structure, parce qu'il est nécessaire qu'elles disposent d'informations de validité individuelles de façon optimisée.

La plupart des listes de variables mises en œuvre dans l'environnement MPS sont composées de variables périodiques ayant la même période, toutefois les listes de variables non périodiques sont également envisagées.

6.2.1.4.1.2 Statut associé à une liste

Au moment de lecture d'une liste, l'utilisateur peut obtenir des informations supplémentaires relatives à la cohérence de variable de cette liste. Ces statuts sont élaborés ou pas en fonction de la configuration associée à la liste. Ils informent l'utilisateur concernant l'intégrité de toutes les variables qui composent la liste.

Ces statuts sont retournés par le service d'accès à la liste.

Dans l'environnement MPS, les différents statuts associés aux listes de variables informent l'utilisateur concernant:

- la TRANSMISSION CONSISTENCY (Cohérence d'émission) de la liste
- la PRODUCTION CONSISTENCY (Cohérence de production) de la liste
- la PUNCTUAL TRANSMISSION CONSISTENCY (Cohérence d'émission ponctuelle) de la liste
- la PUNCTUAL PRODUCTION CONSISTENCY (Cohérence de production ponctuelle) de la liste
- la SPATIAL CONSISTENCY (Cohérence spatiale) de la liste

Tous ces statuts associés à une liste par un utilisateur sont élaborés en utilisant:

- les informations locales à une AE consommatrice, et par rapport à la validité de l'émission des variables de la liste,
- les informations élaborées par les producteurs distants des variables de la liste, et par rapport à leur validité de production.

Les informations relatives à la validité de l'émission, impliquées dans l'élaboration du statut de cohérence pour chacune des variables de la liste sont

- la promptitude asynchrone ou synchrone,
- la promptitude ponctuelle

Les informations relatives à la validité de production, impliquées dans l'élaboration du statut de cohérence pour chacune des variables de la liste sont

- le rafraîchissement asynchrone ou synchrone,
- le rafraîchissement ponctuel.

NOTE les informations de validité de production élaborées au sein des entités productrices des variables de la liste sont transportées sur le réseau avec les valeurs des variables.

6.2.1.4.1.3 Éléments de contrôle et fiabilité

La fiabilité de l'émission des variables d'une liste peut être améliorée en utilisant un mécanisme de récupération; ce mécanisme effectue une demande de réémission pour un nombre de fois limité à une valeur maximale définie pour chacune des instances de cette liste.

L'évaluation du statut de cohérence spatiale est obtenue localement à partir d'une fonction purement logique sur la base des valeurs de variables de cohérence.

Le concept de la cohérence spatiale est basé sur:

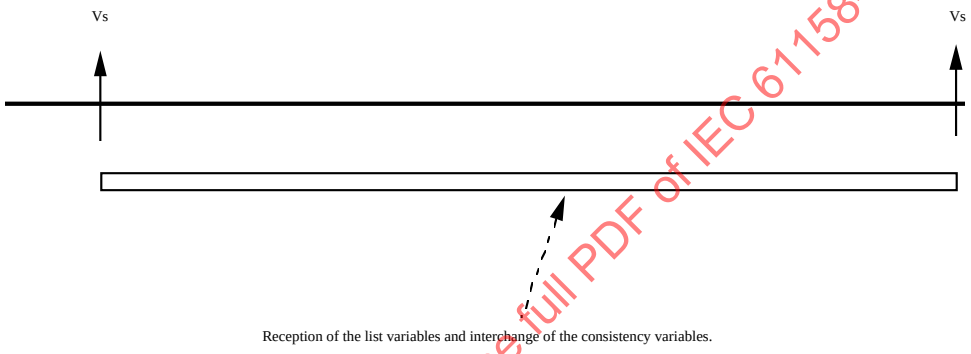
- L'existence d'une variable de cohérence produite par chaque abonné consommant une image locale de la liste, de même que l'existence d'un mécanisme d'échange de ces variables de cohérence entre les différents abonnés.

- L'existence d'un mécanisme local de gestion de valeur de la variable de cohérence produite.
- La définition d'une certaine action d'ordre séquentiel pour l'échange des variables de la liste, et les variables de cohérence.

Ordre séquentiel de l'échange de variable sur le réseau

Afin d'élaborer un statut de cohérence spatiale sur une liste de variables, il est nécessaire de définir un cycle correspondant à une émission de toutes les variables de la liste afin d'être en mesure de mettre en place une référence temporelle pour la gestion de variables de cohérence.

Ce cycle est délimité par un ordre de synchronisation Vs dont l'occurrence définit le début d'un nouveau cycle d'échange de variables. Toutes les variables de liste relatives à la liste nécessitent d'être reçues et toutes les variables de cohérence nécessitent d'être échangées entre deux occurrences consécutives de l'ordre de synchronisation Vs. Cela est montré à la Figure 38.

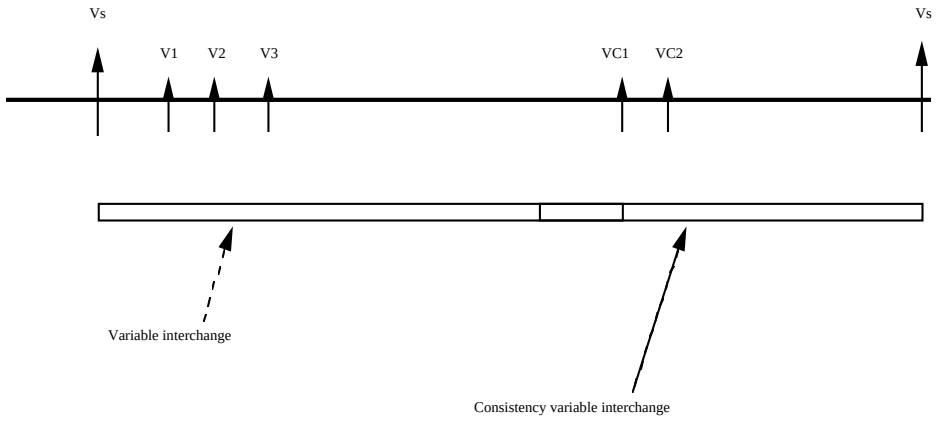


Légende

Anglais	Français
Reception of the list variables and interchange of the consistency variables	Réception des variables de la liste et échange des variables de cohérence

Figure 38 – Mécanisme d'échange des variables de liste de cohérence spatiale

En outre, une condition d'utilisation des listes exige que l'échange de variables de cohérence soit effectué après l'échange de toutes les variables de liste, et à l'intérieur du cycle courant (voir Figure 39).

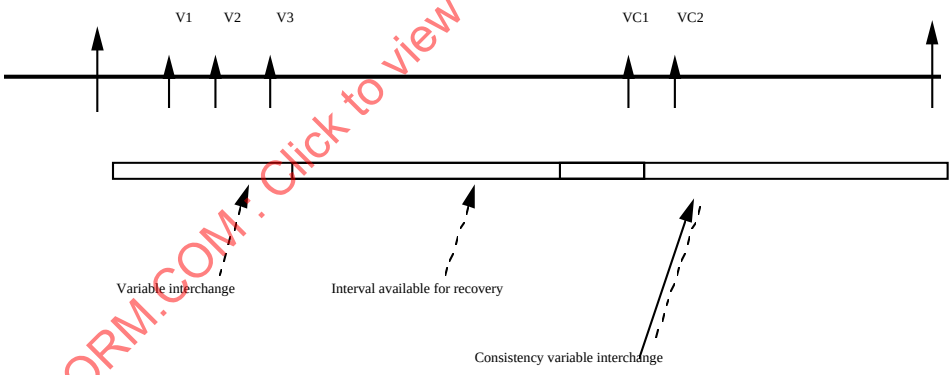


Légende

Anglais	Français
Variable interchange	Échange de variables
Consistency variable interchange	Échange de variables de cohérence

Figure 39 – Cohérence spatiale – Mécanisme d'échange de variables de cohérence

Lorsque le mécanisme d'élaboration du statut de cohérence spatiale est impliqué dans un mécanisme de récupération (voir Figure 40), la récupération éventuelle des variables concernées par les erreurs de transmission nécessite d'être effectuée après la réception théorique de toutes les variables de liste, et avant l'échange de variables de cohérence. Cela rend le mécanisme de récupération et le mécanisme de cohérence spatiale significatifs.

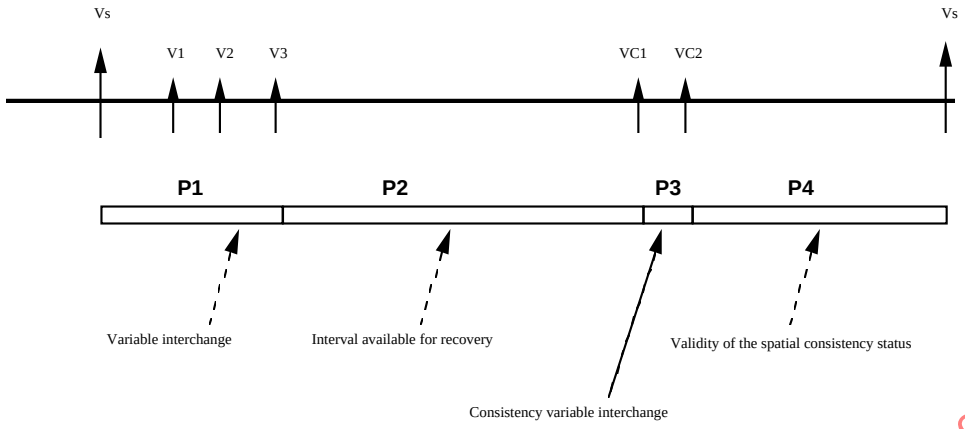


Légende

Anglais	Français
Variable interchange	Échange de variables
Interval available for recovery	Intervalle disponible pour la récupération
Consistency variable interchange	Échange de variables de cohérence

Figure 40 – Cohérence spatiale – Mécanisme de récupération de listes

Enfin, le statut de cohérence spatiale localement élaboré à l'intérieur d'une AE et mis à disposition de l'utilisateur n'est significatif que lorsque toutes les variables de cohérence ont été échangées. Cela conduit à la définition d'un intervalle de validité relatif au statut de cohérence spatiale (voir Figure 41).



Légende

Anglais	Français
Variable interchange	Échange de variables
Interval available for recovery	Intervalle disponible pour la récupération
Consistency variable interchange	Échange de variables de cohérence
Validity of the spatial consistency status	Validité du statut de cohérence spatiale

Figure 41 – Cohérence spatiale – validité du statut de cohérence spatiale

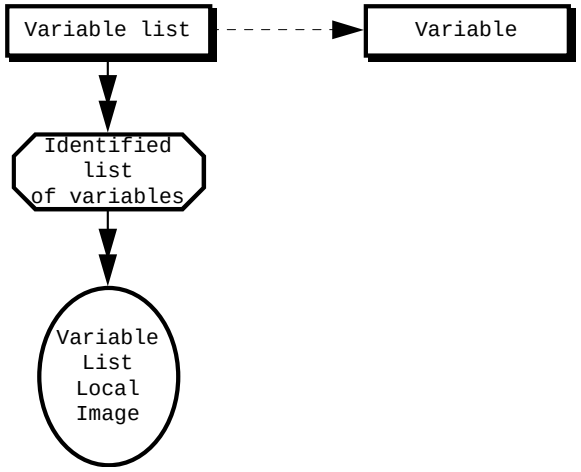
6.2.1.4.1.4 Modèle d'objets d'une liste de variables

Une liste de variables dans l'environnement MPS est modélisée en utilisant l'objet *Variable List Local Image* qui contient tous les attributs requis par une entité d'application consommant cette liste de variables.

Cet objet provient de l'instanciation de la classe *Identified Variable List* qui contient tous les attributs qui caractérisent une liste de variables.

Les attributs communs d'une liste de variables dans les différentes entités, décrits dans la classe *Identified variable List* proviennent de l'instanciation d'une classe générique *Variable List* qui décrit l'ensemble des attributs génériques d'une liste de variables.

En outre, il est signalé que la classe générique *Variable List* fait référence à la classe générique *Variable* afin de décrire les variables qui composent cette liste de variables. Tout cela est montré à la Figure 42.



Légende

Anglais	Français
Variable List	Liste de variables
Variable	Variable
Identified list of variables	Liste identifiée de variables
Variable list Local Image	Image locale de liste de variables

Figure 42 – Modèle d'objets d'une liste de variables

6.2.1.4.2 Spécification de la classe Variable list

6.2.1.4.2.1 Spécification de la classe générique *Variable List*

6.2.1.4.2.1.1 Modèle de la classe générique Variable list

FAL ASE: **MPS ASE**
CLASS: Variable list
CLASS ID: non utilisé
PARENT CLASS: non utilisée
Generic Class: VARIABLE LIST
CLASS ATTRIBUTES:
Key-Attribute : A_Name
Attribute : List of Reference Consumed Variable
Attribute : Spatial consistency required [TRUE,FALSE]
Attribute : Recovery required [TRUE,FALSE]
Constraint : Spatial consistency required = TRUE ***OR*** Recovery required = TRUE
Attribute : *Reference* (Synchronization) Variable
Constraint : Spatial consistency required = TRUE
Attribute : Inconsistency detection [TRANSITORY,PERMANENT,UNDEFINED]
Attribut: *List of*
Attribute: *Reference* (Consistency) Variable
Constraint : Recovery required = TRUE
Attribute : Recovery period
Instance Attributes:
Attribute : Transmission consistency required [TRUE,FALSE]
Constraint : Transmission consistency required = TRUE
Attribute : Transmission consistency status [TRUE,FALSE]
Attribute : Production consistency required [TRUE,FALSE]
Constraint : Production consistency required = TRUE

Attribute : Production consistency status [TRUE,FALSE]
Attribute : Punctual transmission consistency required [TRUE,FALSE]
Constraint : Punctual transmission consistency required = TRUE
Attribute : Punctual transmission consistency status [TRUE,FALSE]
Attribute : Punctual production consistency required [TRUE,FALSE]
Constraint : Punctual production consistency required = TRUE
Attribute : Punctual production consistency status [TRUE,FALSE]
Constraint : Spatial consistency required = TRUE
Attribute : *Reference* (Consistency) Produced Variable
Attribute : Spatial consistency status [TRUE,FALSE]
Constraint : Recovery required = TRUE
Attribute : Recovery list size
Attribute : Recovery list contents
Attribute : Recovery Nature [FRAGMENTED,INDIVISIBLE]
Attribute : Recovery List Identifier
Attribute : *Inherit from* Time-out

6.2.1.4.2.1.2 Attributs

A_Name:

Cet attribut prend en charge une identification unique au niveau de l'interface entre la couche application et l'utilisateur. Le A_Name d'une liste de variables appartient à l'espace d'adressage des services MPS

En outre, il convient que la longueur maximale d'un A_Name d'une liste de variables ne dépasse pas 14 caractères.

List Of Reference Consumed Variable:

Cet attribut permet à une liste de variables de faire référence aux variables constitutives. Cet ensemble de références entre une classe *Identified Variable List* et des classes *Consumed variable* est identique pour chaque image locale.

Spatial consistency required:

Le statut de cohérence spatiale associé à une liste de variables est facultatif.

Une valeur "TRUE" pour cet attribut booléen indique qu'il convient d'élaborer un statut de cohérence spatiale au niveau de la liste.

Recovery required:

La mise en œuvre du mécanisme de récupération est facultative.

Une valeur "TRUE" pour cet attribut booléen indique qu'il convient de mettre en œuvre un mécanisme de récupération pour la liste de variables.

***Reference* (synchronization) Variable:**

Les listes de variables se rapportent à un ordre de synchronisation lorsqu'un statut de cohérence spatiale est élaboré ou lorsque le mécanisme de récupération est mis en œuvre. Cet ordre de synchronisation est utilisé d'une part par les éléments de service de différents abonnés qui élaborent les valeurs des variables de cohérence qui sont impliquées dans l'élaboration du statut de cohérence spatiale, et d'autre part par les éléments de service qui gèrent localement le mécanisme de récupération au niveau de chaque entité d'application.

Inconsistency detection:

Cet attribut spécifie le type des variables de cohérence qui sont impliquées dans l'élaboration du statut de cohérence spatiale.

La détection d'incohérence est TRANSITORY lorsque le type de la valeur de la variable de cohérence prend en compte les erreurs de transmission lors du dernier échange.

La détection d'incohérence est PERMANENT lorsque le type de la valeur de la variable de cohérence prend en compte les erreurs de transmission des derniers 2^{16} échanges précédents.

La détection d'incohérence est UNDEFINED lorsque le type de la valeur de la variable de cohérence est librement défini par l'utilisateur pour l'élaboration du statut de cohérence spatiale dans son application utilisateur.

***Reference* (consistency) Variable:**

Cet ensemble de références à des variables de cohérence prend en charge localement le statut d'émission des variables de la liste dans les autres entités d'application consommant cette liste.

Une variable de cohérence caractérise une liste de variables au niveau d'une entité d'application particulière. Elle fournit des informations concernant la validité de l'émission de chacune des variables de cette liste depuis l'émission de la dernière variable de synchronisation associée à la liste.

Recovery period:

Cet attribut représente la durée maximale autorisée entre deux réinitialisations du contenu de la liste de récupération.

Cette période est exprimée en millisecondes.

Transmission consistency required:

La cohérence d'émission associée à une liste de variables est facultative.

Lorsqu'il a la valeur 'TRUE", cet attribut indique qu'il convient d'élaborer un statut de cohérence d'émission pour cette liste.

Transmission consistency status:

Le statut de cohérence d'émission fourni à l'utilisateur est de type boolean.

Ce statut est élaboré au niveau de chaque entité d'application recevant les variables d'une liste définie avec un statut de cohérence d'émission.

Production consistency required:

La cohérence de production associée à une liste de variables est facultative.

Lorsqu'il a la valeur 'TRUE", cet attribut indique qu'il convient d'élaborer un statut de cohérence de production pour cette liste.

Production consistency status:

La cohérence de production est un statut fourni à l'utilisateur.

Ce statut, de type Boolean (booléen), est élaboré au niveau de chaque entité d'application recevant les variables d'une liste avec un statut de cohérence de production.

Punctual transmission consistency required:

La cohérence d'émission ponctuelle associée à une liste de variables est facultative.

Lorsqu'il a la valeur 'TRUE", cet attribut indique qu'il convient d'élaborer un statut de cohérence d'émission ponctuelle pour cette liste.

Punctual transmission consistency status:

La cohérence d'émission ponctuelle est un statut fourni à l'utilisateur.

Ce statut, de type Boolean (booléen), est élaboré au niveau de chaque entité d'application recevant les variables d'une liste définie avec un statut de cohérence d'émission ponctuelle.

Punctual production consistency required:

La cohérence de production ponctuelle associée à une liste de variables est facultative.

Cet attribut de type booléen indique qu'il convient d'élaborer un statut de cohérence de production ponctuelle pour cette liste quand elle a la valeur TRUE.

Punctual production consistency status:

La cohérence de production ponctuelle est un statut fourni à l'utilisateur.

Ce statut, de type booléen, est élaboré au niveau de chaque entité d'application recevant les variables d'une liste définie avec un statut de cohérence de production ponctuelle.

Spatial consistency status:

La cohérence spatiale est un statut fourni à l'utilisateur.

Ce statut, de type booléen, est élaboré au niveau de chaque entité d'application recevant les variables d'une liste définie avec un statut de cohérence spatiale.

***Reference* (consistency) Produced Variable:**

Cette référence désigne la variable de cohérence produite localement et émise aux autres entités d'application consommant la liste afin de les informer de l'opération d'émission des variables localement dans l'entité produisant la variable de cohérence.

Recovery list size:

Cet attribut caractérise le nombre maximal d'éléments de la liste de variables pour laquelle une retransmission sera demandée pendant le même cycle.La taille de la liste de récupération est toujours inférieure ou égale au nombre d'éléments qui composent la liste de variables

Recovery list contents:

Cet attribut caractérise un ensemble d'identificateurs associés à des éléments d'une liste de variables, qui n'ont pas été reçus avec succès après la dernière réception sur le réseau de la variable de synchronisation associée à cette liste.

Ce contenu n'est pas accessible à l'utilisateur, mais il est directement interprétable par un élément de service de liaison de données pour récupérer les émissions échouées au cours du dernier échange.

Le contenu d'une liste de récupération est utilisé pour effectuer une demande de récupération d'émission du nombre maximal d'éléments définis par la taille de la liste de récupération.

La valeur codée de la liste de récupération est vide en cas de non récupération, c'est-à-dire lorsque toutes les variables de la liste de variables ont été correctement émises au cours du dernier échange.

Recovery list identifier:

Cet attribut spécifie l'identificateur auquel le contenu de la liste de récupération est associé. Cet identificateur fait référence d'une manière unique à cette liste de récupération associée à une image locale d'une liste de variables au sein d'une entité d'application MPS.

Inherit from time-out:

La gestion du contenu de la liste de récupération prend en compte l'événement associé à l'expiration du délai d'attente qui indique un temps considéré trop long sans initialisation de la liste de récupération, ainsi qu'un événement associé à l'occurrence de l'ordre de synchronisation qui initialise le contenu de la liste de récupération. Cette temporisation est établie à la suite de l'occurrence de la synchronisation de la liste de récupération et elle est réinitialisée automatiquement lorsqu'elle expire.

Recovery nature:

Cet attribut définit la qualité de service associée au mécanisme de récupération sur une liste de variables. Lorsque cet attribut a la valeur INDIVISIBLE, la récupération est accomplie immédiatement après émission du contenu de la liste de récupération sur le réseau. Lorsque cet attribut a la valeur FRAGMENTED, la récupération est retardée dans un intervalle de temps réservé au trafic non périodique; de plus, il convient qu'il soit effectué uniquement selon la charge du réseau.

6.2.1.4.2.2 Spécification de la classe *Identified Variable List*

La classe *Identified Variable List* provient de l'instanciation de la classe générique *Variable List*.

6.2.1.4.2.3 Spécification de l'objet *Variable List Local Image*

L'objet *Variable List Local Image* provient de l'instanciation de la classe *Identified Variable List*.

6.2.1.4.3 Services

6.2.1.4.3.1 Service List read

6.2.1.4.3.1.1 Fonction

Cette fonction permet à l'utilisateur d'accomplir une lecture locale de toutes les valeurs des variables composant une liste localement déclarée comme étant consommée.

En plus des valeurs des variables de la liste, ce service fournit facultativement les divers statuts de cohérence associés à cette liste de variables.

6.2.1.4.3.1.2 Primitives de service

La lecture d'une liste de variables est accomplie en utilisant les primitives de service A_READLIST (voir Tableau 31):

A_READLIST.Rq (Arguments): Demande de lecture de liste

A_READLIST.Cnf (Results): Retour de la valeur lue

Tableau 31 – Paramètres du service A_Readlist

Nom de paramètre	Req	Cnf
Argument		
List specification	M	M (=)
Result(+)		S
Value		M
Transmission consistency status		C
Production consistency status		C
Punctual consistency status		C
Punctual production consistency status		C
Spatial consistency status		S
Result(-)		S
Type of error		M

NOTE La méthode par laquelle une primitive "confirm" est corrélée à sa primitive "request" précédente correspondante relève d'une initiative locale. Voir 1.2.

Argument

List specification:

Cet argument correspond à l'attribut A_Name de l'objet *Variable List Local Image* qui doit être accessible.

Result(+)

Valeur

Liste des valeurs de variables appartenant à la liste qui est lue.

Transmission consistency status

Statut facultatif décrivant la cohérence d'émission de toutes les variables de la liste.

Production consistency status

Statut facultatif décrivant la cohérence de production de toutes les variables de la liste.

Punctual transmission consistency status

Statut facultatif décrivant la cohérence d'émission ponctuelle de toutes les variables de la liste.

Punctual production consistency status

Statut facultatif décrivant la cohérence de production ponctuelle de toutes les variables de la liste.

Spatial consistency status

Statut facultatif décrivant la cohérence spatiale de toutes les variables de la liste.

Result(-)

Error type

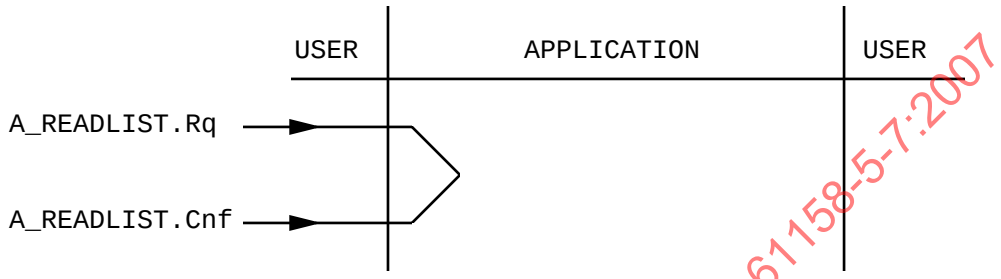
Description des divers types d'erreurs retournés après une demande de lecture inaboutie.

Error_1: Un des tampons contenant la valeur d'une variable de la liste n'est pas accessible.

Error_2: La spécification de la liste ne correspond pas, au sein de l'espace d'adressage, à l'identification de la liste déclarée comme étant consommée dans l'AP considéré.

Error_3: La liste est déclarée localement comme étant invalide.

6.2.1.4.3.1.3 Procédure de service



Légende

Anglais	Français
USER	UTILISATEUR
APPLICATION	APPLICATION

Figure 43 – Procédure du service A_Readlist

Ce service, montré à la Figure 43, fournit toutes les valeurs disponibles localement des valeurs d'une liste à un instant donné.

Les divers statuts élaborés décrivent les divers niveaux de cohérence appliqués aux variables de la liste.

Les demandes de lecture d'une liste sont traitées séquentiellement; une nouvelle demande ne peut être déclenchée qu'après confirmation de la précédente.

6.2.1.4.4 Cohérence d'une liste de variables et mécanismes de récupération

6.2.1.4.4.1 Cohérence d'émission

6.2.1.4.4.1.1 Description

Les informations relatives à la cohérence de l'émission d'une liste de variables peuvent être élaborées au sein d'une entité consommatrice et fournies à l'utilisateur.

Ces informations relatives à la cohérence sont fournies avec deux sémantiques possibles:

- la cohérence d'émission asynchrone donne à l'utilisateur consommateur les informations relatives au respect, pour toutes les variables de cette liste, de leur période respective de consommation par le réseau,
- la cohérence d'émission synchrone donne à l'utilisateur consommateur les informations relatives au respect, pour toutes les variables de cette liste, d'une disponibilité dans les limites de leur période respective de consommation initialisée à la suite de l'occurrence d'une commande unique de synchronisation associée à la liste.

En outre, ces promptitudes synchrone ou asynchrone sont élaborées avec une valeur obligatoire de l'attribut *Consumption Period* qui est identique pour toutes les variables de cette liste.

Dans le cas d'une cohérence d'émission asynchrone, il convient que toutes les variables qui composent cette liste élaborent un statut de promptitude asynchrone.

Dans le cas d'une cohérence d'émission synchrone, il convient que toutes les variables qui composent cette liste élaborent un statut de promptitude synchrone.

6.2.1.4.4.1.2 Règle de composition

Le statut de cohérence d'émission est le produit logique du statut de promptitude des différentes variables de la liste. Selon les caractéristiques synchrones ou asynchrones du statut de promptitude associé aux variables de la liste, il convient que le statut de cohérence de l'émission qui en résulte représente la cohérence d'émission qui est respectivement synchrone ou asynchrone.

6.2.1.4.4.2 Cohérence de production

6.2.1.4.4.2.1 Description

Les informations relatives à la cohérence de la production d'une liste de variables peuvent être élaborées au niveau d'une entité consommatrice et mises à la disposition de l'utilisateur.

Ces informations relatives à la cohérence sont fournies avec deux sémantiques possibles:

- la cohérence de production asynchrone donne à l'utilisateur consommateur les informations relatives au respect, par tous les utilisateurs produisant les variables de cette liste, de leur période respective de production,
- la cohérence de production asynchrone donne à l'utilisateur consommateur les informations relatives au respect, par tous les utilisateurs produisant les variables de cette liste, d'un rafraîchissement dans les limites de leur période respective de production déclenchée par l'entité productrice à la suite de l'occurrence d'un ordre de synchronisation associé à la liste de variables.

Dans le cas d'une cohérence de production asynchrone, il convient que toutes les variables qui composent cette liste élaborent un statut de rafraîchissement asynchrone, alors que dans le cas d'une cohérence de production synchrone, il convient que toutes les variables qui composent cette liste élaborent un statut de rafraîchissement synchrone. En outre, ces statuts de rafraîchissement asynchrone et synchrone sont élaborés avec une valeur obligatoire de l'attribut *Production Period* qui est identique pour toutes les variables de la liste.

6.2.1.4.4.2.2 Règle de composition

Le statut de cohérence de production est le produit logique du statut de rafraîchissement des diverses variables de la liste.

Selon les caractéristiques synchrones ou asynchrones du statut de rafraîchissement élaboré par les entités produisant les variables de la liste, il convient que le statut représente la cohérence de production qui est respectivement synchrone ou asynchrone.

6.2.1.4.4.3 Cohérence d'émission ponctuelle

6.2.1.4.4.3.1 Description

Les informations relatives à la cohérence de l'émission ponctuelle d'une liste de variables peuvent être élaborées au sein d'une entité consommatrice et fournies à l'utilisateur.

Le statut de cohérence d'émission ponctuelle donne à l'utilisateur consommateur les informations relatives au respect d'une disponibilité par le réseau de toutes les variables de cette liste dans les limites leur intervalle de temps respectif de consommation.

Les intervalles de temps respectifs de consommation sont déclenchés par l'entité consommatrice à la suite de l'occurrence de l'ordre de synchronisation associé à la liste.

Il convient que toutes les variables composant la liste de variables qui élaborent un statut de cohérence d'émission ponctuelle élaborent un statut de promptitude ponctuelle avec la même valeur de l'attribut *consumption time slot*.

6.2.1.4.4.3.2 Règle de composition

Le statut de cohérence d'émission ponctuelle est le produit logique du statut de promptitude ponctuelle de toutes les variables composant cette liste.

6.2.1.4.4.4 Cohérence de production ponctuelle

6.2.1.4.4.4.1 Description

Les informations relatives à la cohérence de la production ponctuelle d'une liste de variables peuvent être élaborées au sein d'une entité consommatrice et fournies à l'utilisateur.

Le statut de cohérence de production ponctuelle donne à l'utilisateur consommant la liste de variables les informations relatives au respect, par tous les utilisateurs produisant les variables de cette liste, d'une production dans les limites leur intervalle de temps respectif.

Tous les intervalles de temps de production des variables de la liste sont déclenchés par les entités productrices à la suite de l'occurrence de l'ordre de synchronisation associé à cette liste. Il convient que les variables composant la liste de variables qui élaborent un statut de cohérence de rafraîchissement ponctuel élaborent un statut de promptitude ponctuelle avec la même valeur de l'attribut *production time slot*.

6.2.1.4.4.4.2 Règle de composition

Le statut de cohérence de production ponctuelle est le produit logique du statut de rafraîchissement ponctuel de toutes les variables composant la liste.

6.2.1.4.4.5 Cohérence spatiale

6.2.1.4.4.5.1 Description

Un statut relatif à la cohérence spatiale d'une liste de variables peut être élaboré au niveau d'une entité consommatrice et mis à la disposition de l'utilisateur.

Le statut de cohérence spatiale est un élément d'information à partir duquel l'utilisateur sera à même de bâtir un statut de cohérence dont la sémantique est adaptée aux besoins de l'application. (Par exemple, la sémantique du statut de cohérence de l'utilisateur peut être l'égalité des valeurs de toutes les multiples copies des variables de la liste.)

Le mécanisme d'élaboration d'un statut de cohérence spatiale s'appuie sur la diffusion par toutes les entités consommant la même liste déclarée avec une caractéristique de cohérence spatiale d'une variable de cohérence. Cette variable de cohérence reflète localement la validité de l'émission de chacune des variables de la liste.

Une variable de cohérence est élaborée par chacune des entités consommant une seule liste de variables et elle est fournie aux autres entités consommant cette liste.

L'élaboration de ce statut de cohérence spatiale exige que chacune des AE consommant une seule liste de variables soit pourvue d'un rapport de réception des variables de liste des autres AE consommatrices. Afin de fournir le rapport de réception, des échanges supplémentaires permettent à chaque AE de diffuser aux autres sa variable de cohérence relative à cette même liste.

6.2.1.4.4.5.2 Règle de composition

Au niveau d'une entité consommant une liste variable, le traitement associé à la cohérence spatiale s'appuie sur les valeurs des variables de cohérence associées à cette liste. Le statut de cohérence spatiale est élaboré à partir de la comparaison des valeurs des variables de cohérence associées à une liste.

La valeur du statut de cohérence spatiale est

TRUE: lorsqu'il y a égalité entre toutes valeurs des variables de cohérence associées à cette liste.

FALSE: dans tous les autres cas.

6.2.1.4.4.5.3 Spécification de la variable de cohérence

Lorsqu'un statut de cohérence spatiale est associé à une liste de variables, il est nécessaire d'élaborer une variable de cohérence au niveau de chaque entité d'application consommant cette liste de variables.

Cette variable de cohérence est fournie aux autres entités consommant cette liste de variables.

Une variable de cohérence dans l'environnement MPS est modélisée de la même manière utilisée pour les variables en utilisant les objets suivants:

– un objet (consistency) *Produced Variable Local Image* qui contient tous les attributs requis pour une entité d'application productrice,
– une ou plusieurs objets (consistency) *Consumed Variable Local Image* qui contiennent tous les attributs requis pour des entités d'application consommatrices.

Ces objets correspondent à l'instanciation particulière des classes décrivant les variables pour lesquelles l'attribut *Class* de l'objet a la valeur "SYNCHRONIZATION".

Une variable de cohérence diffère des autres classes de variables par une référence à la liste de variables associée, afin que l'élément de service produisant cette variable puisse élaborer la valeur correspondante.

6.2.1.4.4.6 Valeurs prédéterminées utilisées par les mécanismes de cohérence de liste

6.2.1.4.4.6.1 Valeur d'attribut prédéterminée de la classe *Identified Variable*

Une classe *Identified Variable* correspondant à une variable de cohérence est caractérisée par des valeurs prédéterminées pour les attributs suivants:

Class:

Cet attribut a la valeur: SYNCHRONIZATION

Variable DE cohérence:

Cet attribut a la valeur: TRUE

6.2.1.4.4.6.2 Valeur prédéterminée des attributs de la classe *Produced Variable Local Image*

Un objet *Produced Variable Local Image* correspondant à une variable de cohérence est caractérisé par des valeurs prédéterminées pour les attributs suivants:

Resynchronized Variable:

Cet attribut a la valeur: FALSE.

Tous les autres attributs non cotés ne se donnent pas de valeurs prédéterminées.

6.2.1.4.4.6.3 Valeur prédéterminée d'attribut de l'objet *Consumed Variable Local Image*

Un objet *Consumed Variable Local Image* correspondant à une variable de cohérence est caractérisé par des valeurs prédéterminées pour les attributs suivants:

Resynchronized Variable:

Cet attribut a la valeur: FALSE.

Tous les autres attributs non cotés ne se donnent pas de valeurs prédéterminées.

6.2.1.4.4.6.4 Spécification de type de la variable de cohérence

6.2.1.4.4.6.4.1 Fonctionnalité de la variable de cohérence

Les valeurs des variables de cohérence sont assignées avec un type issu de l'instanciation de la classe *Type Constructor*.

Lorsque l'attribut *Inconsistency Detection* a la valeur 'TRANSITORY', il convient que le type de la variable de cohérence soit 'K_SCONS'.

Lorsque l'attribut *Inconsistency Detection* a la valeur 'FRAGMENTED', il convient que le type de la variable de cohérence soit 'K_LCONS'.

Lorsque l'attribut *Inconsistency Detection* a la valeur 'UNDEFINED', le type de la variable de cohérence peut être décrit par toute instance de la classe *Type Constructor*.

6.2.1.4.4.6.4.2 Valeur d'attribut prédéterminée de la classe *Type Constructor*

Instances de la classe *Type Constructor* associée à une variable de cohérence pour détecter une incohérence spatiale "TRANSITORY":

A_Name:		K_SCONS
Construction	:	STRUCTURE
Named field	:	FALSE
Reference	:	K_UNS_16 -- Interchange Nr.
Reference	:	K_BOOLAC64 -- Transmission status
A_Name	:	K_UNS_16
Construction	:	SIMPLE
Class of type		UNSIGNED
Size	:	16
A_Name	:	K_BOOLAC*nn*
Construction	:	ARRAY
Compacted	:	TRUE
Dimension	:	*nn* (nombre de variables de liste)
Reference	:	K_BOOL

A_Name : K_BOOL
Construction : SIMPLE
Class of type BOOLEAN

Instances de la classe *Type Constructor* associée à une variable de cohérence pour détecter une incohérence spatiale "PERMANENT":

A_Name: K_LCONS
Construction : STRUCTURE
Named field : FALSE
Reference : K_UNS_16 -- Interchange Nr.
Reference : K_UNS_16AC*nn* -- Transmission_date
A_Name : K_UNS_16AC*nn*
Construction : ARRAY
Compacted : TRUE
Dimension : *nn* (nombre de variables de liste)
Reference : K_UNS_16

6.2.1.4.4.6.4.3 Sémantique de champs

La sémantique des champs de valeur de la variable de cohérence lorsque l'attribut *Inconsistency Detection* n'a pas la valeur UNDEFINED est comme suit:

Champs de structure associés au type 'K_SCONS':
Interchange number:

La valeur des variables de la liste est caractérisée par le nombre d'échanges au cours desquels elles ont été mises à jour par le réseau. Ce numéro d'échange est la donnée globale de l'environnement MPS produite par une entité d'application spécifique et il est diffusé aux entités élaborant des informations de cohérence spatiale en utilisant la variable de synchronisation associée à la liste.

Ce numéro d'échange est utilisé pour vérifier que les dates des valeurs de variables de cohérence traitées sont identiques pendant le traitement de la cohérence spatiale.

Transmission_status:

Le statut d'émission de la liste de variables est une matrice dont les éléments représentent respectivement le statut d'émission pour chacune des variables de la liste.

Le statut d'émission d'une variable au niveau d'une entité d'application consommatrice indique si cette variable a été reçue depuis la dernière occurrence de la variable de synchronisation associée à la liste.

Champs de structure associés au type 'K_LCONS':

Interchange number:

Ce champ est pourvu de la même sémantique que celle ayant le même nom dans le type "K_SCONS".

Transmission_date:

La date d'émission de la liste de variables est une matrice dont les éléments représentent respectivement le numéro du dernier échange de chacune des variables de la liste.

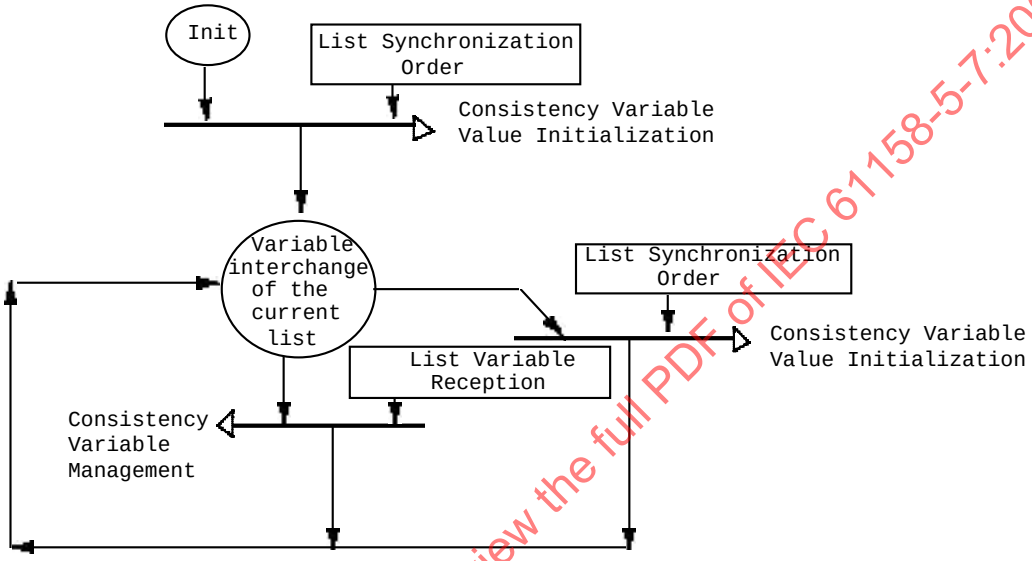
Le numéro du dernier échange d'une variable au niveau d'une entité d'application consommatrice correspond à celui qui est pris en compte pendant la dernière réception de cette variable.

Ce numéro d'échange, pris en considération pour une variable pendant sa réception, correspond à la valeur de la variable de synchronisation de la liste à laquelle cette variable appartient.

6.2.1.4.4.7 Spécification du mécanisme d'élaboration des valeurs de variables de cohérence

Ce mécanisme est impliqué seulement lorsque l'attribut *Inconsistency detection* associé à la liste de variables a une valeur autre que "UNDEFINED".

Cet élément de service élabore la valeur de type prédéterminé impliquée dans l'élaboration du statut de cohérence spatiale d'une liste de variables (voir Figure 44).



Légende

Anglais	Français
Init	Initialiser
Consistency Variable Value Initialization	Initialisation de la valeur de la variable de cohérence
Variable interchange of the current list	Échange de variables de la liste courante
List Variable Reception	Réception des variables de la liste
List Synchronization Order	Ordre/commande de synchronisation de la liste
Consistency Variable Management	Gestion des variables de cohérence

Figure 44 – Diagramme d'évaluation de la valeur de variable de cohérence

L'initialisation de la valeur de la variable de cohérence à la suite de l'occurrence de l'ordre de synchronisation associé à une liste de variables consiste à donner à ses différents champs une valeur spécifique qui caractérise la mise en œuvre du traitement de cohérence.

La valeur de la variable de cohérence à la fin de l'initialisation est comme suit:

– le champ ' Interchange number ' spécifie le numéro d'échange correspondant à la valeur de la variable de synchronisation qui a déclenché cette initialisation,

– le champ 'Transmission status' (statut d'émission) spécifie, pour chaque variable de la liste, un statut d'émission 'FALSE',

– le champ 'Transmission date' (date d'émission) spécifie, pour chaque variable contenue dans la liste de variables avec laquelle la variable de cohérence est associée, un numéro d'échange correspondant à celui qui a déclenché l'initialisation.

L'initialisation d'un cycle de cohérence est accomplie à la suite de l'occurrence de l'ordre de synchronisation associé à la liste de variables. Cette initialisation correspond à une affectation du champ de variable de cohérence "*Interchange number*" de la valeur de la variable de synchronisation qui correspond de l'ordre de synchronisation de la liste. La gestion de la valeur de la variable de cohérence à la suite de la réception d'une variable consiste à modifier la valeur des champs Transmission_Status et Transmission_Date selon le type prédéterminé sélectionné.

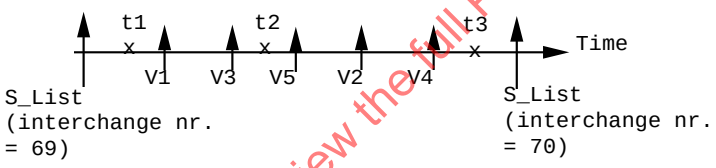
Dans le cas du champ Transmission_Status, la gestion de valeur consiste à réaliser une modification du statut d'émission de chacune des variables de la liste reçue de FALSE à TRUE.

Dans le cas du champ Transmission_date, la valeur de gestion consiste à mémoriser le nombre actuel d'échange dans la date d'émission associée à la variable.

EXEMPLE

Soient la liste {V1, V2, V3, V4, V5} et la variable de synchronisation associée: S_LIST

Chronogramme d'échange:



Légende

Anglais	Français
S_List (interchange nr. = 69)	S_List (nombre d'échanges. = 69)
V1 V3 V5 V2 V4	V1 V3 V5 V2 V4
S_List (interchange nr. = 70)	S_List (nombre d'échanges = 70)
Time	Temps

Figure 45 – Chronogramme d'échange de cohérence

Si nous considérons que toutes variables ont été reçues au cours de l'échange précédent, la variable de cohérence associée aux divers instants de l'échange a les valeurs suivantes:

t1 : Valeur de la variable de cohérence
Inconsistency Detection:= TRANSITORY {69; 0,0,0,0,0}
Inconsistency Detection:= PERMANENT {69; 68,68,68,68,68}
t2 : Valeur de la variable de cohérence
Inconsistency Detection:= TRANSITORY {69; 1,0,1,0,0}
Inconsistency Detection:= PERMANENT {69; 69,68,69,68,68}
t3 : Valeur de la variable de cohérence
Inconsistency Detection:= TRANSITORY {69; 1,1,1,1,1}
Inconsistency Detection:= PERMANENT {69; 69,69,69,69,69}

6.2.1.4.4.8 Accès aux variables de cohérence

Les variables de cohérence sont traitées par un sous-ensemble des services fournis à l'utilisateur pour des variables normales.

Les services permettant l'accès aux variables de cohérence sont ceux qui existent pour les variables de synchronisation, avec la restriction suivante:

A_WRITELOC: Seulement lorsque l'attribut *Inconsistency detection* a une valeur autre que UNDEFINED.

6.2.1.4.4.9 Spécification du mécanisme de récupération

Une liste de variables peut être pourvue d'un mécanisme de récupération dont la mise en œuvre dépend de cette liste selon les besoins de l'utilisateur relatifs à la fiabilité.

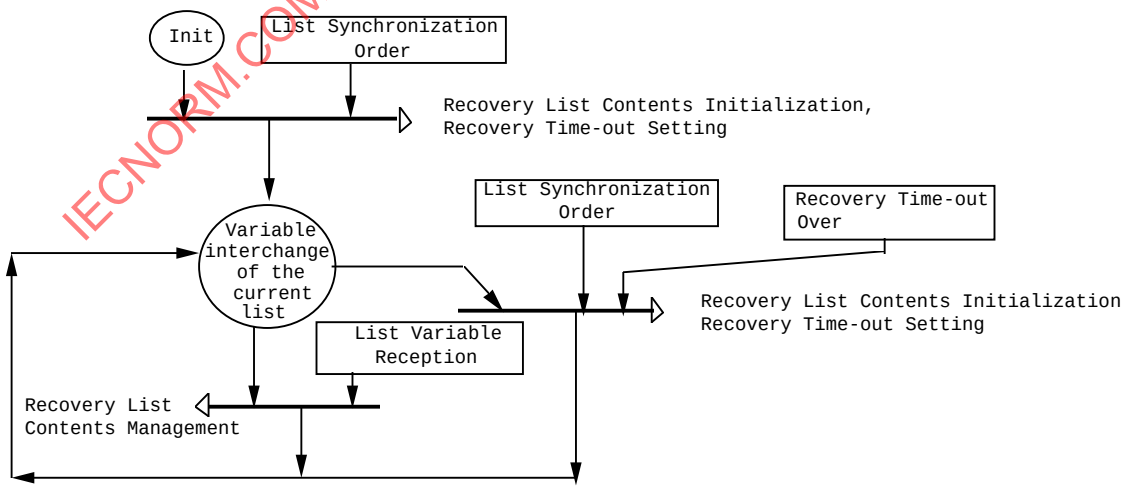
Ce mécanisme de récupération élabore le contenu de la liste de récupération correspondant à un ensemble ordonné de références à des éléments de la liste de variables qui n'ont pas été reçus avec succès depuis la dernière réception de la variable de synchronisation de liste, sous la condition que cette variable de synchronisation soit arrivée dans un délai inférieur à la période de récupération.

L'exploitation de la séquence des références de variables dans la liste de récupération est accomplie dans l'ordre de déclaration des variables de liste.

Les références aux variables de liste, stockées dans le contenu de récupération par ce mécanisme, correspondent aux identificateurs de variables de cette liste. Ce contenu de récupération qui conditionne le déclenchement de nouveaux échanges est limité à un nombre maximal de références définies au moment de la configuration de la liste de variables.

NOTE L'utilisation du mécanisme de récupération, avec une période de récupération équivalant à la période de réseau de la variable de synchronisation de la liste, libère toutes les pertes qui sont apparues pendant l'émission de la variable de synchronisation de liste.

La Figure 46 montre le réseau d'évaluation de l'élément de service qui élabore le contenu de la liste de récupération associée à une liste de variables au niveau de l'application.



Légende

Anglais	Français
Init	Initialiser
List Synchronization Order	Ordre/commande de synchronisation de la liste

Anglais	Français
Recovery List Contents Initialization,Recovery Time-out Setting	Initialisation du contenu de la liste de récupération, réglage de la temporisation de la récupération
Variable interchange of the current list	Échange de variables de la liste courante
Recovery Time-out Ove	Temporisation de récupération terminée
List Variable Reception	Réception des variables de la liste
Recovery List Contents Managment	Gestion du contenu de la liste de récupération

Figure 46 – Diagramme d'évaluation du mécanisme de récupération

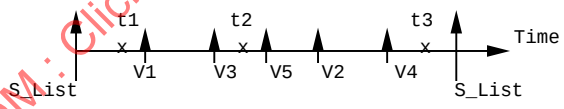
• L'initialisation du contenu de la liste de récupération, à la suite de l'occurrence de l'ordre de synchronisation associé à la liste de variables, consiste à lui donner la liste de tous les identificateurs des variables de la liste comme valeur de contenu.

Sachant que la variable de synchronisation associée à la liste précède toutes les variables de la liste pendant un échange, il convient de considérer que toutes variables de la liste ne sont pas reçues à l'instant de l'occurrence de l'ordre de synchronisation.

- La gestion de la liste de récupération à la suite de la réception d'une variable consiste à retirer l'identificateur associé à cette variable de cette liste de récupération.
- Le réglage de la valeur de la temporisation de la récupération est accompli à la suite de l'occurrence de l'ordre de synchronisation de liste et automatiquement également lorsqu'elle expire. Lorsqu'elle est réglée, la valeur de l'attribut *Preselection* de cette temporisation est celle de l'attribut *Recovery period* de l'objet *Variable List Local Image*.

Exemple:

Soient la liste {V1, V2, V3, V4, V5} et la variable de synchronisation associée: S_LIST. La taille maximale de la liste de récupération est limitée à 4 éléments.



Légende

Anglais	Français
Time	Temps

Figure 47 – Chronogramme d'échange de récupération

La liste de récupération, montrée à la Figure 47, a les valeurs suivantes aux différents instants d'échange:

t1 : Contenu de la liste de récupération = {V1, V2, V3, V4}

t2 : Contenu de la liste de récupération = {V2, V4, V5}

t3 : Contenu de la liste de récupération = { ∅ }

6.2.1.5 Description de la configuration

La spécification de la description des objets de la couche application se rapporte aux attributs statiques des objets:

– Variable de classe normale, synchronisation (en se limitant aux attributs partagés par le producteur et le consommateur plus ceux qui sont spécifiques au producteur).
– Type d'une variable.
– Accès à une variable.
– Liste de variables (en limitant à ces attributs partagés par les consommateurs de cette liste).

6.2.1.5.1 Variables de description

Il convient que la description de configuration accessible à l'utilisateur n'importe où sur le réseau soit adressée à l'interface entre la couche application et l'utilisateur.

À cet effet, la description de configuration associée à divers objets de la couche application est modélisée par les variables de description.

Ces variables de description sont elles-mêmes soumises à des contraintes particulières qui leur apportent la même configuration sur le réseau, afin de ne pas définir de variables de description récursives décrivant la configuration de variable de description.

Ces contraintes particulières sont converties au niveau de la couche application vers des valeurs prédéfinies des objets *Consumed Variable Local Image* et *Produced Variable Local Image* associés aux variables de description.

6.2.1.5.2 Spécification d'une variable de description

6.2.1.5.2.1 Vue d'ensemble d'une variable de description

Une variable de description dans l'environnement AL est modélisée de la même manière que les autres variables en utilisant les objets suivants:

- Un objet *Produced Variable Local Image*, qui inclut tous les attributs requis pour la définition d'une variable de description dans une entité d'application productrice.
– Un objet *Consumed Variable Local Image*, qui inclut tous les attributs requis pour la définition d'une variable de description dans une entité d'application consommatrice.

Ces deux catégories d'objet correspondent à l'instanciation particulière de la classe *Identified Variable*, pour laquelle l'attribut *Class* de l'objet a la valeur DESCRIPTION.

Un objet *Produced Variable Local Image* ou *Consumed Variable Local Image* correspondant à une variable de la classe DESCRIPTION est différent de celui correspondant à des variables 'normales' par des valeurs prédéterminées de certains de ses attributs.

6.2.1.5.2.2 Valeurs prédéterminées des attributs de la classe *Identified Variable*.

Une classe *Identified Variable* correspondant à une variable de description est caractérisée par des valeurs prédéterminées pour les attributs suivants:

Transmitted Status:

Cet attribut a la valeur ' FALSE '.

Class:

Cet attribut a la valeur ' DESCRIPTION '.

Tous les attributs non cotés ne se donnent pas de valeurs prédéterminées.

6.2.1.5.2.3 Valeurs prédéterminées des attributs de l'objet *Produced Variable Local Image*

Un objet *Produced Variable Local Image* correspondant à une variable de description est caractérisé par des valeurs prédéterminées pour les attributs suivants:

Resynchronized Variable:

Cet attribut a la valeur ' FALSE '.

Elaborated Refreshment:

Cet attribut a la valeur ' FALSE '.

Elaborated Punctual Refreshment:

Cet attribut a la valeur ' FALSE '.

Required Universal Services:

Cet attribut a la valeur ' FALSE '.

Tous les attributs non cotés ne se donnent pas de valeurs prédéterminées.

6.2.1.5.2.4 Valeurs prédéterminées des attributs de l'objet *Consumed Variable Local Image*

Un objet *Consumed Variable Local Image* correspondant à une variable de description est caractérisé par des valeurs prédéterminées pour les attributs suivants:

Resynchronized Variable:

Cet attribut a la valeur ' FALSE '.

Required Universal Services:

Cet attribut a la valeur ' FALSE '.

Tous les attributs non cotés ne se donnent pas de valeurs prédéterminées.

6.2.1.5.2.5 Noms de variables de description

Les noms de variables de description traités au niveau de l'interface entre la couche application et l'utilisateur, appartiennent à l'unique espace d'adressage faisant référence aux objets d'application. Dans cette espace d'adressage MPS, la norme définit quatre (ensembles de noms réservés) pour adresser les variables de description:

A_Name de la variable commençant par:

"**V**_" Description d'une variable

"**A**_" Description d'un accès de variable

"**T**_" Description d'un type de variable

"**L**_" Description d'une liste de variables.

6.2.1.5.2.6 Types de variables de description

6.2.1.5.2.6.1 Définition de types de variables de description

Les valeurs des variables de description ont un type qui dépend de la nature de la description, correspondant aux instances prédéfinies de la classe *Type Constructor*.

6.2.1.5.2.6.2 Instances prédéfinies de la classe *Type Constructor*

Quatre instances prédéfinies fournissent la sémantique aux types des variables de description.

- variables de la classe NORMAL, SYNCHRONIZATION.
- Accès renommés aux variables de la classe NORMAL, SYNCHRONIZATION.
- types de variables.
- listes de variables.

Instance de la classe *Type Constructor* associée au type d'une variable de description relative à une variable de la classe NORMAL, SYNCHRONIZATION:

A_Name : K_DVAR
Construction : PREDEFINED

Instance de la classe *Type Constructor* associée au type d'une variable de description d'un accès renommé à une variable:

A_Name : K_DACCESS
Construction : PREDEFINED

Instance de la classe *Type Constructor* associée au type d'une variable de description d'un type:

A_Name : K_DTYPE
Construction : PREDEFINED

Instance de la classe *Type Constructor* associée au type d'une variable de description d'une liste de variables:

A_Name : K_DLIST
Construction : PREDEFINED

NOTE Il doit être énoncé que la description des types prédéfinis de l'environnement signifie seulement qu'ils sont prédéfinis. La connaissance précise de la sémantique des types prédéfinis dans l'environnement AL fait partie de la conformité à la norme MPS.

6.2.1.5.2.6.3 Sémantique des types prédéfinis des variables de description

Les types des variables de description fournissent une sémantique à ces variables. Cette sémantique reflète les valeurs des attributs statiques partagés par toutes les instances des objets décrits ainsi qu'un certain nombre d'attributs spécifiques au producteur de l'objet.

La sémantique du type de variable de description d'une variable de la classe NORMAL ou SYNCHRONISATION décrit les valeurs des attributs suivants, s'ils existent:

A_Name

Type Constructor Reference

Transmitted Status

Significant Status

Identifier (Identificateur)

Transmission Mode

Network Period

Class

Consistency Variable

Variable List Reference

Resynchronized Variable

Synchronization Variable Reference

Elaborated Refreshment

Refreshment Characteristics

Production Period

Synchronization Variable Reference

Elaborated Punctual Refreshment

Time slot

Synchronization Variable Reference

La sémantique du type de variable de description d'un accès renommé à une variable décrit les valeurs des attributs suivants, s'ils existent:

A_Name

Variable Reference

Access Mode

Access Path

La sémantique du type de variable de description d'un type de variable décrit les valeurs des attributs suivants, s'ils existent:

A_Name

Construction

Primitive Type

Size

Compacted

Dimension

Reference Type Constructor

Named Field

Field Name

Reference Type Constructor

La sémantique du type de variable de description d'une liste de variables décrit les valeurs des attributs suivants, s'ils existent:

A_Name

List of Reference Variable

Spatial Consistency Required

Recovery Required

Reference Synchronization Variable

Inconsistency Detection

List of Reference Consistency Variable

Recovery Period.

6.2.1.5.2.6.4 Identificateur de variables de description

Les identificateurs acceptés pour les variables de description appartiennent à un sous-ensemble réservé défini par la norme de gestion de réseau.

En outre, il existe un rapport d'adressage entre les identificateurs des variables et ceux des variables de description qui est défini par la norme de gestion de réseau.

6.2.1.5.2.6.5 Accès aux variables de description

Les variables de description peuvent être traitées par un sous-ensemble de services fournis à l'utilisateur pour des variables normales.

Les services fournissant l'accès aux variables de description sont

- A_WRITELOC
- A_WRITEFAR
- A_READLOC
- A_READFAR
- A_SENT
- A_RECEIVED
- A_UPDATE

Tous les autres services, fournis à l'utilisateur pour une variable normale, ne sont pas autorisés pour une variable de description.

6.2.2 ASE Modèle virtuel d'appareil (VMD)

6.2.2.1 Vue d'ensemble

La modélisation d'un VMD de la Sub-MMS au sein d'une application est similaire à son MMS équivalent.

6.2.2.2 Concepts

6.2.2.2.1 Domaine d'application des objets

Afin de simplifier l'identification des objets, les classes d'objets énumérées n'ont pas le domaine d'application (au sens de la MMS) spécifié explicitement.

En d'autres termes, l'appartenance d'un objet d'une classe donnée à un autre objet d'une autre classe est implicitement connue (variable appartenant à un domaine particulier, par exemple) et il convient qu'elle n'apparaisse pas dans l'identification de l'objet.

6.2.2.2.2 Identification d'objets

L'identification des objets "variable", "variable-liste", "pi", "domaine" et "event" est réalisée par un identificateur capable de prendre la forme soit d'une valeur entière (Index), soit d'une chaîne de caractères (Name). Le nombre de caractères dans la chaîne est indéterminé ou déterminé par les normes d'accompagnement.

Prenant en compte l'élimination du concept de domaine d'application, il convient que l'identification d'un objet soit unique, dans sa classe ou dans un espace commun à toutes classes. Un objet peut être identifié par un index (indice) et/ou un name (nom).

6.2.2.2.3 Gestion d'environnement

6.2.2.2.3.1 Services de gestion d'environnement

Les services de gestion de l'environnement Sub-MMS sont

1- Initiate, permet de négocier l'établissement d'un environnement Sub-MMS.

2- Conclude, permet de proposer la terminaison d'un environnement Sub-MMS négocié.

3- Abort, permet d'arrêter brusquement un environnement Sub-MMS négocié.

Les services Initiate et Abort sont obligatoires pour les appareils prenant en charge la gestion de l'environnement Sub-MMS négocié. Le service Conclude est négociable.

Le service Reject permet la notification d'erreurs de protocole. Il est obligatoire pour tous les équipements.

6.2.2.2.3.2 Type de communication

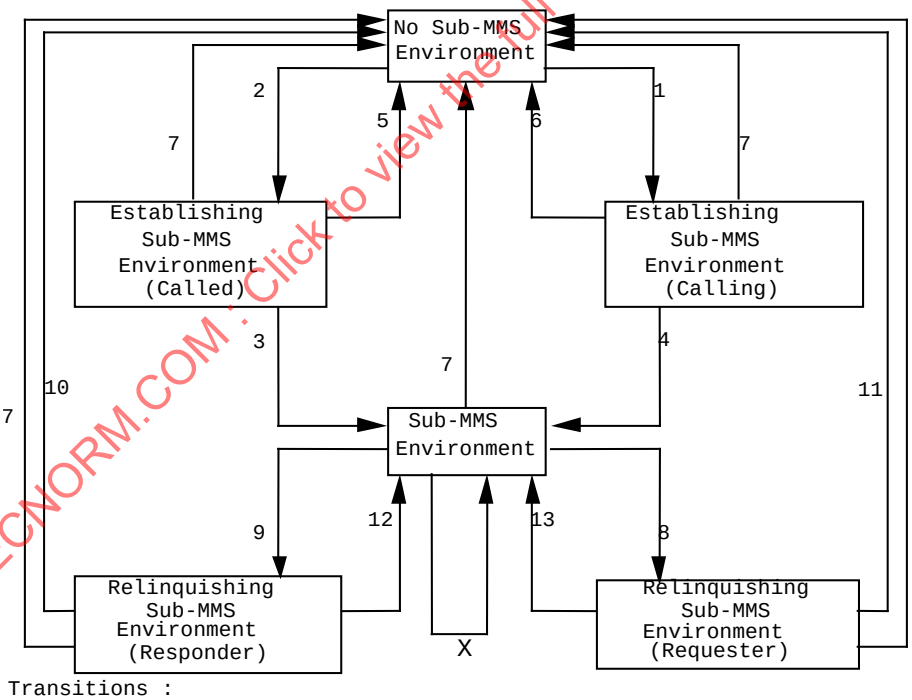
6.2.2.2.3.2.1 Types de communication pris en charge

Un appareil peut prendre en charge les types de communication suivants:

a) communication dans un environnement Sub-MMS négocié,

b) communication dans un environnement Sub-MMS prédéfini,

c) communication sans environnement Sub-MMS.



Transitions :

1-initiate.request
2-initiate.indication
3-initiate.response +
4-initiate.confirm +
5-initiate.response -
6-initiate.confirm -
7-Abort.req or abort.indication
8-conclude.request
9-conclude.indication
10-conclude.response +
11-conclude.confirm +
12-conclude.response -
13-conclude.confirm -
X-other services.Req/ind

Légende

Anglais	Français
No Sub-MMSEnvironment	Pas d'environnement Sub-MMS

Anglais	Français
Establishing Sub-MMS Environment (Called)	Établissement de l'environnement Sub-MMS (Appelé)
Establishing Sub-MMS Environment(Calling)	Établissement de l'environnement Sub-MMS (Appelant)
Relinquishing Sub-MMS Environment(Responder)	Abandon de l'environnement Sub-MMS (Répondeur)
Sub-MMS Environment	Environnement Sub-MMS
Relinquishing Sub-MMS Environment (Requester)	Abandon de l'environnement Sub-MMS (Demandeur)
Transitions :	Transitions :
1-initiate.request	1-initiate.request
2-initiate.indication	2-initiate.indication
3-initiate.response +	3-initiate.response +
4-initiate.confirm +	4-initiate.confirm +
5-initiate.response -	5-initiate.response -
6-initiate.confirm -	6-initiate.confirm -
7-Abort.req or abort.indication	7-Abort.req or abort.indication
8-conclude.request	8-conclude.request
9-conclude.indication	9-conclude.indication
10-conclude.response +	10-conclude.response +
11-conclude.confirm +	11-conclude.confirm +
12-conclude.response -	12-conclude.response -
13-conclude.confirm -	13-conclude.confirm -
X-other services.Req/ind	X-autres services.Req/ind

Figure 48 – Ordinogramme de l'état de gestion de l'environnement Sub-MMS

6.2.2.2.3.2.2 Communication dans un environnement sub-MMS négocié

L'environnement Sub-MMS est négocié entre un utilisateur de la Sub-MMS appelant et un utilisateur de la Sub-MMS appelée.

Les ressources allouées pour cet environnement sont réservées pour ces deux utilisateurs.

L'environnement Sub-MMS connu comme étant une association a plusieurs états (voir Figure 48). Les services initiate, conclude et abort peuvent agir sur l'association.

Il convient que l'état initial pour la Sub-MMS appelante ainsi que pour la Sub-MMS appelée soit l'état "No Sub-MMS Environment" (c'est-à-dire Aucun environnement Sub-MMS).

Dans l'état "No Sub-MMS Environment" (Aucun environnement Sub-MMS), il convient qu'un utilisateur de la sub-MMS n'invoque que la demande de la primitive de service Initiate.

Dans l'état "Establishing Sub-MMS Environment (Calling)" (Établissement d'un environnement Sub-MMS (Appelant)), il convient qu'un utilisateur de la sub-MMS n'invoque que la demande de la primitive de service Abort.

Dans l'état "Establishing Sub-MMS Environment (Called)" (Établissement d'un environnement Sub-MMS (Appelé)), un utilisateur de la sub-MMS ne peut invoquer que la réponse de la primitive de service initiate ou la demande de la primitive de service Abort.

Dans l'état "Relinquishing Sub-MMS Environment (Requester)" (Renonciation d'un environnement Sub-MMS (Demandeur)), la seule demande pour une primitive qu'un utilisateur de la sub-MMS peut invoquer est la demande de la primitive de service Abort. Il est à noter que les réponses (+) ou (-) peuvent continuer à être invoquées.

Dans l'état "Relinquishing Sub-MMS Environment (Responder)" (Renonciation d'un environnement Sub-MMS (Répondeur)), un utilisateur de la sub-MMS ne peut invoquer que la demande de la primitive de service Abort et la réponse de la primitive de service Conclude.

Les seuls événements qui provoquent la sortie de l'état "Sub-MMS Environment" (Environnement Sub-MMS), sont les invocations de la demande de la primitive de service Abort, la demande de la primitive de service Conclude ainsi que la réception des indications de la primitive de service Abort et de la primitive de service Conclude.

Dans l'état "Sub-MMS Environment" (Environnement Sub-MMS), il est possible qu'un utilisateur de la sub-MMS invoque toutes demandes ou réponses des primitives de service précédemment négociées, à condition que les points suivants soient respectés :

a) d'autres sections du présent document restreignant l'utilisation des services par le biais de contraintes d'ordonnancement.

b) il convient qu'une réponse à une primitive ne soit pas invoquée à moins qu'une demande de la primitive correspondante à une indication de service ne soit reçue;

c) il convient de ne pas invoquer la demande de la primitive de service Initiate.

6.2.2.2.3.2.3 Communication dans un environnement sub-MMS prédéfini

L'environnement SUB-MMS est établi implicitement entre deux ou plusieurs utilisateurs. Les ressources allouées pour cet environnement sont réservées pour ses utilisateurs.

Les services de gestion de l'environnement Sub-MMS (Initiate, Conclude, Abort) ne sont pas signifiants.

Si une PDU Request de service Initiate ou Conclude est reçue, une PDU Reject est émise avec le paramètre "Reject PDU type" égal à PDU ERROR et le paramètre "Reject Code" égal à ILLEGAL-MAPPING.

Si une PDU Request d'un service confirmé non pris en charge dans cet environnement est reçue, un message de PDU Reject est émis avec le paramètre "Reject PDU Type" égal à CONFIRMED. REQUEST PDU et Reject Code égaux à UNRECOGNIZED SERVICE.

NOTE 1 Un environnement Sub-MMS établi entre plus de deux utilisateurs ne prend en charge que des services non confirmés.

NOTE 2 La réception d'une Abort.Indication est ignorée.

6.2.2.2.3.2.4 Communication sans environnement Sub-MMS

Aucun environnement Sub-MMS n'est établi entre deux ou plusieurs utilisateurs.

Les ressources allouées à ce type d'émission sont mises à la disposition de tous les utilisateurs distants potentiels.

Les services de gestion de l'environnement Sub-MMS (Initiate, Conclude, Abort) ne sont pas pris en charge.

Si une PDU Request de service Initiate ou Conclude est reçue, une PDU Reject est émise avec le paramètre "Reject PDU Type" égal à PDU ERROR et le paramètre "Reject code" égal à ILLEGAL-MAPPING.

Si une PDU Request d'un service confirmé non pris en charge par l'équipement est reçue, une PDU Reject est émise avec le paramètre "Reject PDU Type" égal à CONFIRMED.REQUEST PDU et le paramètre Reject Code égal à UNRECOGNIZED-SERVICE.

NOTE La réception d'une Abort.Indication est ignorée.

6.2.2.2.3.3 Types d'erreur

6.2.2.3 Spécification de la classe VMD

6.2.2.3.1 Modèle de la classe VMD

FAL ASE:	**VMD ASE**
CLASS: VMD	
CLASS ID:	non utilisé
PARENT CLASS:	non utilisée
Object: VMD	
(m) Key Attribute:	Executive Function
(m) Attribute:	Vendor Name
(m) Attribute:	Model Name
(m) Attribute:	Revision
(m) Attribute:	Logical Status (STATE-CHANGES-ALLOWED,NO-STATE-CHANGES-ALLOWED, OTHER)
(m) Attribute:	Physical Status (OPERATIONAL, PARTIALLY-OPERATIONAL, INOPERABLE, NEEDS-COMMISSIONING)
(m) Attribute:	List of Capabilities
(m) Attribute:	List of Program Invocations
(m) Attribute:	List of Domains
(m) Attribute:	List of Variables
(m) Attribute:	List of Variable Lists
(m) Attribute:	List of Events
(o) Attribute:	Additional detail

6.2.2.3.2 Attributs

Executive function:

Cet attribut représente l'ensemble de logiciels assurant les procédures de services.

Vendor name:

Cet attribut identifie le constructeur de l'appareil.

Model name:

Cet attribut identifie le modèle de l'appareil qui prend en charge le VMD.

Revision:

Cet attribut identifie l'indice de version du système d'appareil prenant en charge le VMD. La valeur de cet attribut est affectée par le vendeur.

Logical status:

La Sub-MMS distingue deux niveaux de fonctions qui sont décrits dans cet attribut. D'autres niveaux des fonctions pourraient être finalement définis sans modifier l'interopérabilité de l'appareil.

1-STATE-CHANGES-ALLOWED:

Tous les services pris en charge par le VMD peuvent être réalisés dans ce statut.

2-NO-STATE-CHANGES-ALLOWED:

Dans ce statut, seuls les services suivants peuvent être réalisés, s'ils sont pris en charge par le VMD::

- Initiate
- Conclude
- Abort
- Identify
- Status
- Read
- Get Alarm summary
- Get Name List
- Get Domain Attributes
- Get Program Invocation Attributes
- Get Variable Access Attributes
- Get Variable List Attributes
- Get Event Condition Attributes

3-OTHER:

D'autres statuts peuvent être définis. Il convient que ces statuts soient décrits dans la PICS.

Physical Status:

Cet attribut donne le statut du matériel de l'appareil

List of capabilities:

Cet attribut permet de définir un ensemble de caractéristiques spécifiques au VMD.

List of program invocations:

Cet attribut donne l'ensemble de programmes d'invocation appartenant au VMD. Les invocations de programme peuvent être prédéfinies dans le VMD ou créées dynamiquement par un service spécifique de la Sub-MMS ou suivant un chargement de domaine.

List of domains:

Cet attribut donne l'ensemble de domaines appartenant au VMD. Les domaines peuvent être prédéfinis dans le VMD ou créés dynamiquement par les services de chargement des domaines SUB-MMS.

List of variables:

Cet attribut donne l'ensemble d'objets de la classe "variable" appartenant au VMD. Les variables peuvent être prédéfinies dans le VMD ou créées dynamiquement suivant un chargement de domaine.

List of variable lists

Cet attribut donne l'ensemble de listes de variables d'objets appartenant au VMD. Les listes de variables peuvent être prédéfinies dans le VMD ou créées dynamiquement par un service spécifique de la Sub-MMS ou suivant un chargement de domaine.

List of events:

Cet attribut donne l'ensemble d'événements appartenant au VMD. Les événements pourraient être prédéfinis dans le VMD ou créés dynamiquement suivant un chargement de domaine.

Additional detail:

Cet attribut peut contenir des informations supplémentaires