ANSI/ISA-S72.02-1993

Approved March 24, 1994

American National Standard

Manufacturing Message Specification: Companion Standard for Process Control

Identical to ISO/IEC 9506-6

ANSI/ISA-S72.02, Manufacturing Message Specification: Companion Standard for Process Control

ISBN: 1-55617-519-1

ISA
67 Alexander Drive
P.O. Box 12277
Research Triangle Park, North Carolina 27709

Preface

This preface is included for information purposes and is not part of ISA S72.02.

This standard has been prepared as part of the service of ISA, the international society for measurement and control, toward a goal of uniformity in the field of instrumentation. To be of real value, this document should not be static, but should be subject to periodic review. Toward this end, the Society welcomes all comments and criticisms, and asks that they be addressed to the Secretary, Standards and Practices Board, ISA, 67 Alexander Drive, P. O. Box 12277, Research Triangle Park, NC 27709, Telephone (919) 990 9227, Fax (919) 549 8288, e-mail: standards@isa.org.

The ISA Standards and Practices Department is aware of the growing need for attention to the metric system of units in general, and the International System of Units (SI) in particular, in the preparation of instrumentation standards. The Department is further aware of the benefits to USA users of ISA Standards of incorporating suitable references to the SI (and the metric system) in their business and professional dealings with other countries. Toward this end, this Department will endeavor to introduce SI-acceptable metric units in all new and revised standards to the greatest extent possible. *The Metric Practice Guide*, which has been published by the Institute of Electrical and Electronic Engineers as ANSI/IEEE Std. 268 1982, and future revisions will be the reference guide for definitions, symbols, abbreviations, and conversion factors.

It is the policy of ISA to encourage and welcome the participation of all concerned individuals and interests in the development of ISA standards. Participation in the ISA standards making process by an individual in no way constitutes endorsement by the employer of that individual, of the Instrument Society of America, or of any of the standards that ISA develops.

This standard has been developed in cooperation with IEC SC65C/WG1, "Message data format for information transferred on process and control data highways." The ISA committee and IEC working group held concurrent meetings and have harmonized the developing drafts throughout the standards process.

The following people served as members of ISA Subcommittee SP72.02:

NAME	COMPANY
C. Williams, Chairman	Eastman Kodak Company
C. Gross, Managing Director	Dow Chemical Company

The following people served as members of ISA Committee SP72:

NAME	COMPANY
C. Williams, Chairman	Eastman Kodak Company
C. Gross, Managing Director	Dow Chemical Company
F. Allegrezza	Moore Products Company

*One vote per company

*One vote per company
**Directors Emeriti

**Directors Emeriti

Contents

Annexes

Figures

Tables

Industrial Automation Systems —

Manufacturing Message Specification —

Part 6: Comparison Standard for Process Control

Foreword

This part of ISO/IEC 9506 was developed by IEC/TC 65C and circulated to the IEC national bodies for approval. It presents new material not previously covered by other standards. Annex A is normative; Annexes B, C and D are informative. This is one of several standards intended as a companion to ISO/IEC 9506, Manufacturing Message Specification (MMS). This part of this Standard deals with control systems as found in the process industries. This part of this Standard should be used when process control systems are connected to a network employing the MMS services and protocol. Together with MMS and its other companion standards, this part of this Standard will enable the networking of different classes of programmable devices on the factory floor.

Introduction

General

This document is a part of a standard developed to facilitate the interconnection of information processing systems. It is positioned within the application layer of the Open Systems Interconnection environment as an application service element (ASE) with respect to other related standards by the basic reference model for open systems interconnection (ISO 7498)

The aim of open systems interconnection is to allow, with a minimum of technical agreement outside the interconnection standards, the interconnection of information processing systems:

a) From different manufacturers

b) Under different managements

c) Of different levels of complexity

d) Of different evolutionary implementations

This part of ISO/IEC 9506 is concerned with the communications and interworking of programmable devices of industrial process control systems utilized in the process industries.

Purpose

The purpose of this part of ISO/IEC 9506 is to augment the use of the Manufacturing Message Specification, ISO/IEC 9506 1 and ISO/IEC 9506 2, for process control applications

This part of ISO/IEC 9506 is a companion standard to the Manufacturing Message Specification (MMS). It also uses and references the Association Control Service Element Definition (ISO 8649) whose provisions it assumes in order to accomplish the aims of the Manufacturing Message Specification. In the Process Control Environment, three forms of communication are recognized; these classes are depicted in Figure 1 and described as follows:

Class A - Communications between a computer and a process control system (PCS), or communications between a computer and a distributed process control system (DCS). The computer performs "higher level" functions which are not part of the PCS or DCS functionality. These functions may include supervisory control, production control and management, remote diagnosis, expert system advice, or any combination of these or other functions not contained within the specific PCS or DCS employed to serve the application. Communications within a DCS may be proprietary to the manufacturer of the DCS, in which case a gateway is required at the DCS to provide communications services and protocols in conformance with this part of ISO/IEC 9506. The gateway functions may reside within a special device at the DCS or may share a DCS device with other functionality. Communications with a gateway may be used to connect PCS systems, or DCS systems, or both where necessary.

Class B - Communications between the component devices of the DCS. The DCS is composed of devices from multiple manufacturers, and communications between devices take place using the services and protocol specified by this part of ISO/IEC 9506.

Class C - Communications between a PCS or a DCS and field devices including sensors, actuators and field multiplexers, as well as communication between field devices.

The application description in Clause 5 of this part of ISO/IEC 9506 focuses primarily on Class A communications. The description may be partially applicable to Class B communications. Interworking of systems conforming to this part of ISO/IEC 9506 and systems supporting Class C communications may be the subject of other standardization efforts, e.g., the International Fieldbus work.

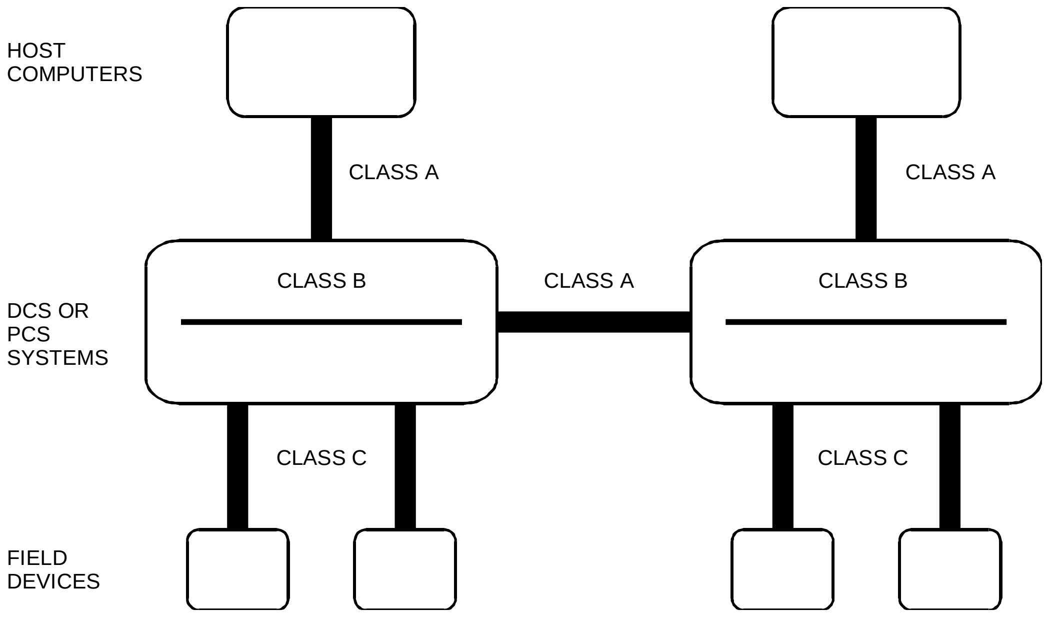

Figure 1 — Classes of communications

This part of ISO/IEC 9506 emphasizes communications in support of supervisory monitoring and control, or class A communications. Specific features of this communication include, but are not limited to:

a) Peer to peer communications between computers of various implementations, used for production control and management, and various process control systems

b) Access to control and monitoring Blocks and their attributes for the purpose of achieving improved control

c) Peer to peer communication with other equipment used in the process control application, such as programmable controllers

Industrial automation systems —

Manufacturing Message Specification —

Part 6: Companion Standard for Process Control

1 Scope

This part of ISO/IEC 9506 describes the use of the Manufacturing Message Specification in the Process Control Environment in terms of:

a) The interaction requirements of process control applications

b) A set of abstract models defining the interaction between process control applications

c) The externally visible functionality of implementations conforming to this part of ISO/ IEC 9506 in the form of procedural requirements associated with the execution of service requests

d) Objects required to be externally visible at implementations conforming to this part of ISO/IEC 9506, in order to support the interactions of process control applications

e) Conformance requirements to this part of ISO/IEC 9506, which include minimum requirements or simple applications as described in Clause 5

This part of ISO/IEC 9506 specifies requirements for the server role, except where specifically noted otherwise.

This part of ISO/IEC 9506 is for use between a supervisory computer and any type of programmable equipment used in process control, including programmable controllers. In addition, it may be used between two instances of programmable equipment.

This part of ISO/IEC 9506 is a companion standard to the Manufacturing Message Specification, ISO/IEC 9506 1 and ISO/IEC 9506 2, designed to support messaging communications to and from programmable devices in an integrated process control application environment. This environment is referred to in this part of ISO/IEC 9506 as the Process Control Environment.

This part of ISO/IEC 9506 specifies a particular mapping of the Process Control Environment onto the Virtual Manufacturing Device (VMD) model and describes the use of the Manufacturing Message Specification services within this environment. This part of ISO/IEC 9506 specifies additions to the services of the Manufacturing Message Specification to support the requirements of the Process Control Environment and specifies additional protocol to support these services.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9506. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9506 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 646:1983	*Information Processing - ISO 7 bit Coded Character Set for Information Interchange*
ISO 7498 1:1984	*Information processing systems - Open systems Interconnection Basic Reference Model*
ISO TR/8509:1987	*Information processing systems - Open systems interconnection Service Conventions*
ISO 8824:1987	*Information processing systems - Open systems interconnection Specification of Abstract Syntax Notation One (ASN.1)*
ISO 8649:1988	*Information processing systems - Open systems interconnection Service definition for the Association Control Service Element*
ISO 8649/AM 1:1990	*Information processing systems - Open systems interconnection Service definition for the Association Control Service Element, Amendment 1 covering authentication during association establishment*
ISO 8650:1988	*Information processing systems - Open systems interconnection Protocol Specification for the Association Control Service Element*
ISO 8650/AM 1:1990	*Information processing systems - Open systems interconnection Protocol Specification for the Association Control Service Element, Amendment 1 covering authentication during association establishment*
ISO/IEC 9506 1:1990	*Industrial automation systems - Manufacturing message specification Service definition*
ISO/IEC 9506 2:1990	*Industrial automation systems - Manufacturing message specification Protocol specification*

3 Definitions

For the purposes of this part of ISO/IEC 9506, the definitions in the following subclauses apply.

3.1 Reference model definitions

Clause 3 of ISO/IEC 9506 1 and ISO/IEC 9506 2 list a number of terms defined in ISO 7498, in ISO TR 8509, and in ISO 8824 as well as its own definitions. These definitions are included in this part of ISO/IEC 9506 by reference.

3.2 Definitions unique to this part of ISO/IEC 9506

For the purpose of this part of ISO/IEC 9506, the following definitions also apply:

3.2.1 batch control computer: A computer used to monitor and control the execution of a product that is manufactured in discrete batches.

3.2.2 cascade structure: A control structure in which the output variable of one controller is the reference variable for one or more secondary control loops.

3.2.3 configuration: The result of customizing a general control system for a particular physical location, application, class of application, or any combination thereof.

3.2.4 control, feedback: Control in which a measured variable is compared to its desired value to produce an actuating error signal that is acted upon in such a way as to reduce the magnitude of the error.

3.2.5 control loop: The collection of I/O equipment, algorithms and other hardware and software modules used to implement feedback control.

3.2.6 control, supervisory: Control in which the control loops operate independently subject to intermittent corrective action, e.g., set point changes from an external source.

3.2.7 control system: A system in which deliberate guidance or manipulation is used to achieve a prescribed value of a variable.

3.2.8 controller: A device that operates automatically to regulate a controlled variable.

3.2.9 controller, PID: A controller that produces proportional plus integral (reset) plus derivative (rate) control.

3.2.10 device: An apparatus for performing a prescribed function.

3.2.11 element: A component of a device or system.

3.2.12 hardware: Physical equipment directly involved in performing industrial process measuring and controlling functions.

3.2.13 historian: A system whose function is to archive records of the actions taken within the process control system.

3.2.14 linking device: A device which provides a gateway function between two dissimilar networks, e.g., between the backbone and the control network, or between the backbone and the Fieldbus.

3.2.15 loop controller: A controller which operates a control loop.

3.2.16 multiloop controller: A controller which operates more than one control loop.

3.2.17 operator's console: A system for the monitoring or operation of a process or both. An operator's console includes the Operator Station object of ISO/IEC 9506 1. It is usually capable of colour graphic displays with live updating data.

3.2.18 parameter: A quantity or property treated as a constant but which may sometimes vary or be adjusted.

3.2.19 process: Physical or chemical change of matter or conversion of energy, e.g., change in pressure, temperature, speed, electrical potential, etc.

3.2.20 process control: The regulation or manipulation of variables influencing the conduct of a process in such a way as to obtain a product of desired quality and quantity in an efficient manner.

3.2.21 process monitoring application: An application that may provide any of the following functions: observe process trends, initiate actions for process optimization, involve the use of expert system technology.

3.2.22 reflexive action: See reflexive processing.

3.2.23 reflexive processing: Action taken without consultation with an operator or supervisory computer, usually conditioned on the occurrence of specified condition or event.

3.2.24 regulatory control: A synonym for feedback control. (See 3.2.4.)

3.2.25 signal conditioning; The act of operating on a signal to make it useful to the device which will utilize the signal. Examples include noise suppression and inversion.

3.2.26 signal inversion: A process in signal conditioning in which the sign of the signal (+, -) is "inverted" to the other sign.

4 Abbreviations

CBB	Conformance Building Block
DCS	Distributed Process Control System
MMS	Manufacturing Message Specification
PDU	Protocol Data Unit
PICS	Proforma Implementation Conformance Statement
PID	Proportional Integral Derivative
PC	Programmable Controller
PCS	Process Control System
VMD	Virtual Manufacturing Device

5 Application description

The interactions between applications in the Process Control Environment take place between communicating peers in support of the exchange of control information or monitoring information or both. During the lifetime of any instance of an application association, a real system may adopt the client role or the server role or both. This part of ISO/IEC 9506 places no restrictions on the behaviour of a client, other than the implied requirement that the system acting in the client role be capable of issuing appropriate requests and receiving appropriate responses. In very simple environments, a single application association may suffice between two communicating peers. In more complex environments, there may be a necessity for more than one application association.

5.1 Process control models

5.1.1 Process manufacturing communication models

5.1.1.1 Peer to peer communication model

Figure 2 illustrates peer to peer communication which may be applied between a cooperating client and server in the Process Control Environment for the purpose of optimizing behaviour in a control system. This model may be applied to communications between a batch control application and a monitoring and control device or between a process monitoring application and a large, distributed process control system, where the distributed process control system is treated as a single device containing numerous monitoring and control elements. This model may be used between two distributed process control systems, each treated as a single device, and may also be applied to interactions between elements of a distributed process control system or between control elements distributed on a control communications highway.

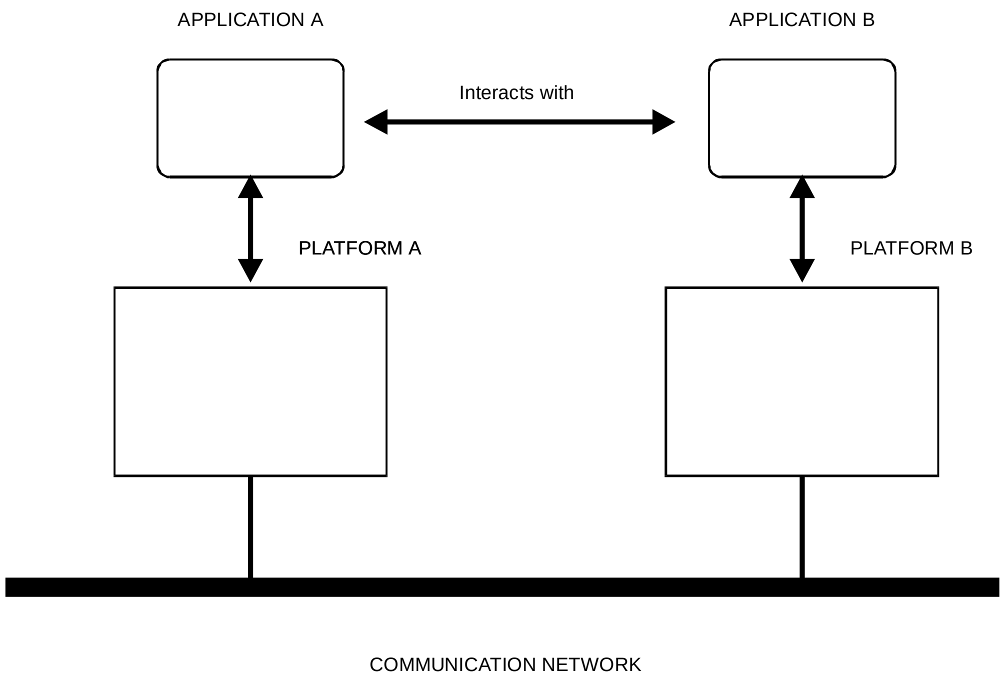

Figure 2 — Interaction in a peer to peer environment

5.1.1.2 Multi-tiered example:

Figure 3 shows an example of a hardware configuration which integrates a distributed process control system with one or more host computers. The distributed process control system's internal network is represented by the lower horizontal line, and its component elements are shown as boxes attached to the internal network. The elements shown are representative of those commonly found in distributed control systems, and include analog and digital input/output controllers, single and multiloop controllers, a programmable controller (PC), batch control computers, a historian, operator workstations and a device (linking device) which performs a linking (potentially a gateway) function between the distributed process control system's internal network and the supervisory network. The supervisory network is represented by the upper horizontal line, and the host computers are attached to the supervisory network.

In this example, a Production Control application operating on a host computer may be used to download batch recipes and retrieve batch product data. The host computer may also communicate with the operator's consoles to setup displays or give operator instructions. The host computer or operator's console may also recall the data from the historian. In this case, the historian may act as a client when collecting data from controllers and as a server when providing data for the host computer or the operator's console. Where application circumstances warrant, some equipment may be provided with connections to both the control network and the supervisory network. One example of such an architecture might include a historian and an operator's console, which may rely on the control network for data acquisition, and may additionally rely on the supervisory network for communications of a supervisory nature.

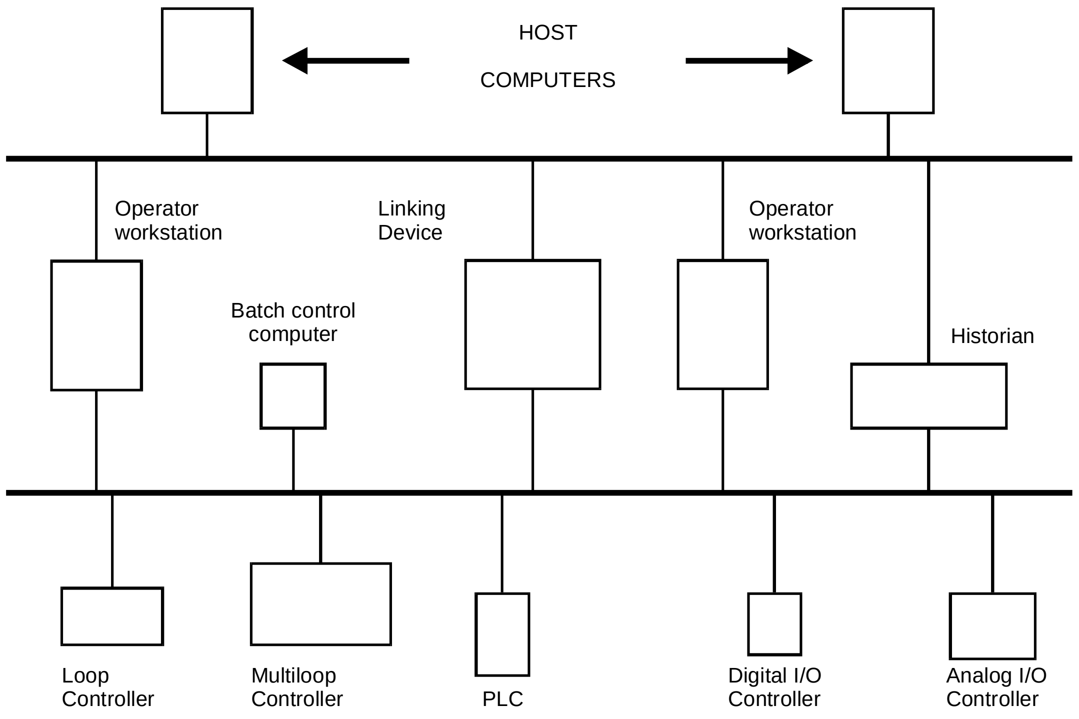

Figure 3 — An example hardware configuration

5.2 Process control functions

5.2.1 Support for regulatory control

5.2.1.1 General

In the Process Control Environment, regulatory control is conceptually accomplished through the utilization of Blocks, such as signal conditioning, mathematical conversions, X Y characterizations and Proportional Integral Derivative (PID) control loops. Blocks represent real instances of functions in a real process control system.

Simple functions may be combined to form complex functions by connecting the output of one Block to the input of another Block. Each Block has associated with it a set of Block parameters which represent the inputs, outputs, and internal state variables required by the function.

5.2.1.2 Protection

Blocks are subject to alteration during operation of the process control system. This alteration may take the form of tuning block parameters for optimal operation. It is a requirement that Blocks be protected against unauthorized modification; in order to accomplish this, it is necessary that the identity and privilege of the requester be determined before a Block is modified.

5.2.1.3 Groups of Blocks

Since, in general, there will be a large number of Block instances employed in a typical process control system, it is a requirement that there be an efficient way of handling groups of Blocks. This includes facilities for downloading and uploading Blocks, starting and stopping groups of Blocks, and otherwise dealing with them in groups.

5.2.2 Management of events and alarms

5.2.2.1 General

There is a functional requirement to continuously monitor processes under control and identify as "process events" the occurrence of certain conditions related to the state of the process under control. Management of a process event implies an ability to communicate the occurrence of the process event to a remote application. There is a functional requirement for the creation of a "notification request," either by or on behalf of a communicating peer, which will result in the creation and transmission of a notification message to the communicating peer containing information about the transition into the specified condition and process state.

When the occurrence of an event is communicated, there is a functional requirement to allow inclusion of textual strings, or identifying numbers, in the message communicating the occurrence. Depending on application considerations, the content of the string or identifying number may be identical for each reported occurrence, or different for each reported occurrence. This information is referred to as "display enhancements."

> **NOTE —** Provision has been included for both numbers and text strings due to the requirement to be able to display the string directly on an operator's console or to use the number to index into an array containing multiple strings in different languages.

Under certain circumstances determined by an application or a configuration process, the response to a process event may include "reflexive actions" which are executed as autonomous actions, as a direct result of detection of the process event. A requirement exists to communicate the results of the reflexive action to communicating peers which have a desire to be aware of the results. As an efficiency move, the results of reflexive actions are communicated with the report of the occurrence of a process event. The creation of a notification request will be used to specify a request for reflexive processing.

Example:

1 - A reflexive action may be a simple act, such as the starting of a Program Invocation, which may lead to several complex actions managed by the system.

Example:

2 - A reflexive action may be an action which must be taken in the interests of personal safety or safety to equipment. The dumping of the contents of a quench tank into an exothermic chemical reaction which is running away constitutes another example.

When process events require acknowledgement by a remote entity, whether a human operator or a supervisory application, the event is termed an "alarm" and there is a functional requirement for related alarm management techniques such as display, recording, and acknowledgement.

5.2.2.2 Event and alarm groupings

In certain process control applications, there is a functional requirement for a capability to group process events into "Process Event Groups," for purposes of status inquiry and alarm management. In most cases, such groups are associated with a unit of equipment or plant, or in some cases represent a grouping of groups, or an instance of a hierarchical relationship of groups. It is a functional requirement that the hierarchical relationships between groups be preserved, while allowing changes of groups at lower levels in the hierarchy to be reflected in higher level groups.

Example:

The addition of a temperature sensor to a feedwater pump group should be automatically included when the points in the boiler support groups are requested.

An additional function requirement is a capability to dynamically override the priority of individual events in process event groups.

> **NOTE —** This capability has utility during process start up and shutdown, as well as facility maintenance, where groups of event and alarm conditions are managed in correspondence to units of affected plant. Priority of individual events is described in ISO 9506 1.

5.2.2.3 Alarm suppression

During maintenance periods, process start up, or abnormal shutdown, there is a functional requirement to suppress notifications of process events which are reporting conditions known to be abnormal. There is a functional requirement for a capability to perform event and alarm suppression on groups of process events, corresponding to affected units of plant, or equipment. When conditions are stable, an additional functional requirement exists to return event and alarm reporting to normal.

5.2.3 Support for batch manufacturing

In many process industries applications, production occurs in "batches," i.e., aggregations of large quantities of material to be made into final products. Typically, one computer, the "batch control computer," will contain the master plan for this batch manufacture, and the process will be carried out under the control of one or more process control computers. In dealing with the interactions between the batch control computer and the process control computers, we can identify five types of requirements.

5.2.3.1 Determination of the status of a batch process

The batch control computer needs to be able to determine the status of the batch process or the status of the components of the batch process as sensed at the process control computer.

5.2.3.2 Control of a batch process

The batch control computer needs to be able to control the batch process by changing the state of the control applications operating in the process control computer.

5.2.3.3 Alteration of the control of a batch process

In some cases, it may be necessary for a human operator present at the process control computer to assume control of the process. There is a need for an orderly way for the batch control computer to relinquish control in such circumstances.

5.2.3.4 Direct communication with a human operator

In some cases, it may be necessary for the batch control computer to display information to a human operator present at the process control computer to solicit his input.

5.2.3.5 Synchronization and coordination

The batch control computer may need to coordinate the use of resources with other processes in the Process Control Environment. Such resources may be a common source of supply of raw materials or a shared device such as an error logging printer.

5.2.4 Support for creation and retrieval of historical records

There is a functional requirement for the retention of information related to a particular process or processes. The information should be available in chronological order when retrieved. Information typically stored in "historical records" includes time stamped records of process events, time stamped records of process variables of interest, either recorded in conjunction with records of process events or independently, and comments entered by a human operator.

5.2.5 Support for process control system configuration and management

A functional requirement exists for the transfer of configuration data from a host computer into an unconfigured as well as into an operational process control system; conversely, a functional requirement exists for a capability to retrieve configuration data for off system storage following debugging. Configuration data includes, but is not limited to, program code, data tables, and code patches.

5.3 Application Models

5.3.1 Block

The Block conceptually represents an installed instance of a control or monitoring function in a process control system. The basic functional aspects of a Block necessary for prescribing communication behaviour are described here. Further information about Blocks and their use in the Process Industry may be found in Annex B.

Example:

Blocks may be utilized to perform signal conditioning, mathematical or complex monitoring, and control functions.

```
Model: Block
    Key Attribute: Block Tag
    Attribute: InService (TRUE, FALSE)
    Attribute: Algorithm reference
    Attribute: Mode State
    Attribute: List of Block Parameters
```

5.3.1.1 Block Tag

This attribute is the name of the Block. It is unique throughout the process control system. It is assigned by the end user or configuration manager of the process control system.

5.3.1.2 InService

This attribute indicates whether (true) or not (false) the Block is actively operating.

5.3.1.3 Algorithm Reference

This attribute contains a reference to the function algorithm. There is a reference for each Block in a real control system; however, use of a common algorithm for multiple Blocks is not precluded so long as individual references from each Block are maintained.

5.3.1.4 Mode State

This attribute governs the operation of the Block and the protection of its elements. Further information can be found in Annex B.

5.3.1.5 List of Block Parameters

This attribute is the list of Block parameters which represents the inputs, outputs, and internal state variables of the Block.

5.3.2 Algorithm

The Algorithm object represents the algorithm used by a Block instance to transform its inputs into its outputs. It is a separate object to reflect the fact that many Block instances may share a common algorithm.

```
Object: Algorithm
    Key Attribute: Algorithm Name
    Attribute: Algorithm content
```

5.3.2.1 Algorithm Name

This attribute is the name associated with the algorithm.

5.3.2.2 Algorithm content

This attribute represents the substance of the algorithm, the program or rules by which the output is determined based on the input, and the internal state.

6 Process control context mapping

This clause prescribes how the facilities of MMS shall be used to satisfy the communication requirements of the Process Control Environment described in the previous clause. This clause also relates the models of process control systems to the abstract model of the Virtual Manufacturing Device (VMD) described in ISO/IEC 9506 1.

6.1 Mapping the process control model to the VMD

Each device of a process control system that offers Class A communication capabilities as a server shall be mapped onto at least one VMD.

Example:

A programmable controller engaged in a process control application may consist of a single VMD. A large distributed process control system, when viewed through a gateway from a production control computer, may also consist of a single VMD. A distributed process control system, when viewed from an internal element of the control system, such as an operator's console, may contain multiple VMDs with at least one VMD representing each station connected to the distributed process control system. A large distributed process control system with multiple connections to the supervisory network from the system gateway, the historian, and operator's console may appear, from the supervisory network, as multiple VMDs.

6.2 Definition of process control objects that map to Domains

6.2.1 The Block object

The Block object shall map to a Domain and to a Program Invocation bound to this Domain. The Domain shall contain Domain specific Named Variable objects that shall represent the inputs, outputs, and internal state parameters of the Block. The Mode State attribute of the Block shall also be represented as a Named Variable object of the Domain. If the syntax of the Block Tag is compatible with the syntax of an Identifier (see 7.6.2 of ISO/IEC 9506 2), the name of the Domain shall be the same as the Block Tag. If the syntax of the Block Tag is not compatible with Identifier syntax, there are no requirements on the Domain name. However, Annex B.4 provides guidance on a recommended method for deriving a Domain name from the Block Tag.

6.2.2 The Algorithm object

The Algorithm object shall map to a Domain. The Domain name shall be the name of the Algorithm.

6.3 Definition of process control objects that map to Program Invocations

6.3.1 The Block object

The Block object shall map to a Domain and to a Program Invocation bound to this Domain. If the syntax of the Block Tag is compatible with the syntax of an Identifier (see 7.6.2 of ISO/IEC 9506 2), the name of the Program Invocation shall be the same as the Block Tag. If the syntax of the Block Tag is not compatible with Identifier syntax, there are no requirements on the Program

Invocation name. However, Annex B.4 provides guidance on a recommended method for deriving a Program Invocation name from the Block Tag.

The Program Invocation representing the Block will normally have two Domains bound to it, one whose name is the same as the Program Invocation supports the Named Variables that represent the inputs, outputs, internal state variables, and Mode State of the Block. The other Domain, which may be sharable, represents the algorithm for the Block execution. Depending on the nature of the Block, there may be additional Domains bound to the Program Invocation. In some cases, the entire Block may be represented by a single Domain bound to a Program Invocation.

The state of the Program Invocation is used in this model to represent the InService attribute of the Block. When the Program Invocation is in the IDLE or STOPPED state, the InService attribute is FALSE. When the Program Invocation is in the RUNNING state, the InService attribute is TRUE.

6.4 Process control requirements that affect other MMS objects

6.4.1 Extensions to the Event Condition object

The Event Condition object in Clause 15 of ISO/IEC 9506 1 is extended by the following attributes:

```
Object: Event Condition
  All MMS defined attributes
  Attribute:  Display Enhancement Class (TEXT, NUMBER, UNDEFINED)
    Constraint:  Display Enhancement Class  = TEXT
  Attribute:  Display Enhancement (String)
    Constraint:  Display Enhancement Class  = NUMBER
  Attribute:  Display Enhancement (Integer)
  Attribute:  Group Priority Override
  Attribute:  List of referencing Event Condition List references
```

6.4.1.1 Display Enhancement Class

This attribute identifies the type of the Display Enhancement attribute of the Event Condition object. If the value of this attribute is TEXT, the Display Enhancement is of type character string. If the value of this attribute is NUMBER, the Display Enhancement is of type integer. If the value of this attribute is UNDEFINED, the Display Enhancement attribute is not present.

6.4.1.2 Display Enhancement

This attribute is used to contain information useful in preparing operator displays at the MMS user receiving notification of the event. The type of this attribute is either character string or integer, depending on the value of the Display Enhancement Class attribute.

6.4.1.3 Group Priority Override

This attribute shall contain a value that is either UNDEFINED or an integer value between zero (0) and one hundred twenty-seven (127). Zero shall represent the highest priority and one hundred twenty-seven shall represent the lowest priority. If the value of this attribute is defined, it shall represent an override priority value that shall be utilized by the VMD in place of the value contained in the Priority attribute. If the value of the Group Priority Override attribute is UNDEFINED, the VMD shall utilize the value of the Priority attribute in determining the importance of the Event Condition object. When an Event Condition object is created using the Define Event Condition service, the value of this attribute shall be initialized to UNDEFINED.

Example:

Consider the case in which an Event Condition object's priority is modified using a service operating on an Event Condition List, and then individually altered using the AlterEventConditionMonitoring service, and then modified again at the group priority level using a service operating on an Event Condition List. Following the first change, the Group Priority Override attribute has changed and takes precedence over the Priority attribute. The Priority attribute has not changed, but is not considered since there is a group override in effect. The Priority attribute does change following the Priority change, but is still not considered until the group override is removed.

The Group Priority Override attribute imposes an absolute override, whether the value is higher or lower than the value of the Priority attribute.

This attribute need not be implemented if Event Condition List objects are not supported.

6.4.1.4 List of referencing Event Condition List references

This attribute shall contain a list of references to Event Condition List objects that reference the specific Event Condition object. The Event Condition List object is defined in of this part of ISO/ IEC 9506. When an Event Condition object is created using the DefineEventCondition service, the value of this attribute shall be initialized to an empty list.

This attribute need not be implemented if Event Condition List objects are not supported.

6.4.2 Extensions to Event Enrollment objects

The Event Enrollment object in described in Clause 15 of ISO/IEC 9506 1 is extended as follows:

```
Object: Event Enrollment
  All MMS defined attributes
  Attribute:  Display Enhancement Class (TEXT, NUMBER, UNDEFINED)
    Constraint:  Display Enhancement Class  = TEXT
  Attribute:  Display Enhancement (String)
    Constraint:  Display Enhancement Class  = NUMBER
  Attribute:  Display Enhancement (Integer)
```

6.4.2.1 Display Enhancement Class

This attribute identifies the type of the Display Enhancement attribute of the Event Enrollment object. If the value of this attribute is TEXT, the Display Enhancement is of type character string. If the value of this attribute is NUMBER, the Display Enhancement is of type integer. If the value of this attribute is UNDEFINED, the Display Enhancement attribute is not present.

6.4.2.2 Display Enhancement

This attribute is used to contain information useful in preparing operator displays at the MMS user receiving notification of the event. The type of this attribute is either character string or integer, depending on the value of the Display Enhancement Class attribute.

6.5 Definitions of new MMS abstract objects

6.5.1 Event Condition List object

The Event Condition List object shall be used to reference groups of Event Condition objects that are required to be operated on as groups. Support of the RECL (See 6.6.3) CBB indicates support for Event Condition Lists that may refer to other Event Condition Lists. If this CBB is not supported, the list may contain only references to Event Conditions.

```
Object:  Event Condition List
  Key Attribute: Event Condition List name
  Attribute: MMS Deletable
  Attribute: List of Event Condition references
  Attribute: List of Event Condition List references
  Attribute: List of referencing Event Condition List references
```

6.5.1.1 Event Condition List name

This attribute, of type Object Name, shall be the name by which the Event Condition List is identified. An Event Condition List name may have VMD, Domain specific, or AA specific scope.

6.5.1.2 MMS deletable

This attribute, of type Boolean, shall indicate whether (true) or not (false). This object may be deleted through the use of the service defined in 7.3.15 of this part of ISO/IEC 9506.

6.5.1.3 List of Event Condition references

This attribute shall contain a list of references to Event Condition objects. The scope of services operating on the Event Condition List object shall include objects in both the List of Event Condition references attribute and the List of Event Condition List references attribute.

Because of visibility constraints, if the Event Condition List name has VMD specific or Domain specific scope, this attribute shall contain references only to Event Condition objects which have VMD specific or Domain specific scope. If the Event Condition List name attribute has AA specific scope, this attribute may reference Event Condition objects of any scope.

6.5.1.4 List of Event Condition List references

This attribute shall contain a list of references to Event Condition List objects that are hierarchically subordinate to this Event Condition List object. It shall not be permissible for this attribute, nor for this attribute in subordinate Event Condition List objects, to contain circular references. The scope of services operating on the Event Condition List object shall include objects included in both the List of Event Condition references attribute and the List of Event Condition List references attribute. The List of Event Condition List references attribute shall be present only if the RECL CBB has been negotiated.

Because of visibility constraints, if the Event Condition List name has VMD specific or Domain specific scope, this attribute shall contain references only to Event Condition List objects which have VMD specific or Domain specific scope. If the Event Condition List name attribute has AA specific scope, this attribute may reference Event Condition List objects of any scope.

6.5.1.5 List of referencing Event Condition List references

This attribute shall contain a list of references to other Event Condition List objects that reference this specific Event Condition List object. If there are no references to this Event Condition List object, the value of this attribute shall be an empty list. This attribute shall be present only if the RECL CBB has been negotiated.

> **NOTE —** This attribute is necessary to fully describe the service procedures that operate on the Event Condition List object. This attribute is not visible nor modifiable via the services provided in this part of ISO/IEC 9506.

6.5.2 Unit Control object

The Unit Control object shall represent a collection of MMS objects representing groups of Domains and/or Program Invocations that may be loaded and managed as a collective unit. Downloads and uploads may be performed on a Unit Control object to efficiently transmit the information representing multiple Domains to and from a process control system in one or more download or upload sequences. While introduced to satisfy the requirement of manipulating groups of Blocks, the Unit Control object may be used to manipulate groups of Domains and/or Program Invocations. Services that operate on the Unit Control object are described in 7.3.1 through 7.3.13.

> NOTE – The Unit Control object may be utilized to minimize the number of PDU's necessary to download large numbers of Domains or to create large numbers of Program Invocations. Rules concerning the applicability of grouping of these objects is outside the scope of this part of ISO/IEC 9506.

```
Object: Unit Control
  Key Attribute: Unit Control Name
  Attribute: MMS Deletable (TRUE, FALSE)
  Attribute: List of Domain references
  Attribute: List of Program Invocation references
```

6.5.2.1 Unit Control Name

This attribute shall uniquely identify the Unit Control object at the VMD. The name scope of the Unit Control object shall be VMD specific.

6.5.2.2 MMS Deletable

This attribute shall indicate whether (true) or not (false) the Unit Control object may be deleted using the DeleteUnitControl service.

6.5.2.3 List of Domain references

This attribute shall identify the Domain objects that are constituents of the Unit Control object and that may be affected by operations on the Unit Control object.

6.5.2.4 List of Program Invocation references

This attribute shall identify the Program Invocation objects that are constituents of the Unit Control object and may be affected by operations on the Unit Control object.

6.6 Parameter conformance

This part of ISO/IEC 9506 introduces three new parameter conformance building blocks (CBB's) in the system specification.

6.6.1 DES

The DES parameter conformance building block shall establish the validity of the text form of the display enhancement parameter, whenever it occurs in a request or indication in a service table.

If DES is supported, the parameter is valid. Otherwise, the parameter is invalid. If DES is not supported, a request specifying this parameter shall constitute a protocol error.

If DES is supported, DEI shall not be supported.

6.6.2 DEI

The DEI parameter conformance building block shall establish the validity of the numeric form of the display enhancement parameter, whenever it occurs in a request or indication in a service table.

If DEI is supported, the parameter is valid. Otherwise, the parameter is invalid. If DEI is not supported, a request specifying this parameter shall constitute a protocol error.

If DEI is supported, DES shall not be supported.

6.6.3 RECL

The RECL parameter conformance building block shall establish the validity of the List of Event Condition List parameter, whenever it occurs in a service table.

If RECL is supported, this parameter is valid. Otherwise, the parameter is invalid. A request or response specifying this parameter shall constitute a protocol error. If RECL is not supported, a request which, if honoured, would require a response specifying this parameter shall result in a service error specifying error class equal to ACCESS and error code equal to OBJECT ACCESS UNSUPPORTED.

7 Services

7.1 Use of ACSE services

In order to support certain MMS operations that affect the state of the process control system, the MMS server shall require that the MMS client has provided the AP title in the A ASSOCIATE service and further that the MMS client has specified the Authentication functional unit of ACSE and provided proper values of Privilege Classification and Privilege Identifier as part of the Authentication Value of the A ASSOCIATE service. These values shall become values of the corresponding attributes of the Application Association object (See Annex A). If the MMS client has not supplied such values, the corresponding attributes shall be set to a value of UNDEFINED or the MMS server may, at its option, refuse establishment of the association.

For the purpose of providing a syntax definition for the Authentication mechanism, this part of ISO/IEC 9506 assigns the ASN.1 object identifier value and the associated ASN.1 module for the external choice of Authentication value.

```
ISO 9506 Authentication { iso standard 9506 part(6)
authentication(3) }
DEFINITIONS ::= BEGIN

Authentication syntax ::= SEQUENCE {
  user                        VisibleString,
  password                    OCTET STRING OPTIONAL,
  privilege classification    INTEGER OPTIONAL
  }
END
```

7.2 Use of MMS services

This subclause specifies the use of services and protocol defined in ISO/IEC 9506 1 and ISO/IEC 9506 2, extensions to some of those services, and protocol to support the extensions.

7.2.1 Process application context name

For the purpose of being able to use an application that only contains the ACSE and MMS as ASEs, this part of ISO/IEC 9506 uses the object identifier value and the object descriptor value defined in 17.12 of ISO/IEC 9506 2.

7.2.2 Process control abstract syntax definition

This part of ISO/IEC 9506 assigns the ASN.1 object identifier value

```
{iso standard 9506 part (6) mms process syntax version(1)}
```

to the abstract syntax defined in this clause.

7.2.3 Process control specific ASN.1 module definition

The MMS services and protocol were developed to be used by a wide range of manufacturing devices. This section defines the process control services and protocol for those elements identified as requiring companion standard definition in MMS. These definitions should be used when the abstract syntax defined in this part of ISO/IEC 9506 is negotiated.

All ASN.1 definitions provided in this part of ISO/IEC 9506 are part of the ASN.1 Module "ISO 9506 MMS PROCESS 1." The beginning and closing statements indicating that each ASN.1 definition provided is a part of this module is omitted in order to make reading of the document easier. Each ASN.1 definition provided implicitly contains the statement:

```
ISO 9506 MMS PROCESS 1 {iso standard 9506 part(6) mms-process-module-
version1(2)}

DEFINITIONS ::= BEGIN
```

at the beginning of the definition and contains the keyword "END" at the end of the definition.

> **NOTE —** ISO 9506 MMS PROCESS 1 represents version number 1 of the MMS (ISO 9506) Companion Standard for Process Control.

Many of the terms and abbreviations used in this clause use the terminology of MMS service and protocol descriptions (See Clause 5 of ISO/IEC 9506 1 and Clause 5 of ISO/IEC 9506 2). In particular, refer to Clause 5 of ISO/IEC 9506 1 for general rules on how to interpret the service tables in this part of ISO/IEC 9506.

```
IMPORTS MMSpdu,
  ParameterSupportOptions,
  ServiceSupportOptions,
  Integer16,
  StatusResponse,
  Identifier,
  ObjectName,
  ProgramInvocationState,
  FileName,
  ApplicationReference,
  Priority,
  EventTime,
  EC State
FROM MMS General Module 1
  {iso standard 9506 part(2)  mms-general-module-version1(2)};
```

7.2.4 VMD support services

7.2.4.1 GetNameList service extensions

The GetNameList service shall be extended to include the Event Condition List object and the Unit Control object.

7.2.4.1.1 CS Object Class service parameter

This parameter, of type integer, shall identify the Event Condition List object and the Unit Control object. The value zero (0) shall represent the Event Condition List object, and the value one (1) shall represent the Unit Control object.

7.2.4.1.2 Protocol extensions

The abstract syntax of the CsAdditionalObjectClasses parameter shall be specified as follows:

```
CsAdditionalObjectClasses ::=  IMPLICIT INTEGER {
  eventConditionList      (0),
  unitControl             (1)
  }
```

7.2.5 Event management services

This subclause specifies the extension of the Event Management services.

7.2.5.1 DefineEventCondition service

This subclause specifies the extension of the DefineEventCondition service.

7.2.5.1.1 Parameter extensions

The DefineEventCondition service shall be extended to include specification of the value of the Display Enhancement attribute of the Event Condition object. The structure of the DefineEventCondition parameter extensions is specified in Table 1.

Table 1 — DefineEventCondition parameter extensions

Parameter name	Req	Ind	Rsp	Cnf	CBB
Display Enhancement	M	M(=)			
Display Enhancement string	S	S(=)			DES
Display Enhancement index	S	S(=)			DEI
No Enhancement	S	S(=)			

7.2.5.1.1.1 Display Enhancement

Selection of this parameter indicates that the Display Enhancement attribute of the Event Condition shall be altered by this service. If this parameter is selected, one of the following parameters shall appear.

7.2.5.1.1.1.1 Display Enhancement string

This parameter, of type character string, is the string form of the Display Enhancement parameter. This selection may be made only if the DES CBB has been negotiated.

7.2.5.1.1.1.2 Display Enhancement index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter. This selection may be made only if the DEI CBB has been negotiated.

7.2.5.1.1.1.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present. This parameter shall be selected if neither DES nor DEI has been negotiated.

7.2.5.1.1.2 Service procedure extensions

The service procedure of 15.2.2 of ISO/IEC 9506 1 shall be performed. If the Display Enhancement string has been selected, the value of the Display Enhancement Class attribute of the Event Condition shall be TEXT and the Display Enhancement attribute shall be set to the value of the Display Enhancement string parameter. If the Display Enhancement index is selected, the Display Enhancement Class attribute of the Event Condition shall be NUMBER and the Display Enhancement attribute shall be set to the value of the Display Enhancement index parameter. Otherwise the Display Enhancement Class attribute shall be set to UNDEFINED.

7.2.5.1.1.3 Protocol extensions

The extensions to the DefineEventCondition service argument shall be the CS DefineEventCondition Request.

```
CS DefineEventCondition Request ::=  [0] Choice {
  enhancementString      [0] IMPLICIT VisibleString,
  enhancementIndex       [1] IMPLICIT INTEGER,
  noEnhancement          NULL
  }
```

NOTE — As a result of the way in which Confirmed RequestPDU is specified in ISO/IEC 9506 2, a NULL specified as a Companion Standard extension is not transmitted. The effect is as though the parameter were not included in the protocol. Specification of the NULL is required at the service interface, however.

7.2.5.2 DeleteEventCondition service extensions

The service procedure of the DeleteEventCondition service shall be extended to include a verification that the value of the List of referencing Event Condition List references attribute of the specified Event Condition object is equal to an empty list. The verification shall be performed prior to performing the procedure specified in ISO/IEC 9506 1. If the value of this attribute is not equal to an empty list, the specified Event Condition object shall be counted as a Candidate Not Deleted, the Event Condition shall not be deleted, and a Response(-) shall be returned.

7.2.5.3 GetEventConditionAttributes service

This subclause specifies the extension of the GetEventConditionAttributes service.

7.2.5.3.1 Parameter extensions

The GetEventConditionAttributes service response shall be extended to include the Group Priority Override parameter, the List of referencing Event Condition Lists parameter and the Display Enhancement parameter. The structure of the GetEventConditionAttributes parameter extensions is specified in Table 2.

Table 2 — GetEventConditionAttributes parameter extensions

Parameter name	Req	Ind	Rsp	Cnf	CBB
Group Priority Override			C	C (=)	
List of referencing Event Condition Lists			C	C (=)	
Display Enhancement			M	M (=)	
Display Enhancement string			S	S (=)	
Display Enhancement index			S	S (=)	
No Enhancement			S	S (=)	

7.2.5.3.2 Group priority override

This parameter, of type integer, shall contain the value of the Group Priority Override attribute of the Event Condition object.

7.2.5.3.3 List of referencing Event Condition Lists

This parameter shall contain a list of names, derived from the contents of the List of referencing Event Condition List references attribute of the specified Event Condition object. Each name shall be equal to the value of the name attribute of a referencing Event Condition List object.

7.2.5.3.4 Display Enhancement

This parameter indicates the value of the Display Enhancement Class attribute of the Event Condition object. Depending on its value, one of the following parameters shall be selected.

7.2.5.3.4.0.1 Display Enhancement string

This parameter, of type character string, is the string form of the Display Enhancement parameter.

7.2.5.3.4.0.2 Display Enhancement index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter.

7.2.5.3.4.0.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present.

7.2.5.3.5 Service procedure extensions

The service procedure of 15.4.2 of ISO/IEC 9506 1 shall be performed. If the Display Enhancement Class attribute of the Event Condition has the value TEXT, the Display Enhancement string parameter shall be selected. If the Display Enhancement Class attribute has the value NUMBER, the Display Enhancement index shall be selected. If the Display Enhancement Class attribute has the value UNDEFINED, the No Enhancement parameter shall be selected.

The service procedure of the GetEventConditionAttributes service shall be extended to include the value of the Group Priority Override attribute of the Event Condition object as the Group Priority Override parameter and the value of the List of referencing Event Condition List references attribute as the value of the List of referencing Event Condition Lists parameter. These parameters shall only appear if and only if one or more of the services for Event Condition List objects are supported. The value of the Display Enhancement attribute of the Event Condition object shall be included as the Display Enhancement parameter.

7.2.5.3.6 Protocol extensions

The extensions to the GetEventConditionAttributes service response protocol shall be the CS GetEventConditionAttributes Response.

```
CS GetEventConditionAttributes Response ::= CHOICE {
  supplied           SEQUENCE {
    groupPriorityOverride     [0] CHOICE {
       priority               [0] IMPLICIT Priority,
       undefined              [1] IMPLICIT NULL
       } OPTIONAL,
    listOfEventConditionList  [1] IMPLICIT SEQUENCE OF ObjectName
                              OPTIONAL,
    displayEnhancement        [2] CHOICE {
      enhancementString        [0] IMPLICIT VisibleString,
      enhancementIndex         [1] IMPLICIT INTEGER,
      noEnhancement            [2] IMPLICIT NULL
    } },
  default              NULL
  }
```

The default choice shall be chosen if and only if the groupPriorityOverride is not transmitted, the listOfEventConditionList field is not transmitted, and the noEnhancement choice is selected for the display Enhancement field.

7.2.5.3.7 GroupPriorityOverride

The priority choice shall be selected when the value of the Group Priority Override parameter of the GetEventConditionAttributes service response is not equal to UNDEFINED; otherwise, the undefined choice shall be selected.

7.2.5.3.8 Display Enhancement

If the value of the Display Enhancement Class attribute of the Event Condition object is equal to TEXT, the enhancementString choice shall be selected. If the value of the Display Enhancement Class attribute of the Event Condition object is equal to NUMBER, the enhancementIndex choice shall be selected. Otherwise the noEnhancement choice shall be selected.

7.2.5.4 AlterEventConditionMonitoring service

This subclause specifies the extensions to the AlterEventConditionMonitoring service.

7.2.5.4.1 Parameter extensions

The AlterEventConditionMonitoring service shall be extended to include specification of the value of the Display Enhancement attribute of the Event Condition object. The structure of the AlterEventConditionMonitoring parameter extensions is specified in Table 3.

Table 3 — AlterEventConditionMonitoring parameter extensions

Parameter name	Req	Ind	Rsp	Cnf	CBB
Display Enhancement	S	S(=)			
Display Enhancement string	S	S(=)			DES
Display Enhancement index	S	S(=)			DEI
No Enhancement	S	S(=)			
Unchanged Display	S	S(=)			

7.2.5.4.1.1 Display Enhancement

Selection of this parameter indicates that the Display Enhancement attribute of the Event Condition shall be altered by this service. If this parameter is selected, one of the following parameters shall appear.

7.2.5.4.1.1.1 Display Enhancement string

This parameter, of type character string, is the string form of the Display Enhancement parameter. This selection may be made only if the DES CBB has been negotiated.

7.2.5.4.1.1.2 Display Enhancement index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter. This selection may be made only if the DEI CBB has been negotiated.

7.2.5.4.1.1.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present. This parameter shall be selected if neither DES nor DEI has been negotiated.

7.2.5.4.1.2 Unchanged Display

If this parameter is selected, the Display Enhancement attribute of the Event Condition shall not be changed from its present value.

7.2.5.4.2 Service procedure extensions

If the Display Enhancement selection is made, the value of the Display Enhancement attribute of the Event Condition object shall be altered. If the Display Enhancement string is selected, the Display Enhancement Class attribute of the Event Condition shall be set to TEXT and the Display Enhancement attribute shall be set to the value of the Display Enhancement string parameter. If the Display Enhancement index is selected, the Display Enhancement Class attribute of the Event Condition shall be set to NUMBER and the Display Enhancement attribute shall be set to the value of the Display Enhancement index parameter. Otherwise, the Display Enhancement Class shall be set to UNDEFINED.

7.2.5.4.3 Protocol extensions

The extension to the AlterEventConditionMonitoring service argument shall be the CS AlterEventConditionMonitoring Request.

```
CS AlterEventConditionMonitoring Request ::= SEQUENCE {
  changeDisplay           CHOICE {
    enhancementString        [0] IMPLICIT VisibleString,
    enhancementIndex         [1] IMPLICIT INTEGER,
    noEnhancement            [2] NULL
    } OPTIONAL }
```

The choice of Display Enhancement for Display Option shall be indicated by the inclusion of the changeDisplay field in the PDU. If Unchanged Display is chosen, the changeDisplay field shall not occur.

7.2.5.5 DefineEventEnrollment service

This subclause specifies the extensions to the DefineEventEnrollment service.

7.2.5.5.1 Parameter extensions

The DefineEventEnrollment service shall be extended to include specification of the Display Enhancement attribute of the Event Enrollment object. The structure of the DefineEventEnrollment parameter extensions is specified in Table 4.

Table 4 — DefineEventEnrollment parameter extensions

Parameter name	Req	Ind	Rsp	Cnf	CBB
Display Enhancement	M	M(=)			
Display Enhancement string	S	S(=)			DES
Display Enhancement index	S	S(=)			DEI
No Enhancement	S	S(=)			

7.2.5.5.1.1 Display Enhancement

The value of this parameter indicates the type of Display Enhancement desired. Depending on its value, one of the following parameters shall be selected.

7.2.5.5.1.1.1 Display Enhancement string

This parameter, of type character string, is the string form of the Display Enhancement parameter. This selection may be made only if the DES CBB has been negotiated.

7.2.5.5.1.1.2 Display Enhancement index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter. This selection may be made only if the DEI CBB has been negotiated.

7.2.5.5.1.1.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present. This parameter shall be selected if neither DES nor DEI has been negotiated.

7.2.5.5.2 Service procedure extensions

The service procedure of 15.12.2 of ISO/IEC 9506 1 shall be performed. If the Display Enhancement string has been selected, the value of the Display Enhancement Class attribute of the Event Enrollment shall be TEXT and the Display Enhancement attribute shall be set to the value of the Display Enhancement string parameter. If the Display Enhancement index is selected, the Display Enhancement Class attribute of the Event Enrollment shall be NUMBER and the Display Enhancement attribute shall be set to the value of the Display Enhancement index parameter. Otherwise the Display Enhancement Class attribute shall be set to UNDEFINED.

7.2.5.5.3 Protocol extensions

The extensions to the DefineEventEnrollment service argument shall be the CS DefineEventEnrollment Request.

```
CS DefineEventEnrollment Request  ::=  [0] Choice {
  enhancementString      [0] IMPLICIT VisibleString,
  enhancementIndex       [1] IMPLICIT INTEGER,
  noEnhancement          NULL
  }
```

NOTE — As a result of the way in which Confirmed RequestPDU is specified in ISO/IEC 9506 2, a NULL specified as a Companion Standard extension is not transmitted. The effect is as though the parameter were not included in the protocol. Specification of the NULL is required at the service interface, however.

7.2.5.6 GetEventEnrollmentAttributes service

This subclause specifies the extension of the GetEventEnrollmentAttributes service.

7.2.5.6.1 Parameter extensions

The GetEventEnrollmentAttributes service response shall be extended to include the Display Enhancement parameter. The structure of the GetEventEnrollmentAttributes parameter extensions is specified in Table 5.

Table 5 — GetEventEnrollmentAttributes parameter extensions

Parameter name	Req	Ind	Rsp	Cnf	CBB
Display Enhancement			M	M(=)	
Display Enhancement string			S	S(=)	
Display Enhancement index			S	S(=)	
No Enhancement			S	S(=)	

7.2.5.6.1.1 Display Enhancement

The value of this parameter indicates the type of Display Enhancement desired. Depending on its value, one of the following parameters shall be selected.

7.2.5.6.1.1.1 Display Enhancement string

This parameter, of type character string, is the string form of the Display Enhancement parameter.

7.2.5.6.1.1.2 Display Enhancement index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter.

7.2.5.6.1.1.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present.

7.2.5.6.2 Service procedure extensions

The service procedure of 15.14.2 of ISO/IEC 9506 1 shall be performed. If the Display Enhancement Class attribute of the Event Enrollment has the value TEXT, the Display Enhancement string parameter shall be selected. If the Display Enhancement Class attribute has the value NUMBER, the Display Enhancement index shall be selected. If the Display Enhancement Class attribute has the value UNDEFINED, the No Enhancement parameter shall be selected.

7.2.5.6.3 Protocol extensions

The extensions to the GetEventEnrollmentAttributes service response protocol shall be the CS GetEventEnrollmentAttributes Response.

```
CS GetEventEnrollmentAttributes Response ::= [0] Choice {
  enhancementString      [0] IMPLICIT VisibleString,
  enhancementIndex       [1] IMPLICIT INTEGER,
  noEnhancement          NULL
  }
```

NOTE — As a result of the way in which Confirmed ResponsePDU is specified in ISO/IEC 9506 2, a NULL specified as a Companion Standard extension is not transmitted. The effect is as though the parameter were not included in the protocol. Specification of the NULL is required at the service interface, however.

7.2.5.7 AlterEventEnrollment service

This subclause specifies the extension of the AlterEventEnrollment service.

7.2.5.7.1 Parameter extensions

The AlterEventEnrollment service shall be extended to include specification of the Display Enhancement attribute of the Event Enrollment object. The structure of the AlterEventEnrollment parameter extensions is specified in Table 6.

Table 6 — AlterEventEnrollment parameter extensions

Parameter name	Req	Ind	Rsp	Cnf	CBB
Display Enhancement	S	S(=)			
Display Enhancement string	S	S(=)			DES
Display Enhancement index	S	S(=)			DEI
No Enhancement	S	S(=)			
Unchanged Display	S	S(=)			

7.2.5.7.1.1 Display Enhancement

Selection of this parameter indicates that the Display Enhancement attribute of the Event Condition shall be altered by this service. If this parameter is selected, one of the following parameters shall appear.

7.2.5.7.1.1.1 Display Enhancement string

This parameter, of type character string, is the string form of the Display Enhancement parameter. This selection may be made only if the DES CBB has been negotiated.

7.2.5.7.1.1.2 Display Enhancement index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter. This selection may be made only if the DEI CBB has been negotiated.

7.2.5.7.1.1.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present. This parameter shall be selected if neither DES nor DEI has been negotiated.

7.2.5.7.1.2 Unchanged Display

If the parameter is selected, the Display Enhancement attribute of the Event Enrollment shall not be changed from its present value.

7.2.5.7.2 Service procedure extensions

If the Display Enhancement selection is made, the value of the Display Enhancement attribute of the Event Enrollment object shall be altered. If the Display Enhancement string is selected, the Display Enhancement Class attribute of the Event Enrollment shall be set to TEXT and the Display Enhancement attribute shall be set to the value of the Display Enhancement string parameter. If the Display Enhancement index is selected, the Display Enhancement Class attribute of the Event Enrollment shall be set to NUMBER and the Display Enhancement attribute shall be set to the value of the Display Enhancement index parameter. Otherwise, the Display Enhancement Class shall be set to UNDEFINED.

7.2.5.7.3 Protocol extensions

The extension to the AlterEventEnrollment service argument shall be the CS AlterEventEnrollment Request.

```
CS AlterEventEnrollment Request ::= SEQUENCE {
  changeDisplay         CHOICE {
    enhancementString      [0] IMPLICIT VisibleString,
    enhancementIndex       [1] IMPLICIT INTEGER,
    noEnhancement          [2] NULL
    } OPTIONAL }
```

The choice of Display Enhancement for Display Option shall be indicated by the inclusion of the changeDisplay field in the PDU. If Unchanged Display is chosen, the changeDisplay field shall not occur.

7.2.5.8 EventNotification service

This subclause specifies the extensions to the EventNotification service.

7.2.5.8.1 Parameter extensions

The EventNotification service shall be extended to include specification of the Display Enhancement attribute of the Event Condition object. The structure of the EventNotification parameter extensions is specified in Table 7.

Table 7 — EventNotification parameter extension

Parameter name	Req	Ind	Rsp	Cnf	CBB
Display Enhancement	M	M(=)			
Display Enhancement string	S	S(=)			DES
Display Enhancement index	S	S(=)			DEI
No Enhancement	S	S(=)			

7.2.5.8.1.1 Display Enhancement

The value of this parameter indicates the type of Display Enhancement desired. Depending on its value, one of the following parameters shall be selected.

7.2.5.8.1.1.1 Display Enhancement string

This parameter, of type character string, is the string form of the Display Enhancement parameter. This selection may be made only if the DES CBB has been negotiated.

7.2.5.8.1.1.2 Display Enhancement index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter. This selection may be made only if the DEI CBB has been negotiated.

7.2.5.8.1.1.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present. This parameter shall be selected if neither DES nor DEI has been negotiated.

7.2.5.8.2 Service procedure extensions

The service procedure of 15.17.2 of ISO/IEC 9506 1 shall be performed, including the value of the Display Enhancement parameter in the response PDU. If the value of the Display Enhancement Class attribute of the Event Enrollment object referenced in the Event Condition request is not equal to UNDEFINED, the value of the Display Enhancement parameter shall be derived from the value of the Display Enhancement attribute of the Event Enrollment object. Otherwise, the value of the Display Enhancement parameter shall be derived from the value of the Display enhancement attribute of the Event Condition object referenced in the Event Notification request.

7.2.5.8.3 Protocol extensions

The extensions to the EventNotification service argument shall be the CS EventNotification.

```
CS EventNotification ::=  [0] Choice {
  enhancementString         [0] IMPLICIT VisibleString,
  enhancementIndex          [1] IMPLICIT INTEGER,
  noEnhancement             NULL
  }
```

> **NOTE —** As a result of the way in which Unconfirmed PDU is specified in ISO/IEC 9506 2, a NULL specified as a Companion Standard extension is not transmitted. The effect is as though the parameter were not included in the protocol. Specification of the NULL is required at the service interface, however.

7.2.5.9 Event Condition Additional Detail

The Additional Detail parameter that is used in the Get Alarm Summary service and in the Get Alarm Enrollment Summary service is specified to be the Display Enhancement parameter defined in 6.4.1.1. The value of this parameter shall be derived from the attributes of the Event Condition object being referenced in the respective service response.

7.2.5.9.1 Additional Detail protocol

This part of ISO/IEC 9506 defines the EN Additional Detail production as follows:

```
EN Additional Detail ::=  [0] Choice {
  enhancementString          [0] IMPLICIT VisibleString,
  enhancementIndex           [1] IMPLICIT INTEGER,
  noEnhancement              NULL
  }
```

7.2.6 Other productions

The following productions, required for the MMS General Module and representing Companion Standard Extensions to the MMS services, are not used in the Process Control Companion Standard and are set equal to NULL.

```
CS Status Request ::= NULL
CS Input Request ::= NULL
CS Output Request ::= NULL
CS InitiateDownloadSequence Request ::= NULL
CS DownloadSegment Request ::= NULL
CS TerminateDownloadSequence Request ::= NULL
CS InitiateUploadSequence Request ::= NULL
CS UploadSegment Request ::= NULL
CS TerminateUploadSequence Request ::= NULL
CS RequestDomainDownload Request ::= NULL
CS RequestDomainUpload Request ::= NULL
CS LoadDomainContent Request ::= NULL
CS StoreDomainContent Request ::= NULL
CS DeleteDomain Request ::= NULL
CS GetDomainAttributes Request ::= NULL
CS CreateProgramInvocation Request ::= NULL
CS DeleteProgramInvocation Request ::= NULL
CS Start Request ::= NULL
CS Stop Request ::= NULL
CS Reset Request ::= NULL
CS Resume Request ::= NULL
CS Kill Request ::= NULL
CS GetProgramInvocationAttributes Request ::= NULL
CS DeleteEventCondition Request ::= NULL
CS GetEventConditionAttributes Request ::= NULL
CS ReportEventConditionStatus Request ::= NULL
CS TriggerEvent Request ::= NULL
CS DefineEventAction Request ::= NULL
CS DeleteEventAction Request ::= NULL
CS GetEventActionAttributes Request ::= NULL
CS ReportEventActionStatus Request ::= NULL
CS DeleteEventEnrollment Request ::= NULL
CS ReportEventEnrollmentStatus Request ::= NULL
CS GetEventEnrollmentAttributes Request ::= NULL
CS AcknowledgeEventNotification Request ::= NULL
CS GetAlarmSummary Request ::= NULL
CS GetAlarmEnrollmentSummary Request ::= NULL
CS ReadJournal Request ::= NULL
CS WriteJournal Request ::= NULL
CS InitializeJournal Request ::= NULL
CS ReportJournalStatus Request ::= NULL
CS CreateJournal Request ::= NULL
CS DeleteJournal Request ::= NULL
CS GetCapabilityList Request ::= NULL
CS Status Response ::= NULL
CS Input Response ::= NULL
CS Output Response ::= NULL
CS InitiateDownloadSequence Response ::= NULL
CS DownloadSegment Response ::= NULL
CS TerminateDownloadSequence Response ::= NULL
CS InitiateUploadSequence Response ::= NULL
CS UploadSegment Response ::= NULL
CS TerminateUploadSequence Response ::= NULL
CS RequestDomainDownload Response ::= NULL
CS RequestDomainUpload Response ::= NULL
CS LoadDomainContent Response ::= NULL
CS StoreDomainContent Response ::= NULL
CS DeleteDomain Response ::= NULL
CS CreateProgramInvocation Response ::= NULL
CS DeleteProgramInvocation Response ::= NULL
CS Start Response ::= NULL

CS Stop Response ::= NULL

CS Resume Response ::= NULL

CS Reset Response ::= NULL

CS Kill Response ::= NULL

CS DefineEventCondition Response ::= NULL

CS DeleteEventCondition Response ::= NULL

CS GetEventConditionAttributes Response ::= NULL

CS ReportEventConditionStatus Response ::= NULL

CS AlterEventConditionMonitoring Response ::= NULL

CS TriggerEvent Response ::= NULL

CS DefineEventAction Response ::= NULL

CS DeleteEventAction Response ::= NULL

CS GetEventActionAttributes Response ::= NULL

CS ReportEventActionStatus Response ::= NULL

CS DefineEventEnrollment Response ::= NULL

CS DeleteEventEnrollment Response ::= NULL

CS AlterEventEnrollment Response ::= NULL

CS ReportEventEnrollmentStatus Response ::= NULL

CS AcknowledgeEventNotification Response ::= NULL

CS GetAlarmSummary Response ::= NULL

CS GetAlarmEnrollmentSummary Response ::= NULL

CS ReadJournal Response ::= NULL

CS WriteJournal Response ::= NULL

CS InitializeJournal Response ::= NULL

CS ReportJournalStatus Response ::= NULL

CS CreateJournal Response ::= NULL

CS DeleteJournal Response ::= NULL

CS GetCapabilityList Response ::= NULL

CS UnsolicitedStatus ::= NULL

AdditionalService Error ::= NULL

AdditionalUnconfirmedService ::= NULL

EE Additional Detail ::= NULL

JOU Additional Detail ::= NULL

CS GetNameList Request ::= NULL

CS GetNameList Response ::= NULL
```

7.3 Definition and use of process control specific services

7.3.1 Control Element

The Control Element is a complex parameter used in several services to describe a single element of a Unit Control object.

7.3.1.1 Structure

The structure of the Control Element parameter is shown in Table 8.

Table 8 — Control Element Parameter

Parameter name	Rsp	Cnf
Begin Domain Definition	S	S(=)
Domain Name	M	M(=)
List of Capabilities	M	M(=)
Sharable	M	M(=)
Load Data	M	M(=)
Continue Domain Definition	S	S(=)
Domain Name	M	M(=)
Load Data	M	M(=)
End Domain Definition	S	S(=)
Domain Name	M	M(=)
Program Invocation Definition	S	S(=)
Program Invocation Name	M	M(=)
List of Domains	M	M(=)
Reusable	M	M(=)
Monitor	U	U(=)
Monitor Type	C	C(=)
Program Invocation State	C	C(=)

7.3.1.1.1 Begin Domain Definition

Selection of this parameter shall indicate that a new Domain is about to be defined. If selected, the following parameters shall appear.

7.3.1.1.1.1 Domain Name

This parameter, of type Identifier, shall identify the Domain to be processed.

7.3.1.1.1.2 List of Capabilities

This parameter, of type list of character string, shall identify the capabilities associated with this Domain.

7.3.1.1.1.3 Shareable

This parameter, of type boolean, shall indicate if true that the Domain is shareable, as defined in 10.8.1.1.3 of ISO/IEC 9506 1.

7.3.1.1.1.4 Load Data

This parameter shall be the initial (or the total) content of the Domain.

7.3.1.1.2 Continue Domain Definition

Selection of this parameter shall indicate that a Domain identified in a previous Control Element is about to have more data associated with it. If selected, the following parameters shall appear.

7.3.1.1.2.1 Domain Name

This parameter, of type Identifier, shall identify the Domain to be processed.

7.3.1.1.2.2 Load Data

This parameter shall be the (partial) content of the Domain.

7.3.1.1.3 End Domain Definition

Selection of this parameter shall indicate that a Domain identified in a previous Control Element is now complete. If selected, the following parameter shall appear.

7.3.1.1.3.1 Domain Name

This parameter, of type Identifier, shall identify the Domain to be processed.

7.3.1.1.4 Program Invocation Definition

Selection of this parameter shall indicate that a Program Invocation definition follows. If selected, the following parameters shall appear.

7.3.1.1.4.1 Program Invocation Name

This parameter, of type Identifier, shall indicate the Program Invocation to be defined.

7.3.1.1.4.2 List of Domains

This parameter, of type list of Identifier, shall indicate the Domains that are bound to the Program Invocation.

7.3.1.1.4.3 Reusable

This parameter, of type boolean, shall be utilized as specified in 11.2.1.1.3 of ISO/IEC 9506 1.

7.3.1.1.4.4 Monitor

This parameter, of type boolean, shall be utilized as specified in 11.2.1.1.4 of ISO/IEC 9506 1.

7.3.1.1.4.4.1 Monitor Type

This parameter, of type boolean, shall be present if and only if the value of the Monitor parameter is true. The use of this parameter is defined in 11.2.1.1.4.1 of ISO/IEC 9506 1.

7.3.1.1.4.5 Program Invocation State

This parameter, of type integer, shall indicate, if present, the state of the Program Invocation. When used with UnitControlLoad service, it shall indicate the state in which the Program Invocation shall be placed. When used with the UnitControlUpload service, it shall indicate the actual state of the Program Invocation.

7.3.1.2 Protocol

```
ControlElement ::= CHOICE {
  beginDomainDef      [0] SEQUENCE {
    domainName              [1] IMPLICIT Identifier,
    capabilities            [2] IMPLICIT SEQUENCE OF VisibleString,
    sharable                [3] IMPLICIT BOOLEAN,
    loadData                [4] LoadData OPTIONAL
    },
    continueDomainDef [1] SEQUENCE {
    domainName              [1] IMPLICIT Identifier,
    loadData                [3] LoadData
    },
  endDomainDef        [2] IMPLICIT Identifier,
  piDefinition        [3] IMPLICIT SEQUENCE {
    piName                  [0] IMPLICIT Identifier,
    listOfDomains           [1] IMPLICIT SEQUENCE OF Identifier,
    reusable                [2] IMPLICIT BOOLEAN DEFAULT TRUE,
    monitorType             [3] IMPLICIT BOOLEAN OPTIONAL,
    pIState           [4] IMPLICIT ProgramInvocationState OPTIONAL
  }
}

LoadData ::= CHOICE {
  non coded       [0] IMPLICIT OCTET STRING,
  coded           EXTERNAL
  }
```

7.3.1.2.1 Monitor

The abstract syntax of the Monitor parameter of the control element parameter shall be inferred from the presence or absence of the monitorType field. If the monitorType field is present, it shall indicate that the Monitor parameter is true. The value of the monitorType field shall indicate the value of the Monitor Type parameter of the service request as specified in 11.2.1.1.4 of ISO/IEC 9506 1.

7.3.2 InitiateUnitControlLoad service

The InitiateUnitControlLoad service may be used by a client to request a VMD to create a Unit Control object and prepare it for loading. The loading process uses two confirmed MMS services such that the MMS server requests elements of the contents of the Unit Control object. Table 9 shows the sequence of the service primitives.

Table 9 — Interaction of Unit Control primitives

MMS Client	MMS Server
InitiateUnitControlLoad.req -->	
	<-- UnitControlLoadSegment.req
UnitControlLoadSegment.rsp -->	
...	...
	<-- UnitControlLoadSegment.req
UnitControlLoadSegment.rsp -->	
	<-- InitiateUnitControlLoad.rsp

7.3.2.1 Structure

The structure of the component service primitives of the InitiateUnitControlLoad service is shown in Table 10.

Table 10 — InitiateUnitControlLoad service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result (+)			S	S(=)	
Result (-)			S	S(=)	
Error Type			M	M(=)	
Initiate Unit Control Error			M	M(=)	
Domain Name			S	S(=)	
Program Invocation Name			S	S(=)	

7.3.2.1.1 Argument

This parameter shall convey the parameters of the InitiateUnitControlLoad service request.

7.3.2.1.1.1 Unit Control Name

This parameter, of type Identifier, shall specify the name of the Unit Control object.

7.3.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request has succeeded.

7.3.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.2.1.3.1 Domain Name

This parameter shall indicate the Domain that was being created when the error was detected. Either this parameter or the Program Invocation Name parameter shall be selected.

7.3.2.1.3.2 Program Invocation Name

This parameter shall indicate the Program Invocation that was being created when the error was detected. Either this parameter or the Domain Name parameter shall be selected.

7.3.2.2 Service procedure

If a Unit Control object of the specified name already exists, the MMS server shall return a Result(-). Otherwise, the MMS server shall then issue one or more UnitControlLoadSegment requests, as appropriate, until it receives a response in which the More Follows parameter is set false. It shall perform the service procedure prescribed for that service. If, during the processing of the information contained in the response to the UnitControlLoadSegment service request, it detects an error either in creating a Domain or a Program Invocation, it shall halt the loading process and return a Result(-) indicating the Domain or Program Invocation for which loading was in progress when the error occurred.

If no error occurs in the processing of the UnitControlLoadSegment services, it shall create a Unit Control object with the name specified in the request argument, and initialize its attributes to reference the Domains and Program Invocations created in the loading process. It shall then return a Result(+).

7.3.2.3 InitiateUnitControlLoad protocol

The abstract syntax of the initiateUCLoad choice of the AdditionalService Request and the AdditionalService Response is specified by the InitiateUnitControlLoad Request and the InitiateUnitControlLoad Response respectively. The abstract syntax of the initiateUCLoad choice of the AdditionalService Error is specified by the InitiateUnitControl Error. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
InitiateUnitControlLoad Request  ::=  Identifier Unit Control Name

InitiateUnitControlLoad Response ::= NULL

InitiateUnitControl Error ::= CHOICE {

    domain              [0] IMPLICIT Identifier,

    programInvocation   [1] IMPLICIT Identifier

    }
```

7.3.2.3.1 InitiateUnitControlLoad Request

The abstract syntax of the initiateUCLoad choice of the AdditionalService Request type shall be the InitiateUnitControlLoad Request.

7.3.2.3.2 InitiateUnitControlLoad Response

The abstract syntax of the initiateUCLoad choice of the AdditionalService Response type shall be the InitiateUnitControlLoad Response.

7.3.2.3.3 InitiateUnitControlLoad Error

The abstract syntax of the initiateUCLoad choice of the AdditionalService Error type shall be the InitiateUnitControlLoad Error.

7.3.3 UnitControlLoadSegment service

The UnitControlLoadSegment service is used by an MMS server to obtain load data elements from the MMS client.

7.3.3.1 Structure

The structure of the component service primitives is shown in Table 11.

Table 11 — UnitControlLoadSegment service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result (+)			S	S(=)	
List of Control Elements			M	M(=)	
More Follows			M	M(=)	
Result (–)			S	S(=)	
Error Type			M	M(=)	

7.3.3.1.1 Argument

This parameter shall convey the parameters of the UnitControlLoadSegment service request.

7.3.3.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object in the VMD that is to be loaded.

7.3.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service has succeeded. If success is indicated, the following parameters shall appear.

7.3.3.1.2.1 List of Control Elements

This parameter shall contain the information necessary to construct the constituent Domains and Program Invocations of the Unit Control object. The presence of the Program Invocation State parameter of each Control Element shall be a user option.

7.3.3.1.2.2 More Follows

This boolean parameter shall indicate whether (true) or not (false) more UnitControlSegment service requests are needed to complete the construction of the Unit Control object.

7.3.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.3.2 Service procedure

The MMS server shall issue a UnitControlLoadSegment request specifying the name of the Unit Control object to be loaded. Upon receipt of the response, the MMS server shall process each of the control elements in the response in sequence.

For each item in the List of Control Elements the MMS server shall:

a) If the Control Element specifies the beginning of a Domain Definition, the MMS server shall verify that no Domain of that name exists in the VMD. It shall then create the Domain, using the List of Capabilities parameter provided, and place it in the loading state. If the Load Data parameter is provided, it shall then begin the loading process, using the load data.

b) If the Control Element specifies the continuation of a Domain Definition, the MMS server shall verify that the Domain exists and is in the LOADING state. It shall then continue the loading process, using the Load Data parameter provided.

c) If the Control Element specifies the end of a Domain Definition, the MMS server shall verify that the Domain exists and that it is in the LOADING state. It shall then place the Domain in the READY state.

d) If the Control Element specifies a Program Invocation definition, the MMS server shall verify that all the Domains in the List of Domains parameter exist and that they are in the READY state or that they are in the IN USE state and their sharable attribute is TRUE. It shall then create the named Program Invocation, linking it to the indicated Domains. It shall place each of the Domains in the IN USE state. If the Program Invocation State parameter is present, the MMS server shall place the Program Invocation in the state indicated by this parameter; otherwise, it shall place the Program Invocation in IDLE state.

NOTE — If the association is lost during the course of a sequence of UnitControlLoadSegment services such that a Domain is in an intermediate state, the provisions of 10.1.4.1 of ISO/IEC 9506 1 apply. Also note the restrictions that 8.3.2 of ISO/IEC 9506 2 places on the use of the Conclude service.

7.3.3.3 UnitControlLoadSegment protocol

The abstract syntax of the uCLoad choice of the AdditionalService Request and the AdditionalService Response is specified by the UnitControlLoadSegment Request and the UnitControlLoadSegment Response respectively. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
UnitControlLoadSegment Request  ::=  Identifier -- Unit Control Name
UnitControlLoadSegment Response ::= SEQUENCE {
    controlElements   [0] IMPLICIT SEQUENCE OF ControlElement,
    moreFollows       [1] IMPLICIT BOOLEAN DEFAULT TRUE
}
```

7.3.3.3.1 UnitControlLoadSegment Request

The abstract syntax of the uCLoad choice of the AdditionalService Request type shall be the UnitControlLoadSegment Request.

7.3.3.3.2 UnitControlLoadSegment Response

The abstract syntax of the uCLoad choice of the AdditionalService Response type shall be the UnitControlLoadSegment Response.

7.3.4 UnitControlUpload service

The UnitControlUpload service is used by an MMS client to obtain load data elements from the MMS server. This service may have to be invoked several times to obtain a complete upload of the Unit Control object.

7.3.4.1 Structure

The structure of the component service primitives is shown in Table 12.

Table 12 — UnitControlUpload service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Continue After	U	U(=)			
Domain Name	S	S(=)			
Upload ID	S	S(=)			
Program Invocation	S	S(=)			
Result (+)			S	S(=)	
List of Control Elements			M	M(=)	
Next Element			C	C(=)	
Domain Name			S	S(=)	
Upload ID			S	S(=)	
Program Invocation			S	S(=)	
Result (–)			S	S(=)	
Error Type			M	M(=)	

7.3.4.1.1 Argument

This parameter shall convey the parameters of the UnitControlUpload service request.

7.3.4.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object in the VMD that is to be uploaded.

7.3.4.1.1.2 Continue After

This optional parameter shall indicate where in the list of constituents of the Unit Control object to begin the next Control Element. If this parameter is not present, the upload shall begin at the beginning of the Unit Control object. If this parameter is present, one of the following parameters shall be selected.

7.3.4.1.1.2.1 Domain Name

This parameter, of type Identifier, shall indicate the next Domain that is to be uploaded.

7.3.4.1.1.2.2 Upload ID

This parameter, of type integer, shall indicate the upload state machine currently open for some Domain upload that is to be continued.

7.3.4.1.1.2.3 Program Invocation

This parameter, of type Identifier, shall indicate the next Program Invocation whose definition is to be uploaded.

7.3.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service has succeeded. If success is indicated, the following parameters shall appear.

7.3.4.1.2.1 List of Control Elements

This parameter shall contain the information necessary to construct the constituent Domains and Program Invocations of the Unit Control object. The Program Invocation State parameter shall be included in each Program Invocation definition within a Control Element, and its value shall be set corresponding to the actual state of the Program Invocation.

7.3.4.1.2.2 Next Element

This optional parameter shall indicate, if present, the first element not transmitted in this list of Control Elements that should be the next element to be transmitted if another UnitControlUpload request is received. If this parameter is absent, this shall indicate that uploading of the Unit Control object is complete with this PDU. If this parameter is present, one of the following parameters shall be selected.

7.3.4.1.2.2.1 Domain Name

This parameter, of type Identifier, shall indicate the next Domain that is to be uploaded.

7.3.4.1.2.2.2 Upload ID

This parameter, of type integer, shall indicate the upload state machine currently open for some Domain upload that is to be continued.

7.3.4.1.2.2.3 Program Invocation

This parameter, of type Identifier, shall indicate the next Program Invocation whose definition is to be uploaded.

7.3.4.1.3 Result(–)

The Result(–) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.4.2 Service procedure

For the purposes of responding to the UnitControlUpload service request, the MMS server shall maintain the constituents of a Unit Control object in an ordered list. An ordering of Domain Names based on the collating sequence of ISO 646 followed by a similar ordering of Program Invocations is suggested but not required. This part of ISO/IEC 9506 requires only that the ordering algorithm used be unambiguous and be such that any Program Invocation appear later in the ordering than any Domains on which it depends.

The MMS client may issue a UnitControlUpload request indicating a position in this ordering by identifying the next Domain to be uploaded, the Domain whose upload is partially complete, or the next Program Invocation definition to be uploaded. If the MMS client does not specify any such element, the upload is to start from the beginning of the ordering.

The MMS server shall verify the consistency of the List of Domain references attribute and the List of Program Invocation references attribute of the Unit Control object. If any objects so referenced do not exist, the MMS server shall amend the respective list.

The MMS server shall provide definitions for each constituent element of the Unit Control object in order, determined by its ordering algorithm. For each Domain in the Unit Control object, the MMS server shall create an Upload State Machine (See 10.1.4.2 of ISO/IEC 9506 1) and transmit all or part of the Domain content. The determination of the necessity of segmentation and the size of the segments shall be a local matter. For each Program Invocation in the Unit Control object, the MMS server shall transmit a Program Invocation definition record.

If, for any reason, the entire content of the Unit Control object cannot be contained within a single PDU, the MMS server shall provide an indication of the next element in the order that has not yet been transmitted.

a) If the next element to be transmitted is a Domain, the MMS server shall indicate the name of this Domain.

b) If a Domain content has been partially transmitted and more content of that Domain remains to be transmitted, the MMS server shall indicate the identify of the Upload State Machine currently active.

c) If the next element to be transmitted is a Program Invocation, the MMS server shall indicate the name of this Program Invocation.

If the present transmission exhausts the Unit Control object, that is it transmits the last element on the list, the MMS server shall omit the Next Element parameter.

7.3.4.3 UnitControlUpload protocol

The abstract syntax of the uCUpload choice of the AdditionalService Request and the AdditionalService Response is specified by the UnitControlUpload Request and the UnitControlUpload Response respectively. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
UnitControlUpload Request ::= SEQUENCE {
  unitControlName    [0] IMPLICIT Identifier, - - Unit Control Name
  continueAfter       CHOICE {
    domain              [1] IMPLICIT Identifier,
    ulsmID              [2] IMPLICIT INTEGER,
    programInvocation   [3] IMPLICIT Identifier
    } OPTIONAL
  }

UnitControlUpload Response ::= SEQUENCE {
  controlElements   [0] IMPLICIT SEQUENCE OF ControlElement,
  nextElement        CHOICE {
    domain              [1] IMPLICIT Identifier,
    ulsmID              [2] IMPLICIT INTEGER,
    programInvocation   [3] IMPLICIT Identifier
    } OPTIONAL
  }
```

7.3.4.3.1 UnitControlUpload Request

The abstract syntax of the uCUpload choice of the AdditionalService Request type shall be the UnitControlUpload Request.

7.3.4.3.2 UnitControlUpload Response

The abstract syntax of the uCUpload choice of the AdditionalService Response type shall be the UnitControlUpload Response.

7.3.5 StartUnitControl service

This service allows the MMS client to place all the constituent Program Invocations of a Unit Control object into the RUNNING state.

7.3.5.1 Structure

The structure of the component service primitives is shown in Table 13.

Table 13 — StartUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Execution Argument	U	U(=)			
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	
Start Unit Control Error			C	C(=)	
Program Invocation Name			M	M(=)	
Program Invocation State			M	M(=)	

7.3.5.1.1 Argument

This parameter conveys the parameters of the StartUnitControl service.

7.3.5.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object whose constituent Program Invocations are to be started.

7.3.5.1.1.2 Execution Argument

This parameter may be used to pass an information to the Program Invocations which are to be started.

7.3.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service has succeeded.

7.3.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.5.1.3.1 Start Unit Control Error

This parameter shall be included in the Result(-) if the failure was due to the failure of the derived Start procedure on a specific Program Invocation. If this parameter is included, the following fields shall also appear.

7.3.5.1.3.1.1 Program Invocation Name

This parameter shall indicate the name of the Program Invocation whose Start service failed.

7.3.5.1.3.2 Program Invocation State

This parameter shall indicate the resulting state of the Program Invocation whose Start service has failed. Following an unsuccessful Start service, the Program Invocation shall be returned to its previous state, if possible, or it shall be placed in the UNRUNNABLE state.

7.3.5.2 Service procedure

The MMS server shall:

a) For each entry on the List of Program Invocation references attribute of the Unit Control object, verify that the Program Invocation exists. If a Program Invocation does not exist, remove its reference from the List of Program Invocation references attribute.

b) For each entry on the List of Program Invocation references attribute of the Unit Control object, place the Program Invocation in the RUNNING state. This shall be done as follows:

   1) If the Program Invocation is already in the RUNNING, STARTING, or RESUMING state, do nothing.

   2) If the Program Invocation is in the IDLE or RESETTING state, perform a Start service procedure (See 11.4.2 of ISO/IEC 9506 1).

   3) If the Program Invocation is in the STOPPED or STOPPING state, perform a Resume procedure (See 11.6.2 of ISO/IEC 9506 1).

   4) If the Program Invocation is in the UNRUNNABLE state, return a Result(-) with a Start Unit Control Error parameter indicating the failed Program Invocation and its state.

   5) If any Start procedure on a constituent Program Invocation fails, return a Result(-) with a Start Unit Control Error parameter indicating the failed Program Invocation and its state.

c) Return a Result(+).

7.3.5.3 StartUnitControl protocol

The abstract syntax of the startUC choice of the AdditionalService Request and the AdditionalService Response is specified by the StartUnitControl Request and the StartUnitControl Response respectively. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
StartUnitControl Request ::= SEQUENCE {
  unitControlName      [0] IMPLICIT Identifier,   -- Unit Control Name
  executionArgument      CHOICE {
    simpleString           [1] IMPLICIT VisibleString,
    encodedString          EXTERNAL
    } OPTIONAL
  }
StartUnitControl Response ::= NULL
StartUnitControl Error ::= SEQUENCE {
  programInvocationName  [0] IMPLICIT Identifier,
  programInvocationState [1] IMPLICIT ProgramInvocationState
  }
```

7.3.5.3.1 StartUnitControl Request

The abstract syntax of the startUC choice of the AdditionalService Request type shall be the StartUnitControl Request.

7.3.5.3.2 StartUnitControl Response

The abstract syntax of the startUC choice of the AdditionalService Response type shall be the StartUnitControl Response.

7.3.5.3.3 StartUnitControl Error

The abstract syntax of the startUC choice of the AdditionalService Error type shall be the StartUnitControl Error, which shall be Program Invocation Name sub-parameter and the Program Invocation State sub-parameter, respectively, of the Result(-) parameter of the StartUnitControl.response primitive and shall appear as the Program Invocation Name sub-parameter and the Program Invocation State sub-parameter, respectively, of the StartUnitControl.confirm primitive, if issued.

7.3.6 StopUnitControl service

This service allows the MMS client to place all the constituent Program Invocations of a Unit Control object into the STOPPED state.

7.3.6.1 Structure

The structure of the component service primitives is shown in Table 14.

Table 14 — StopUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	
Stop Unit Control Error			C	C(=)	
Program Invocation Name			M	M(=)	
Program Invocation State			M	M(=)	

7.3.6.1.1 Argument

This parameter shall convey the parameter of the StopUnitControl service.

7.3.6.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object whose constituent Program Invocations are to be stopped.

7.3.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service has succeeded.

7.3.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.6.1.3.1 Stop Unit Control Error

This parameter shall be included in the Result(-) if the failure was due to the failure of the derived Stop procedure on a specific Program Invocation. If this parameter is included, the following fields shall also appear.

7.3.6.1.3.1.1 Program Invocation Name

This parameter shall indicate the name of the Program Invocation whose Stop service failed.

7.3.6.1.3.1.2 Program Invocation State

This parameter shall indicate the resulting state of the Program invocation whose Stop service has failed. Following an unsuccessful Stop service, the Program Invocation shall be returned to its previous state if possible, or it shall be placed in the UNRUNNABLE state.

7.3.6.2 Service procedure

The MMS server shall:

a) For each entry on the List of Program Invocation references attribute of the Unit Control object, verify that the Program Invocation exists. If this condition is not satisfied, remove the reference to the Program Invocation from the List of Program Invocation references attribute.

b) For each entry on the List of Program Invocation references attribute of the Unit Control object, place each Program Invocation that is in the RUNNING state into the STOPPED state. This shall be done as follows:

   1) If the Program Invocation is already in the STOPPED, STOPPING, IDLE, RESETTING, or UNRUNNABLE state, do nothing.

   2) If the Program Invocation is in the RUNNING or STARTING state, perform a Stop procedure (See 11.6.3 of ISO/IEC 9506 1).

   3) If any Stop procedure on a constituent Program Invocation fails, return a Result(-) with a Stop Unit Control Error parameter indicating the failed Program Invocation and its state.

c) Return a Result(+).

7.3.6.3 StopUnitControl protocol

The abstract syntax of the stopUC choice of the AdditionalService Request and the AdditionalService Response is specified by the StopUnitControl Request and the StopUnitControl Response respectively. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
StopUnitControl Request ::= Identifier -- Unit Control Name
StopUnitControl Response ::= NULL
StopUnitControl Error ::= SEQUENCE {
    programInvocationName      [0] IMPLICIT Identifier,
    programInvocationState     [1] IMPLICIT ProgramInvocationState
    }
```

7.3.6.3.1 StopUnitControl Request

The abstract syntax of the stopUC choice of the AdditionalService Request type shall be the StopUnitControl Request.

7.3.6.3.2 StopUnitControl Response

The abstract syntax of the stopUC choice of the AdditionalService Response type shall be the StopUnitControl Response.

7.3.6.3.3 StopUnitControl Error

The abstract syntax of the stopUC choice of the AdditionalService Error shall be the StopUnitControl Error, which shall be Program Invocation Name sub parameter and the Program Invocation State sub parameter, respectively, of the Result(-) parameter of the StopControlUnit.response primitive and shall appear as the Program Invocation Name sub parameter and the Program Invocation State sub parameter, respectively, of the StopUnitControl.confirm primitive, if issued.

7.3.7 CreateUnitControl service

The CreateUnitControl service is used by an MMS client to create a new Unit Control object with a specified set of Domains and/or Program Invocations.

7.3.7.1 Structure

The structure of the component service primitives is shown in Table 15.

Table 15 — CreateUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
List of Domains	M	M(=)			
List of Program Invocations	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

7.3.7.1.1 Argument

This parameter shall convey the parameters of the CreateUnitControl service request.

7.3.7.1.1.1 Unit Control Name

This parameter, of type Identifier, is the name that shall be assigned to the newly created Unit Control object.

7.3.7.1.1.2 List of Domains

This parameter, of type list of Identifier, shall specify zero or more Domains that are to be referenced by the List of Domains attribute of the Unit Control object.

7.3.7.1.1.3 List of Program Invocations

This parameter, of type list of Identifier, shall specify zero or more Program Invocations that are to be referenced by the List of Program Invocations attribute of the Unit Control object.

7.3.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service has succeeded.

7.3.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.7.2 Service procedure

The MMS server shall:

a) Create a Unit Control object and assign it the specified name.

b) For each element of the List of Domains parameter (if any), add a reference to that Domain to the List of Domains attribute of the Unit Control object.

c) For each element of the List of Program Invocations parameter (if any), add a reference to that Program Invocation to the List of Program Invocations attribute of the Unit Control object.

d) Return a Result(+)

7.3.7.3 CreateUnitControl protocol

The abstract syntax of the createUC choice of the AdditionalService Request and the AdditionalService Response is specified by the CreateUnitControl Request and the CreateUnitControl Response respectively. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
CreateUnitControl Request ::= SEQUENCE {
    unitControl        [0] IMPLICIT Identifier, - - Unit Control Name
    domains            [1] IMPLICIT SEQUENCE OF Identifier,
    programInvocations [2] IMPLICIT SEQUENCE OF Identifier
    }
CreateUnitControl Response ::= NULL
```

7.3.7.3.1 CreateUnitControl Request

The abstract syntax of the createUC choice of the AdditionalService Request type shall be the CreateUnitControl Request.

7.3.7.3.2 CreateUnitControl Response

The abstract syntax of the createUC choice of the AdditionalService Response type shall be the CreateUnitControl Response.

7.3.8 AddToUnitControl service

The AddToUnitControl service is used by an MMS client to add Domains and/or Program Invocations to the Unit Control object.

7.3.8.1 Structure

The structure of the component service primitives is shown in Table 16.

Table 16 — AddToUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
List of Domains	M	M(=)			
List of Program Invocations	M	M(=)			
Result(+)			S	S(=)	
Result(–)			S	S(=)	
Error Type			M	M(=)	

7.3.8.1.1 Argument

This parameter shall convey the parameters of the AddToUnitControl service request.

7.3.8.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object in the VMD whose list of constituents is to be altered.

7.3.8.1.1.2 List of Domains

This parameter, of type list of Identifier, shall specify zero or more Domains that are to be added to the List of Domains attribute of the Unit Control object.

7.3.8.1.1.3 List of Program Invocations

This parameter, of type list of Identifier, shall specify zero or more Program Invocations that are to be added to List of Program Invocations attribute of the Unit Control object.

7.3.8.1.2 Result(+)

The Result(+) parameter shall indicate that the service has succeeded.

7.3.8.1.3 Result(–)

The Result(–) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.8.2 Service procedure

The MMS server shall:

a) For each element of the List of Domains parameter (if any), add a reference to that Domain to the List of Domains attribute of the Unit Control object.

b) For each element of the List of Program Invocations parameter (if any), add a reference to that Program Invocation to the List of Program Invocations attribute of the Unit Control object.

c) Return a Result(+)

7.3.8.3 AddToUnitControl protocol

The abstract syntax of the addToUC choice of the AdditionalService Request and the AdditionalService Response is specified by the AddToUnitControl Request and the AddToUnitControl Response respectively. These types are specified below and described in the paragraphs that follow. Clause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
AddToUnitControl Request ::= SEQUENCE {
  unitControl          [0] IMPLICIT Identifier,  - - Unit Control Name
  domains              [1] IMPLICIT SEQUENCE OF Identifier,
  programInvocations   [2] IMPLICIT SEQUENCE OF Identifier
}
AddToUnitControl Response ::= NULL
```

7.3.8.3.1 AddToUnitControl Request

The abstract syntax of the addToUC choice of the AdditionalService Request type shall be the AddToUnitControl Request.

7.3.8.3.2 AddToUnitControl Response

The abstract syntax of the addToUC choice of the AdditionalService Response type shall be the AddToUnitControl Response.

7.3.9 RemoveFromUnitControl service

The RemoveFromUnitControl service is used by an MMS client to remove Domains, or Program Invocations, or both from the Unit Control object.

7.3.9.1 Structure

The structure of the component service primitives is shown in Table 17.

Table 17 — RemoveFromUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
List of Domains	M	M(=)			
List of Program Invocations	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

7.3.9.1.1 Argument

This parameter shall convey the parameters of the RemoveFromUnitControl service request.

7.3.9.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object in the VMD whose list of constituents is to be altered.

7.3.9.1.1.2 List of Domains

This parameter, of type list of Identifier, shall specify zero or more Domains that are to be removed from the List of Domains reference attribute of the Unit Control object.

7.3.9.1.1.3 List of Program Invocations

This parameter, of type list of Identifier, shall specify zero or more Program Invocations that are to be removed from the List of Program Invocations reference attribute of the Unit Control object.

7.3.9.1.2 Result(+)

The Result(+) parameter shall indicate that the service has succeeded.

7.3.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.9.2 Service procedure

The MMS server shall:

a) For each element of the List of Domains parameter (if any), remove the reference to that Domain from the List of Domains reference attribute of the Unit Control object.

b) For each element of the List of Program Invocations parameter (if any), remove the reference to that Program Invocation from the List of Program Invocation references attribute of the Unit Control object.

c) Return a Result(+)

7.3.9.3 RemoveFromUnitControl protocol

The abstract syntax of the removeFromUC choice of the AdditionalService Request and the AdditionalService Response is specified by the RemoveFromUnitControl Request and the RemoveFromUnitControl Response respectively. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
RemoveFromUnitControl Request ::= SEQUENCE {
  unitControl          [0] IMPLICIT Identifier,  -- Unit Control Name
  domains              [1] IMPLICIT SEQUENCE OF Identifier,
  programInvocations   [2] IMPLICIT SEQUENCE OF Identifier
  }
RemoveFromUnitControl Response ::= NULL
```

7.3.9.3.1 RemoveFromUnitControl Request

The abstract syntax of the removeFromUC choice of the AdditionalService Request type shall be the RemoveFromUnitControl Request.

7.3.9.3.2 RemoveFromUnitControl Response

The abstract syntax of the removeFromUC choice of the AdditionalService Response type shall be the RemoveFromUnitControl Response.

7.3.10 GetUnitControlAttributes service

This service allows the MMS client to get the list of constituent Domains and Program Invocations of a Unit Control object.

7.3.10.1 Structure

The structure of the component service primitives is shown in Table 18.

Table 18 — GetUnitControlAttributes service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result(+)			S	S(=)	
List of Domains			M	M(=)	
List of Program Invocations			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

7.3.10.1.1 Argument

This parameter shall convey the parameter of the GetUnitControlAttributes service request.

7.3.10.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object for which the attributes are to be obtained.

7.3.10.1.2 Result(+)

The Result(+) parameter shall indicate that the service has succeeded. If success is indicated, the following parameters shall appear.

7.3.10.1.2.1 List of Domains

This parameter, of type list of Identifier, shall specify the names of the Domains that are constituents of the Unit Control object.

7.3.10.1.2.2 List of Program Invocations

This parameter, of type list of Identifier, shall specify the names of the Program Invocations that are constituents of the Unit Control object.

7.3.10.1.3 Result(-)

The Result(-) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.10.2 Service procedure

The MMS server shall:

a) For each entry in the List of Program Invocation references attribute of the Unit Control object, verify that the Program Invocation exists. If the Program Invocation does not exist, remove its reference from the List of Program Invocation references attribute of the Unit Control object.

b) For each entry in the List of Domain references attribute of the Unit Control object, verify that the Domain exists. If the Domain does not exist, remove its reference from the List of Domain references attribute of the Unit Control object.

c) Return a Result(+) with the list of names of the Domains and Program Invocations as specified in the List of Domains attribute and List of Program Invocations attribute of the Unit Control object.

NOTE — Following the model of Unit Control object given in this part of ISO/IEC 9506, it is possible that the list of constituents of a Unit Control object may become inconsistent with the actual set of Domains and Program Invocations, e.g., following the explicit deletion of a Domain. The service procedure of this section is intended to re-establish consistency for this Unit Control object prior to completion of the service. A real implementation may choose to maintain consistency at all times by employing a set of inverse references in each Domain and Program Invocation. However, this is not required. An alternate implementation technique could be to implement the references within the Unit Control object by name, reestablishing consistency only when required.

7.3.10.3 GetUnitControlAttributes protocol

The abstract syntax of the getUCAttributes choice of the AdditionalService Request and the AdditionalService Response is specified by the GetUnitControlAttributes Request and the GetUnitControlAttributes Response respectively. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetUnitControlAttributes Request ::= Identifier  -- Unit Control Name
GetUnitControlAttributes Response ::= SEQUENCE {
  domains              [0] IMPLICIT SEQUENCE OF Identifier,
  programInvocations   [1] IMPLICIT SEQUENCE OF Identifier
  }
```

7.3.10.3.1 GetUnitControlAttributes Request

The abstract syntax of the getUCAttributes choice of the AdditionalService Request type shall be the GetUnitControlAttributes Request.

7.3.10.3.2 GetUnitControlAttributes Response

The abstract syntax of the getUCAttributes choice of the AdditionalService Response type shall be the GetUnitControlAttributes Response.

7.3.11 LoadUnitControlFromFile service

The LoadUnitControlFromFile service may be used by a MMS client to request a MMS server to create a Unit Control object and load the Unit Control object using information available locally or from a third party.

7.3.11.1 Structure

The Structure of the component service primitives is shown in Table 19.

Table 19 — LoadUnitControlFromFile service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
File Name	M	M(=)			
Third Party	U	U(=)			TPY
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Initiate Unit Control Error			M	M(=)	
Domain Name			S	S(=)	
Program Invocation Name			S	S(=)	

7.3.11.1.1 Argument

This parameter shall convey the parameters of the LoadUnitControlFromFile service request.

7.3.11.1.1.1 Unit Control Name

This parameter shall specify the name of the Unit Control object at the VMD to be loaded.

7.3.11.1.1.2 File Name

This parameter, of type FileName, shall specify the name of the file containing the information to be loaded.

7.3.11.1.1.3 Third Party

This parameter, of type ApplicationReference, shall specify the application reference of the Application Process through which the named file may be accessed. Support of processing for this parameter is an implementation option that shall be implemented if support for the TPY parameter conformance building block is claimed. If it is implemented, its use is a user option. If this parameter is absent, the MMS server shall attempt to access the requested file directly.

7.3.11.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

7.3.11.1.3 Result (-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.11.1.3.1 Domain Name

This parameter shall indicate the Domain that was being created when the error was detected. Either this parameter or the Program Invocation Name parameter may be present, but not both.

7.3.11.1.3.2 Program Invocation Name

This parameter shall indicate the Program Invocation that was being created when the error was detected. Either this parameter or the Domain Name parameter may be present, but not both.

7.3.11.2 Service procedure

The VMD shall verify that the Unit Control object of the specified name does not exist. If a third party is specified, establish an association with that application if none exists; thereafter take appropriate action to cause the named Unit Control object to be loaded. If no third party is specified, perform the necessary steps to obtain the file through local means and load it into the specified Unit Control object. If the loading is successful, return a Result(+); otherwise return a Result(-) indicating in the Initiate Unit Control Error parameter the reason for failure.

7.3.11.3 LoadUnitControlFromFile protocol

The abstract syntax of the loadUCFromFile choice of the ConfirmedService Request and ConfirmedService Response is specified by the LoadUnitControlFromFile Request and LoadUnitControlFromFile Response specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
LoadUnitControlFromFile Request ::= SEQUENCE  {
  unitControlName            [0] IMPLICIT Identifier,
  fileName                   [1] IMPLICIT FileName,
  thirdParty                 [2] IMPLICIT ApplicationReference
                             OPTIONAL
  }
LoadUnitControlFromFile Response ::= NULL
LoadUnitControlFromFile Error ::= CHOICE {
  none                      [0] IMPLICIT NULL,
  domain                    [1] IMPLICIT Identifier,
  programInvocation         [2] IMPLICIT Identifier
  }
```

7.3.11.3.1 LoadUnitControlFromFile Request

The abstract syntax of the loadUCFromFile choice of the AdditionalService Request type shall be the LoadUnitControlFromFile Request.

7.3.11.3.2 LoadUnitControlFromFile Response

The abstract syntax of the loadUCFromFile choice of the AdditionalService Response type shall be the LoadUnitControlFromFile Response.

7.3.11.3.3 LoadUnitControlFromFile Error

The abstract syntax of the loadUCFromFile choice of the AdditionalService Error type shall be the LoadUnitControlFromFile Error.

7.3.12 StoreUnitControlToFile service

The StoreUnitControlToFile service may be used by a MMS client to request a MMS server to store the Domains and Program Invocations of a Unit Control object either at a third party site or locally.

7.3.12.1 Structure

The structure of the component service primitives is shown in Table 20.

Table 20 — StoreUnitControlToFile service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
File Name	M	M(=)			
Third Party	U	U(=)			TPY
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

7.3.12.1.1 Argument

This parameter shall convey the parameters of the StoreUnitControlToFile service request.

7.3.12.1.1.1 Unit Control Name

This parameter shall specify the name of the Unit Control object at the VMD for which the content is to be stored.

7.3.12.1.1.2 File Name

This parameter, of type FileName, shall specify the name of the file to which the information is to be stored.

7.3.12.1.1.3 Third Party

This optional parameter, of type ApplicationReference, shall specify the application reference of the Application Process on which the file store resides that is to receive the contents of the specified Unit Control object. Support of processing for this parameter is an implementation option that shall be implemented if support for the TPY parameter conformance building block is claimed. If it is implemented, its use is a user option. If this parameter is absent, the MMS server shall attempt to store the content of the Unit Control object directly in the specified file.

7.3.12.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not supply service specific parameters.

7.3.12.1.3 Result (-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.12.2 Service procedure

The MMS server shall verify that the Unit Control object of the specified name exists. If the third party parameter has been provided, the MMS server shall establish an association with that application if none exists; thereafter it shall take appropriate action to cause the named Unit Control object to be stored at the third party site. If no third party is specified, the MMS server shall perform the necessary steps to store the Unit Control object in the file specified through local means. If the process is successful, return a Result(+); otherwise return a Result(-).

7.3.12.3 StoreUnitControlToFile protocol

The abstract syntax of the storeUCToFile choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified by the StoreUnitControlToFile Request and StoreUnitControlToFile Response specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
StoreUnitControlToFile Request ::= SEQUENCE  {
  unitControlName           [0] IMPLICIT Identifier,
  fileName                  [1] IMPLICIT FileName,
  thirdParty                [2] IMPLICIT ApplicationReference
                             OPTIONAL
  }
StoreUnitControlToFile Response ::= NULL
```

7.3.12.3.1 StoreUnitControlToFile Request

The abstract syntax of the storeUCToFile choice of the AdditionalService Request type shall be the StoreUnitControlToFile Request.

7.3.12.3.2 StoreUnitControlToFile Response

The abstract syntax of the storeUCToFile choice of the AdditionalService Response type shall be the StoreUnitControlToFile Response.

7.3.13 DeleteUnitControl service

This service allows the MMS client to delete the Unit Control object and all its constituent elements.

7.3.13.1 Structure

The structure of the component service primitives is shown in Table 21.

Table 21 — DeleteUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result (+)			S	S(=)	
Result (-)			S	S(=)	
Error Type			M	M(=)	
Delete Unit Control Error			M	M(=)	
Domain Name			S	S(=)	
Program Invocation Name			S	S(=)	

7.3.13.1.1 Argument

This parameter conveys the parameter of DeleteUnitControl service.

7.3.13.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object that is to be deleted with its constituent elements.

7.3.13.1.2 Result(+)

The Result(+) parameter shall indicate that the service has succeeded.

7.3.13.1.3 Result(-)

The Result(-) parameter shall indicate that the service request has failed. The Error Type parameter, which is defined in detail in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.13.1.3.1 Domain Name

This parameter shall indicate the Domain whose deletion was being attempted when the error was detected. Either this parameter or the Program Invocation Name parameter shall be selected.

7.3.13.1.3.2 Program Invocation Name

This parameter shall indicate the Program Invocation whose deletion was being attempted when the error was detected. Either this parameter or the Domain Name parameter shall be selected.

7.3.13.2 Service procedure

The MMS server shall:

a) For each entry in the List of Program Invocation references attribute of the Unit Control object:

   1) Verify that the Program Invocation exists. If the Program Invocation does not exist, remove its reference from the List of Program Invocation references attribute of the Unit Control object and skip the remainder of this step for this entry.

   2) Verify that the Program Invocation is not in the RUNNING state. If this condition is not satisfied, return a Result(-) and skip the remainder of the procedure.

   3) Perform a DeleteProgramInvocation service procedure as specified in 11.3.2 of ISO/IEC 9506 1 and remove the reference to this Program Invocation from the List of Program Invocation references attribute of the Unit Control object.

b) For each entry in the List of Domain references attribute of the Unit Control object:

   1) Verify that the Domain exists. If the Domain does not exist, remove its reference from the List of Domain references attribute of the Unit Control object and skip the remainder of this step for this entry.

   2) Verify that the Domain is not in the IN USE state. If this condition is not satisfied, return a Result(-) and skip the remainder of the procedure.

   3) Perform a DeleteDomain service procedure as specified in 10.12.2 of ISO/IEC 9506 1 and remove the reference to this Domain from the List of Domain references attribute of the Unit Control object.

c) Delete the Unit Control object from the VMD.

d) Return a Result(+).

If the procedure returns a Result(-), the Unit Control object may have had some of its Domains and Program Invocations deleted. In this case, the Delete Unit Control Error parameter shall indicate the Domain or Program Invocation on which the procedure stopped, and the current contents of the Unit Control object shall indicate the Domains and Program Invocations that still remain on the Unit Control object.

> **NOTE —** Following the model of Unit Control object given in this part of ISO/IEC 9506, it is possible that the list of constituents of a Unit Control object may become inconsistent with the actual set of Domains and Program Invocations, e.g., following the explicit deletion of a Domain. The service procedure of this section is intended to re-establish consistency for this Unit Control object prior to completion of the service. A real implementation may choose to maintain consistency at all times by employing a set of inverse references in each Domain and Program Invocation. However, this is not required. An alternate implementation technique could be to implement the references within the Unit Control object by name, re-establishing consistency only when required.

7.3.13.3 DeleteUnitControl protocol

The abstract syntax of the deleteUC choice of the AdditionalService Request and the AdditionalService Response is specified by the DeleteUnitControl Request and the DeleteUnitControl Response respectively. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteUnitControl Request ::= Identifier  -- Unit Control Name

DeleteUnitControl Response ::= NULL

DeleteUnitControl Error ::= CHOICE {

  domain                    [0] IMPLICIT Identifier,

  programInvocation         [1] IMPLICIT Identifier

  }
```

7.3.13.3.1 DeleteUnitControl Request

The abstract syntax of the deleteUC choice of the AdditionalService Request type shall be the DeleteUnitControl Request.

7.3.13.3.2 DeleteUnitControl Response

The abstract syntax of the deleteUC choice of the AdditionalService Response type shall be the DeleteUnitControl Response.

7.3.13.3.3 DeleteUnitControl Error

The abstract syntax of the deleteUC choice of the AdditionalService Error type shall be the DeleteUnitControl Error.

7.3.14 DefineEventConditionList service

The DefineEventConditionList service shall be used to cause the creation of an Event Condition List object at the VMD.

7.3.14.1 Structure

The structure of the component service primitives is shown in Table 22.

Table 22 — DefineEventConditionList service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List name	M	M(=)			
List of Event Condition names	M	M(=)			
List of Event Condition List names	C	C(=)			RECL
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error type			M	M(=)	
Object in error			C	C(=)	

7.3.14.1.1 Argument

This parameter shall convey the parameters of the DefineEventConditionList service request.

7.3.14.1.1.1 Event Condition List name

This parameter, of type Object Name, shall specify the name of the Event Condition List object that is to be created at the VMD.

7.3.14.1.1.2 List of Event Condition names

This parameter shall contain a list of name attributes of Event Condition objects that shall be included in the specified Event Condition List. This list shall not be empty if RECL has not been negotiated. If there are no Event Condition names in this service request, this parameter shall be an empty list. If the scope of the Event Condition List name parameter is VMD specific or Domain specific, this parameter shall not contain an Event Condition name whose scope is AA specific.

7.3.14.1.1.3 List of Event Condition List names

This parameter shall contain a list of name attributes of Event Condition List objects that shall be included, by reference to each included object, in the Event Condition List. This parameter shall not be present if the RECL CBB has not been negotiated. If RECL has been negotiated but there are no Event Condition List names in this service request, this parameter shall be an empty list. If the scope of the Event Condition List name parameter is VMD specific or Domain specific, this parameter shall not contain an Event Condition List name whose scope is AA specific.

7.3.14.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return no service specific parameters.

7.3.14.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure. In addition, the following parameter may appear.

7.3.14.1.3.1 Object in error

This parameter, of type Object Name, shall be present when the error concerns the nonexistence or inconsistency of an Event Condition object specified in the List of Event Condition names parameter, or an Event Condition List object specified in the List of Event Condition List names parameter. It shall provide the name of the object determined in error at the VMD. This parameter shall not be present when the failure of this service is not due to the nonexistence or inconsistency of an Event Condition object or a Event Condition List object.

7.3.14.2 Service procedure

The VMD shall verify that no other Event Condition List objects exist at the VMD with a name equal to the value of the Event Condition List name parameter.

If one of the Event Condition objects specified in the List of Event Condition names parameter is determined to not exist at the VMD, or if one of the Event Condition List objects specified in the List of Event Condition List names parameter is determined not to exist at the VMD, the VMD shall issue the Result(-) service primitive with an Error Class of ACCESS, Error Code of OBJECT NON EXISTENT, and the Object in error parameter

If the scope of the Event Condition List name parameter is VMD specific or Domain specific, and the scope of the name of any of the Event Condition in the List of Event Condition names parameter or of any of the List of Event Condition List names parameter is AA specific, the VMD shall issue a Result(-) service primitive with an Error Class of DEFINITION, Error Code of OBJECT ATTRIBUTE INCONSISTENT, and the Object in error parameter.

If the requested definition is acceptable, a new Event Condition List object shall be created and initialized as below. This object shall then become part of the state of the association over which the requested definition was received, of the VMD or of a Domain of the VMD, depending on the scope of the Event Condition List name parameter.

If the List of Event Condition names parameter has been provided, for every Event Condition Object specified in the List of Event Condition names parameter, the VMD shall place a reference to the newly created Event Condition List object in the Event Condition object's List of referencing Event Condition List references attribute.

If the List of Event Condition List names parameter has been provided, for every Event Condition List object specified in the List of Event Condition List names parameter, the VMD shall place a reference to the newly created Event Condition List object in the referenced Event Condition List object's List of referencing Event Condition List references attribute.

Finally, a Result(+) shall be issued, indicating that the Event Condition List object was created and references updated.

The initial value for the attributes of the Event Condition List object are specified below:

a) Event Condition List name - Initialized to the value of the Event Condition List name parameter.

b) MMS Deletable - Initialized to true.

c) List of Event Condition references - Initialized to refer to the Event Condition objects specified by the value of the List of Event Condition names parameter.

d) List of Event Condition List references - Initialized to refer to the Event Condition List objects specified by the value of the List of Event Condition List names parameter.

e) List of referencing Event Condition List references - Initialized to an empty list.

7.3.14.3 DefineEventConditionList protocol

The abstract syntax of the defineECL choice of the ConfirmedService Request and ConfirmedService Response is specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DefineEventConditionList Request ::= SEQUENCE {
  eventConditionListName        [0] ObjectName,
  listOfEventConditionName      [1] IMPLICIT SEQUENCE OF ObjectName,
  listOfEventConditionListName  [2] IMPLICIT SEQUENCE OF ObjectName
                                OPTIONAL
                    -- shall appear if an only if RECL has been negotiated.
  }
DefineEventConditionList Response ::= NULL
DefineEventConditionList Error ::= ObjectName
```

7.3.14.3.1 DefineEventConditionList Request

The abstract syntax of the defineECL choice of the ConfirmedService Request type shall be the DefineEventConditionList Request.

7.3.14.3.2 DefineEventConditionList Response

The abstract syntax of the defineECL choice of the ConfirmedService Response type shall be the DefineEventConditionList Response.

7.3.14.3.3 DefineEventConditionList Error

The abstract syntax of the defineECL choice of the AdditionalService Error shall be the DefineEventConditionList Error, which shall be the Object in error parameter of the Result( ) parameter of the DefineEventConditionList.response primitive, and shall appear as the Object in error parameter of the DefineEventConditionList.confirm primitive, if issued.

7.3.15 DeleteEventConditionList service

The DeleteEventConditionList service shall be used to cause the deletion of an Event Condition List object at the VMD.

7.3.15.1 Structure

The structure of the component service primitives is shown in Table 23.

Table 23 — DeleteEventConditionList service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error type			M	M(=)	

7.3.15.1.1 Argument

This parameter shall convey the parameters of the DeleteEventConditionList service request.

7.3.15.1.1.1 Event Condition List name

This parameter, of type Object Name, shall specify the name of the Event Condition List object that is to be deleted at the VMD.

7.3.15.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return no service specific parameters.

7.3.15.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.15.2 Service procedure

If the request is acceptable, the VMD shall ensure that the value of the List of referencing Event Condition List references attribute of the specified Event Condition List object is equal to an empty list. If the value of this attribute is not equal to an empty list, the service request shall fail and a Result(-) shall be returned.

The VMD shall delete the reference to the specified Event Condition List object, contained in the List of referencing Event Condition List references attribute, from all other Event Condition List objects identified by the value of the List of Event Condition List references attribute.

The VMD shall delete references to the specified Event Condition List object, contained in the List of referencing Event Condition List references attribute, from Event Condition objects specified in and referenced by the value of the specified Event Condition List's List of Event Condition references attribute.

The VMD shall delete the specified Event Condition List object, and a Result(+) shall be returned.

7.3.15.3 DeleteEventConditionList protocol

The abstract syntax of the deleteECL choice of the ConfirmedService Request and ConfirmedService Response is specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
DeleteEventConditionList Request ::= ObjectName  --
                                     EventConditionListName

DeleteEventConditionList Response ::= NULL
```

7.3.15.3.1 DeleteEventConditionList Request

The abstract syntax of the deleteECL choice of the ConfirmedService Request type shall be the DeleteEventConditionList Request.

7.3.15.3.2 DeleteEventConditionList Response

The abstract syntax of the deleteECL choice of the ConfirmedService Response type shall be the DeleteEventConditionList Response.

7.3.16 AddEventConditionListReference service

The AddEventConditionListReference service shall be used to cause the addition of Event Condition object references, or Event Condition List object references, or both to a Event Condition List object at the VMD.

7.3.16.1 Structure

The structure of the component service primitives is shown in Table 24.

Table 24 — AddEventConditionListReference service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List name	M	M(=)			
List of Event Condition names	M	M(=)			
List of Event Condition List names	C	C(=)			RECL
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error type			M	M(=)	
Object in error			C	C(=)	

7.3.16.1.1 Argument

This parameter shall convey the parameters of the AddEventConditionListReference service request.

7.3.16.1.1.1 Event Condition List name

This parameter, of type Object Name, shall specify the name of the Event Condition List object that is to be modified at the VMD.

7.3.16.1.1.2 List of Event Condition names

This parameter shall contain a list of name attributes of Event Condition objects that shall be added to the specified Event Condition List. This list shall not be empty if RECL has not been negotiated. If there are no Event Condition names in this service request, this parameter shall be an empty list. If the scope of the Event Condition List name parameter is VMD specific or Domain specific, this parameter shall not contain an Event Condition name whose scope is AA specific.

7.3.16.1.1.3 List of Event Condition List names

This parameter shall contain a list of name attributes of Event Condition List objects that shall be added to the Event Condition List. This parameter shall not be present if the RECL CBB has not been negotiated. If RECL has been negotiated but there are no Event Condition List names in this service request, this parameter shall be an empty list. If the scope of the Event Condition List name parameter is VMD specific or Domain specific, this parameter shall not contain an Event Condition List name whose scope is AA specific.

7.3.16.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return no service specific parameters.

7.3.16.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure. In addition, the following parameter may appear.

7.3.16.1.3.1 Object in error

This parameter, of type Object Name, shall be present when the error concerns the non-existence or attribute inconsistency of an Event Condition object specified in the List of Event Condition names parameter, or an Event Condition List object specified in the List of Event Condition List names parameter. It shall provide the name of the object determined that caused the error at the VMD. This parameter shall not be present when the failure of this service is not due to the nonexistence or inconsistency of an Event Condition object or an Event Condition List object.

7.3.16.2 Service procedure

The VMD shall determine that the Event Condition List object specified by the Event Condition List name parameter exists at the VMD. If the specified Event Condition List object is determined to not exist, the Result(-) service primitive shall be issued with Error Class ACCESS and Error Code OBJECT NON EXISTENT.

If it is determined that one of the Event Condition objects specified in the List of Event Condition names parameter does not exist, or if it is determined that one of the Event Condition List objects specified in the List of Event Condition List names parameter does not exist, the Result(-) service primitive shall be issued with Error Class ACCESS, Error Code OBJECT NON EXISTENT, and the Object in error parameter.

If the scope of the Event Condition List name parameter is VMD specific or Domain specific, and the scope of the name of any of the Event Condition in the List of Event Condition names parameter or of any of the List of Event Condition List names parameter is AA specific, the VMD shall issue a Result(-) service primitive with an Error Class of DEFINITION, Error Code of OBJECT ATTRIBUTE INCONSISTENT, and the Object in error parameter.

If the request is acceptable, and if the List of Event Condition names parameter has been provided, for every Event Condition Object specified in the List of Event Condition names parameter, the VMD shall:

a) Verify that the Event Condition object is not already referenced in this Event Condition List; if it is referenced, skip the remainder of this step for this Event Condition.

b) Place a reference to the specified Event Condition List object in the Event Condition object's List of referencing Event Condition List references attribute.

c) Add the identified Event Condition object reference to the specified Event Condition List object's List of Event Condition references attribute.

If the request is acceptable, and if the List of Event Condition List names parameter has been provided, for every Event Condition List object specified in the List of Event Condition List names parameter, the VMD shall:

a) Verify that the Event Condition List object is not already referenced in this Event Condition List; if it is referenced, skip the remainder of this step for this Event Condition List.

b) Place a reference to the specified Event Condition List object in the referenced Event Condition List object's List of referencing Event Condition List references attribute.

c) Add the identified Event Condition List object reference to the specified Event Condition List object's List of Event Condition List references attribute.

Finally, a Result(+) shall be issued, indicating that the Event Condition List object was modified and references updated.

7.3.16.3 AddEventConditionListReference protocol

The abstract syntax of the addECLReference choice of the ConfirmedService Request and ConfirmedService Response is specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
AddEventConditionListReference Request ::= SEQUENCE {
  eventConditionListName        [0] ObjectName,
  listOfEventConditionName      [1] IMPLICIT SEQUENCE OF ObjectName,
  listOfEventConditionListName  [2] IMPLICIT SEQUENCE OF ObjectName
                                 OPTIONAL
  -- shall appear if and only if RECL has been negotiated
  }
AddEventConditionListReference Response ::= NULL
AddEventConditionListReference Error ::= ObjectName
```

7.3.16.3.1 AddEventConditionListReference Request

The abstract syntax of the addECLReference choice of the ConfirmedService Request type shall be the AddEventConditionListReference Request.

7.3.16.3.2 AddEventConditionListReference Response

The abstract syntax of the addECLReference choice of the ConfirmedService Response type shall be the AddEventConditionListReference Response.

7.3.16.3.3 AddEventConditionListReference Error

The abstract syntax of the addECLReference choice of the AdditionalService Error shall be the AddEventConditionListReference Error, which shall be the Object in error parameter of the Result(-) parameter of the AddEventConditionListReference.response primitive, and shall appear as the Object in error parameter of the AddEventConditionListReference.confirm primitive, if issued.

7.3.17 RemoveEventConditionListReference service

The RemoveEventConditionListReference service shall be used to cause the removal of Event Condition object references, or removal of Event Condition List references, or both, from a specified Event Condition List object at the VMD.

7.3.17.1 Structure

The structure of the component service primitives is shown in Table 25.

Table 25 — RemoveEventConditionListReference service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List name	M	M(=)			
List of Event Condition names	M	M(=)			
List of Event Condition List names	C	C(=)			RECL
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error type			M	M(=)	
Object in error			C	C(=)	

7.3.17.1.1 Argument

This parameter shall convey the parameters of the RemoveEventConditionListReference service request.

7.3.17.1.1.1 Event Condition List name

This parameter, of type Object Name, shall specify the name of the Event Condition List object that is to be modified at the VMD.

7.3.17.1.1.2 List of Event Condition names

This parameter shall contain a list of name attributes of Event Condition objects that shall be removed from the specified Event Condition List. This list shall not be empty if RECL has not been negotiated. If there are no Event Condition names in this service request, this parameter shall be an empty list.

7.3.17.1.1.3 List of Event Condition List names

This parameter shall contain a list of name attributes of Event Condition List objects that shall be removed from the Event Condition List. This parameter shall not be present if the RECL CBB has not been negotiated. If RECL has been negotiated but there are no Event Condition List names in this service request, this parameter shall be an empty list.

7.3.17.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return no service specific parameters.

7.3.17.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure. In addition, the following parameter may appear.

7.3.17.1.3.1 Object in error

This parameter, of type Object Name, shall indicate the Event Condition object specified in the List of Event Condition names parameter, or the Event Condition List object specified in the List of Event Condition List names parameter, which caused the error. This error indicates that either (1) the specified object does not exist, or (2) that it is not referenced by the List of Event Condition reference or List of Event Condition List reference attribute of the Event Condition List object specified by the Event Condition List Name parameter.

7.3.17.2 Service procedure

The VMD shall determine that the Event Condition List object specified by the Event Condition List name parameter exists at the VMD. If the specified Event Condition List object is determined to not exist, the Result(-) service primitive shall be issued with Error Class ACCESS and Error Code OBJECT NON EXISTENT.

If it is determined that one of the Event Condition objects specified in the List of Event Condition names parameter does not exist, or if it is determined that one of the Event Condition List objects specified in the List of Event Condition List names parameter does not exist, the Result( ) service primitive shall be issued with Error Class ACCESS, Error Code OBJECT NON EXISTENT, and the Object in error parameter.

If it is determined that one of the Event Condition objects specified in the List of Event Condition names parameter is not referenced by the List of Event Condition reference attributeof the Event Condition List object, or if it is determined that one of the Event Condition List objects specified in the List of Event Condition List names parameter is not reference by the List of Event Condition List reference attribute of the Event Condition List object, the Result(-) service primitive shall be issued with Error Class ACCESS, Error Code OBJECT NON EXISTENT, and the Object in error parameter.

If the request is acceptable, and if the List of Event Condition names parameter has been provided, for every Event Condition Object specified in the List of Event Condition names parameter, the VMD shall remove the reference to the specified Event Condition List object in the Event Condition object's List of referencing Event Condition List references attribute. The VMD shall remove the reference to each identified Event Condition object from the specified Event Condition List object's List of Event Condition references attribute.

If the List of Event Condition List names parameter has been provided, for every Event Condition List object specified in the List of Event Condition List names parameter, the VMD shall remove the reference to the specified Event Condition List object in the Event Condition List object's List of referencing Event Condition List references attribute. The VMD shall remove the reference to each identified Event Condition List object from the specified Event Condition List object's List of Event Condition List references attribute.

7.3.17.3 RemoveEventConditionListReference protocol

The abstract syntax of the removeECLReference choice of the ConfirmedService Request and ConfirmedService Response is specified below and described in the paragraphs that follow. Subclause 5.5 of part 2 of ISO 9506 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
RemoveEventConditionListReference Request ::= SEQUENCE {
  eventConditionListName         [0] ObjectName,
  listOfEventConditionName       [1] IMPLICIT SEQUENCE OF ObjectName,
  listOfEventConditionListName   [2] IMPLICIT SEQUENCE OF ObjectName
                                 OPTIONAL
   -- shall appear if and only if RECL has been negotiated
}
RemoveEventConditionListReference Response ::= NULL
RemoveEventConditionListReference Error ::= CHOICE {
    eventCondition            [0] ObjectName,
    eventConditionList        [1] ObjectName
    }
```

7.3.17.3.1 RemoveEventConditionListReference Request

The abstract syntax of the removeECLReference choice of the ConfirmedService Request type shall be the RemoveEventConditionListReference Request.

7.3.17.3.2 RemoveEventConditionListReference Response

The abstract syntax of the removeECLReference choice of the ConfirmedService Response type shall be the RemoveEventConditionListReference Response.

7.3.17.3.3 RemoveEventConditionListReference Error

The abstract syntax of the removeECLReference choice of the AdditionalService Error shall be the RemoveEventConditionListReference Error, which shall be the Object in error parameter of the Result(-) parameter of the RemoveEventConditionListReference.response primitive, and shall appear as the Object in error parameter of the RemoveEventConditionListReference.confirm primitive, if issued.

7.3.18 GetEventConditionListAttributes service

The GetEventConditionListAttributes service shall be used to determine the attribute values of a specified Event Condition List object at the VMD.

7.3.18.1 Structure

The structure of the component service primitives is shown in Table 26.

Table 26 — GetEventConditionListAttributes service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List name	M	M(=)			
Result(+)			S	S(=)	
List of Event Condition names			M	M(=)	
List of Event Condition List names			C	C(=)	RECL
Result(-)			S	S(=)	
Error type			M	M(=)	

7.3.18.1.1 Argument

This parameter shall convey the parameters of the GetEventConditionListAttributes service request.

7.3.18.1.1.1 Event Condition List name

This parameter, of type Object Name, shall specify the name of the Event Condition List object at the VMD from which the attribute values are to be obtained.

7.3.18.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the List of Event Condition names and the List of Event Condition List names parameters.

7.3.18.1.2.1 List of Event Condition names

This parameter shall contain a list of name attributes of Event Condition objects that have been derived from the List of Event Condition references attribute of the specified Event Condition List object. This list shall not be empty if RECL has not been negotiated. If there are no Event Condition names in this service request, this parameter shall be an empty list.

7.3.18.1.2.2 List of Event Condition List names

This parameter shall contain a list of name attributes of Event Condition List objects that have been derived from the List of Event Condition List references attribute of the specified Event Condition List object. This parameter shall not be present if the RECL CBB has not been negotiated. If RECL has been negotiated, but there are no Event Condition List names in this service request, this parameter shall be an empty list.

7.3.18.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.18.2 Service procedure

The VMD shall determine that the Event Condition List object specified by the Event Condition List Name parameter exists at the VMD. If the specified Event Condition List object is determined to not exist, the Result(-) service primitive shall be issued with Error Class ACCESS and Error Code OBJECT NON EXISTENT.

If the request is acceptable, the VMD shall obtain the values of the name attributes of the referenced Event Condition objects and Event Condition List objects, place them in the values of the List of Event Condition names parameter and List of Event Condition List names parameter respectively, and return the Result(+).

7.3.18.3 GetEventConditionListAttributes protocol

The abstract syntax of the getECLAttributes choice of the ConfirmedService Request and ConfirmedService Response is specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
GetEventConditionListAttributes Request ::= ObjectName - -
eventConditionListName

GetEventConditionListAttributes Response ::= SEQUENCE {
  listOfEventConditionName      [1] IMPLICIT SEQUENCE OF ObjectName,
  listOfEventConditionListName  [2] IMPLICIT SEQUENCE OF ObjectName
                                OPTIONAL
  -- shall appear if and only if RECL has been negotiated
  }
```

7.3.18.3.1 GetEventConditionListAttributes Request

The abstract syntax of the getECLAttributes choice of the ConfirmedService Request type shall be the GetEventConditionListAttributes Request.

7.3.18.3.2 GetEventConditionListAttributes Response

The abstract syntax of the getECLAttributes choice of the ConfirmedService Response type shall be the GetEventConditionListAttributes Response.

7.3.19 ReportEventConditionListStatus service

The ReportEventConditionListStatus service is used to convey the status of a Event Condition List.

7.3.19.1 Structure:

The structure of the component service primitives is shown in Table 27.

Table 27 — ReportEventConditionListStatus service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List name	M	M(=)			
Continue After	U	U(=)			
Result(+)			S	S(=)	
List of Event Condition Status			M	M(=)	
Current State			M	M(=)	
Number of Event Enrollments			M	M(=)	
Enabled			C	C(=)	
Time of Last Transition To Active			C	C(=)	
Time of Last Transition To Idle			C	C(=)	
More Follows			C	C(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

7.3.19.1.1 Argument

This parameter shall convey the parameter of the ReportEventConditionListStatus service request.

7.3.19.1.1.1 Event Condition List name

This parameter, of type Object Name. shall contain the name of the Event Condition List from which the status report is requested.

7.3.19.1.1.2 Continue After

This parameter, of type Object Name, indicates that the requesting MMS user requests the list of Event Condition Status returned by the responding MMS user to begin with an Event Condition object other than the first object in the list. The collating sequence specified by ISO 646 shall be used by the responding MMS user to determine the Event Condition object in the specified list to start after.

7.3.19.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. When success is indicated the following parameters are returned in the response primitive.

7.3.19.1.2.1 List of Event Condition Status

This parameter, of type Object Name, shall contain zero of more entries describing the status of Event Condition objects hierarchically related to the Event Condition List object.

7.3.19.1.2.1.1 Current State

This parameter, of type EC State, shall contain the value of the Event Condition object's State attribute.

7.3.19.1.2.1.2 Number of Event Enrollments

This parameter, of type integer, shall contain the count of the number of entries in the Event Condition object's List of Event Enrollment references attribute.

7.3.19.1.2.1.3 Enabled

This parameter, of type boolean, shall convey the contents of the Enabled attribute of the Event Condition object, for a monitored Event Condition object. If the Event Condition Class attribute contains the value NETWORK TRIGGERED, this parameter shall be omitted.

7.3.19.1.2.1.4 Time Of Last Transition To Active

If the Event Condition object is monitored and has a Time Of Last Transition To Active attribute value not equal to UNDEFINED, this parameter shall contain the current value of the Time 0f Last Transition To Active attribute. Otherwise, this parameter shall be omitted.

7.3.19.1.2.1.5 Time Of Last Transition To Idle

If the Event Condition object is monitored and has a Time Of Last Transition To Idle attribute with value not equal to UNDEFINED, this parameter shall contain the current value of the Time of Last Transition To Idle attribute. Otherwise, this parameter shall be omitted.

7.3.19.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional ReportEventConditionListStatus requests are necessary to retrieve more of the requested information. If true, more requests are necessary (if the requesting MMS user wishes to retrieve more data). If false, then either the List of Event Condition Status contains the last status in the list, or the List of Event Condition Status is empty. The More Follows parameter shall be false when the List of Event Condition Status parameter is empty.

7.3.19.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in Clause 17 of part 1 of ISO/IEC 9506, shall provide the reason for failure.

7.3.19.1.4 Service Procedure

The MMS server shall construct a list of Event Condition objects, either directly referenced by the named Event Condition List object, or (if RECL has been negotiated) through indirect references of Event Condition List objects referenced by the named Event Condition List object. The MMS server shall order this list using the collating sequence of ISO 646 on the name attributes of the Event Condition objects. It shall return the status information for as many of these Event Condition objects as it can accommodate in a single response, beginning either at the beginning of the list if the Continue After parameter has not been specified, or beginning at or immediately following the Event Condition object specified by the Continue After parameter.

7.3.19.1.5 ReportEventConditionListStatus protocol

The abstract syntax of the reportECLStatus choice of the ConfirmedService Request and ConfirmedService Response is specified below and described in the paragraphs that follow. Subclause 5.5 of part 2 of ISO/IEC 9506 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
ReportEventConditionListStatus Request  ::=  SEQUENCE {
  eventConditionListName           [0]  ObjectName - - Event Condition
                              List Name,
  continueAfter                    [1]  IMPLICIT Identifier OPTIONAL
  }
ReportEventConditionListStatus Response  ::= SEQUENCE  {
  listOfEventConditionStatus       [1] IMPLICIT SEQUENCE OF
                              EventConditionStatus,
  moreFollows                      [2] IMPLICIT BOOLEAN DEFAULT TRUE
  }
EventConditionStatus  ::= SEQUENCE   {
  eventConditionName               [0] ObjectName,
  currentState                     [1] IMPLICIT EC State,
  numberOfEventEnrollments         [2] IMPLICIT Unsigned32,
‘ enabled                          [3] IMPLICIT BOOLEAN OPTIONAL,
‘ timeOfLastTransitionToActive     [4] EventTime OPTIONAL,
  timeOfLastTransitionToIdle       [5] EventTime OPTIONAL
  }
```

7.3.19.1.5.1 ReportEventConditionListStatus Request

The abstract syntax of the reportECLStatus choice of the ConfirmedService Request type shall be the ReportEventConditionListStatus Request.

7.3.19.1.5.2 ReportEventConditionListStatus Response

The abstract syntax of the reportECLStatus choice of the ConfirmedService Response type shall be the ReportEventConditionListStatus Response.

7.3.20 AlterEventConditionListMonitoring service

The AlterEventConditionListMonitoring service shall be used to alter the value of attributes of Event Condition objects referenced by a Event Condition List object at the VMD. Attributes subject to modification by this service include the Enabled attribute and the Group Priority Override attribute.

7.3.20.1 Structure

The structure of the component service primitives is shown in Table 28.

Table 28 — AlterEventConditionListMonitoring service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List name	M	M(=)			
Enabled	U	U(=)			
Priority change	U	U(=)			
Priority value	S	S(=)			
Priority reset	S	S(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error type			M	M(=)	

7.3.20.1.1 Argument

This parameter shall convey the parameters of the AlterEventConditionListMonitoring service request.

7.3.20.1.1.1 Event Condition List name

This parameter, of type Object Name, shall specify the name of the Event Condition List object at the VMD for which attribute values are to be altered.

7.3.20.1.1.2 Enabled

This optional parameter, of type boolean, shall be the replacement value for the contents of the Enabled attribute of all Event Condition objects directly or indirectly referenced by the Event Condition List object. Either this parameter, or the Priority change parameter, or both, shall be provided.

7.3.20.1.1.3 Priority change

This optional parameter shall alter the Group Priority Override attribute of the referenced Event Condition objects. Either this parameter, or the Enabled parameter, or both shall be provided. If this parameter is provided, either the Priority value parameter or the Priority restore parameter shall be provided.

7.3.20.1.1.3.1 Priority value

This parameter, of type integer, shall constitute a replacement value for the value of the Group Priority Override attribute of all referenced Event Condition objects.

7.3.20.1.1.3.2 Priority reset

This parameter, of type NULL, shall indicate that the value of the Group Priority Override attribute shall be reset to UNDEFINED.

7.3.20.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return no service specific parameters.

7.3.20.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in Clause 17 of ISO/IEC 9506 1, shall provide the reason for failure.

7.3.20.2 Service procedure

The VMD shall determine that the Event Condition List object specified by the Event Condition List name parameter exists at the VMD. If the specified Event Condition List object is determined to not exist, the Result(-) service primitive shall be issued with Error Class ACCESS and Error Code OBJECT NON EXISTENT.

If the request is acceptable, for every Event Condition object referenced by the value of the specified Event Condition List object's List of Event Condition references attribute, and for every Event Condition object indirectly referenced through the value of the List of Event Condition List references attribute, the VMD shall:

a) If the Enabled parameter has been provided, the value of the Enabled attribute shall be set equal to the value of the Enabled parameter.

b) If the Priority change parameter has been provided the VMD shall:

   1) If the Priority value has been provided, replace the value of the Group Priority Override attribute with the value provided in the Priority Value parameter.

   2) If the Priority reset parameter has been provided, change the value of the Group Priority Override attribute to UNDEFINED.

c) Finally, the Result(+) shall be returned.

7.3.20.3 AlterEventConditionListMonitoring protocol

The abstract syntax of the alterECLMonitoring choice of the ConfirmedServiceRequest and ConfirmedServiceResponse is specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this clause.

```
AlterEventConditionListMonitoring Request ::= SEQUENCE {
  eventConditionListName      [0] ObjectName,
  enabled                     [1] IMPLICIT BOOLEAN DEFAULT FALSE,
  priorityChange              [2] CHOICE {
    priorityValue                  [0] IMPLICIT INTEGER,
    priorityReset                  [1] IMPLICIT NULL
    } OPTIONAL
  }
AlterEventConditionListMonitoring Response ::= NULL
```

7.3.20.3.1 AlterEventConditionListMonitoring Request

The abstract syntax of the alterECLMonitoring choice of the ConfirmedService Request type shall be the AlterEventConditionListMonitoring Request.

7.3.20.3.2 AlterEventConditionListMonitoring Response

The abstract syntax of the alterECLMonitoring choice of the ConfirmedService Response type shall be the AlterEventConditionListMonitoring Response.

7.4 The Initiate Service and Protocol

This subclause specifies the process control specific use of the Initiate service and protocol.

7.4.1 Init Request Detail parameter

MMS provides for a Init Request Detail parameter to be defined by companion standards. The structure of the Init Request Detail parameter is shown in Table 29.

Table 29 — Init Request Detail parameter

Parameter name	Req	Ind
Proposed Version Number	M	M(=)
Proposed Parameter CBB	M	M(=)
Services Supported Calling	M	M(=)
Additional Services Supported Calling	M	M(=)
Additional CBB Supported Calling	M	M(=)

The Init Request Detail shall contain additional parameters relating to communication in the presentation context derived from the abstract syntax defined in this part of ISO/IEC 9506. The component parameters are specified as follows:

7.4.1.1 Proposed Version Number

This parameter, of type integer, shall contain a number that represents a minor version number of this part of ISO/IEC 9506. This number is the proposed minor version number which will be used in the presentation context derived from the abstract syntax defined in this part of ISO/IEC 9506 for this instance of communication. Proposal of a number greater than one indicates support for all minor versions between one and the number proposed.

> **NOTE —** Major revisions of this part of ISO/IEC 9506 are reflected through the definition and registration of distinct abstract syntaxes (see Clause 5 of ISO/IEC 9506 2). Minor revisions are reflected in the minor version number parameter. Minor versions of this part of ISO/IEC 9506 at the same major revision level are compatible with versions of this part of ISO/IEC 9506 with smaller minor version numbers.

The value of this parameter may be reduced by the MMS provider if it cannot support the requested value. The value in the indication primitive shall be less than or equal to the value in the request primitive, but not less than one.

7.4.1.2 Proposed Parameter CBB

This parameter is specified in 8.2.3.2 of ISO/IEC 9506 1.

7.4.1.3 Services Supported Calling

This parameter is specified in 8.2.3.3 of ISO/IEC 9506 1.

7.4.1.4 Additional Services Supported Calling

This parameter, of type bitstring, shall specify support by the Calling MMS user of a set of additional services that are defined by this part of ISO/IEC 9506 for use in the presentation context derived from the abstract syntax defined in this part of ISO/IEC 9506 on the application association.

The value of the parameter in the indication primitive shall specify the intersection of the set of additional services supported by the Calling MMS user and the set of services supported by the MMS provider.

The assignment of a service to an individual bit of the bitstring type is specified in 7.4.3. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate nonsupport. Any additional bits shall be ignored.

Support for confirmed services shall be defined as the ability to receive a request indication and properly execute the service procedure defined for the responder role.

If a confirmed service is supported, then a reject PDU shall not be issued on receipt of a request for that service, except for a protocol error. If a confirmed service is not supported, then a reject PDU shall be issued on receipt of a request for that service with a reject code of "UNRECOGNIZED SERVICE."

7.4.1.5 Additional CBB Supported Calling

This parameter, of type bitstring, shall specify support by the Calling MMS user of a set of additional parameter CBBs that are defined by this part of ISO/IEC 9506 for use in the presentation context derived from the abstract syntax defined in this part of ISO/IEC 9506 on the application association.

The value of the parameter in the indication primitive shall specify the intersection of the set of additional CBBs supported by the Calling MMS user and the set of CBBs supported by the MMS provider.

The definition of additional CBBs in this part of ISO/IEC 9506 are specified in 6.6. The assignment of a CBB to an individual bit of the bitstring type is specified in 7.4.3. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate nonsupport. Any additional bits shall be ignored.

7.4.2 Init Response Detail Parameter

MMS provides for a Init Response Detail parameter to be defined by companion standards. The structure of the Init Response Detail parameter is shown in Table 30.

Table 30 — Init Response Detail parameter

Parameter name	Req	Ind
Negotiated Version Number	M	M(=)
Negotiated Parameter CBB	M	M(=)
Services Supported Called	M	M(=)
Additional Services Supported Called	M	M(=)
Additional CBB Supported Called	M	M(=)

The Init Response Detail parameter shall contain parameters relating to communication in the presentation context derived from the abstract syntax defined in this part of ISO/IEC 9506. The component parameters are specified as follows:

7.4.2.1 Negotiated Version Number

This parameter, of type integer, shall contain a number that represents a minor version number of this part of ISO/IEC 9506. This number is the minor version number of this part of ISO/IEC 9506 that will be used in the presentation context derived from the abstract syntax defined in this part of ISO/IEC 9506 for this instance of communication. This number shall be less than or equal to the Proposed Version Number parameter in the request primitive. It shall not be reduced to less than one.

> **NOTE —** Major revisions of this part of ISO/IEC 9506 are reflected through the definition and registration of distinct abstract syntaxes (See clause 5 of ISO/IEC 9506 2). Minor revisions are reflected in the minor version number parameter. Minor versions of this part of ISO/IEC 9506 at the same major revision level are compatible with versions of this part of ISO/IEC 9506 with smaller minor version numbers.

7.4.2.2 Negotiated Parameter CBB

This parameter is specified in 8.2.4.2 of ISO/IEC 9506 1.

7.4.2.3 Services Supported Called

This parameter is specified in 8.2.4.3 of ISO/IEC 9506 1.

7.4.2.4 Additional Services Supported Called

This parameter, of type bitstring, shall specify support by the Called MMS user of a set of additional services that are defined in this part of ISO/IEC 9506 for use in the presentation context derived from the abstract syntax defined in this part of ISO/IEC 9506 on the application association.

The value of the parameter in the indication primitive shall specify the intersection of the set of additional services supported by the Called MMS user and the set of additional services supported by the MMS provider.

The assignment of a service to an individual bit of the bitstring type is specified in 7.4.3. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate nonsupport. Any additional bits shall be ignored.

Support for confirmed services shall be defined as the ability to receive a request indication and properly execute the service procedure defined for the responder role.

If a confirmed service is supported, then a Reject PDU shall not be issued on receipt of a request for that service, except for a protocol error. If a confirmed service is not supported, then a Reject PDU shall be issued on receipt of a request for that service with a reject code of "UNRECOGNIZED SERVICE."

7.4.2.5 Additional CBB Supported Called

This parameter, of type bitstring, shall specify support by the Called MMS user of a set of additional parameter CBBs that are defined in this part of ISO/IEC 9506 for use in the presentation context derived from the abstract syntax defined in this part of ISO/IEC 9506 on the application association.

The value of the parameter in the indication primitive shall specify the intersection of the set of additional CBBs supported by the Called MMS user and the set of additional CBBs supported by the MMS provider.

The assignment of a CBB to an individual bit of the bitstring type is specified in 7.4.3. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate nonsupport. Any additional bits shall be ignored. Additional CBBs are defined in this part of ISO/IEC 9506 in 6.6.

7.4.3 Initiate protocol

The abstract syntax of the Init Request Detail and Init Response Detail parameters shall be specified by the InitRequestDetail and InitResponseDetail types respectively. These types are specified below and described in the paragraphs that follow. Subclause 5.5 of ISO/IEC 9506 2 describes the derivation of all parameters for which explicit derivations are not provided in this subclause.

```
InitRequestDetail ::= SEQUENCE {
  proposedVersionNumber           [0] IMPLICIT Integer16,
  proposedParameterCBB            [1] IMPLICIT ParameterSupport
                                  Options,
  servicesSupportedCalling        [2] IMPLICIT ServiceSupportOptions,
  additionalSupportedCalling      [3] IMPLICIT AdditionalSupport
                                  Options,
  additionalCbbSupportedCalling   [4] IMPLICIT AdditionalCbbOptions,
  privilegeClassIdentityCalling   [5] IMPLICIT VisibleString
  }
InitResponseDetail ::= SEQUENCE {
  negotiatedVersionNumber         [0] IMPLICIT Integer16,
  negotiatedParameterCBB          [1] IMPLICIT ParameterSupport
                                  Options,
  servicesSupportedCalled         [2] IMPLICIT ServiceSupportOptions,
  additionalSupportedCalled       [3] IMPLICIT AdditionalSupport
                                  Options,
  additionalCbbSupportedCalled     [4] IMPLICIT AdditionalCbbOptions,
  privilegeClassIdentityCalled     [5] IMPLICIT VisibleString
  }
AdditionalSupportOptions ::= BITSTRING {
  -- Bits 0 - 3 are reserved
  initiateUnitControlLoad             (4),
  unitControlLoadSegment              (5),
  unitControlUpload                   (6),
  startUnitControl                    (7),
  stopUnitControl                     (8),
  createUnitControl                   (9),
  addToUnitControl                    (10),
  removeFromUnitControl               (11),
  getUnitControlAttributes            (12),
  loadUnitControlFromFile             (13),
  storeUnitControlToFile              (14),
  deleteUnitControl                   (15),
  defineEventConditionList            (16),
  deleteEventConditionList            (17),
  addEventConditionListReference      (18),
  removeEventConditionListReference   (19),
  getEventConditionListAttributes     (20),
  reportEventConditionListStatus      (21),
  alterEventConditionListMonitoring   (22)
  }
AdditionalCbbOptions ::= BITSTRING {
  DES       (0),
  DEI       (1),
  RECL      (2)
  }
```

7.5 Generalized protocol extensions

The abstract syntax of ISO/IEC 9506 2 shall, in addition to the extensions specified above, be extended as specified below in order to accommodate the PDU extensions required in support of new services and errors defined in this clause.

7.5.1 ConfirmedService Request extensions

The AdditionalService Request choice of the ConfirmedService Request shall be defined as specified below:

```
AdditionalService Request ::= CHOICE {
  initiateUCLoad       [4]  IMPLICIT InitiateUnitControlLoad Request,
  uCLoad               [5]  IMPLICIT UnitControlLoadSegment Request,
  uCUpload             [6]  IMPLICIT UnitControlUpload Request,
  startUC              [7]  IMPLICIT StartUnitControl Request,
  stopUC               [8]  IMPLICIT StopUnitControl Request,
  createUC             [9]  IMPLICIT CreateUnitControl Request,
  addToUC              [10] IMPLICIT AddToUnitControl Request,
  removeFromUC         [11] IMPLICIT RemoveFromUnitControl Request,
  getUCAttributes      [12] IMPLICIT GetUnitControlAttributes Request,
  loadUCFromFile       [13] IMPLICIT LoadUnitControlFromFile Request,
  storeUCToFile        [14] IMPLICIT StoreUnitControlToFile Request,
  deleteUC             [15] IMPLICIT DeleteUnitControl Request,
  defineECL            [16] IMPLICIT DefineEventConditionList Request,
  deleteECL            [17] IMPLICIT DeleteEventConditionList Request,
  addECLReference      [18] IMPLICIT AddEventConditionListReference
                       Request,
  removeECLReference   [19] IMPLICIT RemoveEventConditionListReference
                       Request,
  getECLAttributes     [20] IMPLICIT GetEventConditionListAttributes
                       Request,
  reportECLStatus      [21] IMPLICIT ReportEventConditionListStatus
                       Request,
  alterECLMonitoring   [22] IMPLICIT AlterEventConditionListMonitoring
                       Request
  }
```

7.5.2 ConfirmedService Response extensions

The ConfirmedService Response choice of the ConfirmedService Response shall be defined as specified below:

```
AdditionalService Response ::= CHOICE {
  initiateUCLoad       [4]  IMPLICIT InitiateUnitControlLoad Response,
  uCLoad               [5]  IMPLICIT UnitControlLoadSegment Response,
  uCUpload            [6]  IMPLICIT UnitControlUpload Response,
  startUC             [7]  IMPLICIT StartUnitControl Response,
  stopUC              [8]  IMPLICIT StopUnitControl Response,
  createUC            [9]  IMPLICIT CreateUnitControl Response,
  addToUC             [10] IMPLICIT AddToUnitControl Response,
  removeFromUC        [11] IMPLICIT RemoveFromUnitControl Response,
  getUCAttributes     [12] IMPLICIT GetUnitControlAttributes Response,
  loadUCFromFile      [13] IMPLICIT LoadUnitControlFromFile Response,
  storeUCToFile       [14] IMPLICIT StoreUnitControlToFile Response,
  deleteUC            [15] IMPLICIT DeleteUnitControl Response,
  defineECL           [16] IMPLICIT DefineEventConditionList Response,
  deleteECL           [17] IMPLICIT DeleteEventConditionList Response,
  addECLReference     [18] IMPLICIT AddEventConditionListReference
                      Response,
  removeECLReference  [19] IMPLICIT RemoveEventConditionListReference
                      Response,
  getECLAttributes    [20] IMPLICIT GetEventConditionListAttributes
                      Response,
  reportECLStatus     [21] IMPLICIT ReportEventConditionListStatus
                      Response,
  alterECLMonitoring  [22] IMPLICIT AlterEventConditionListMonitoring
                      Response
  }
```

7.5.3 Service specific error extensions

The additionalService choice of the serviceSpecificInformation choice of the errorClass component of Service Error shall be defined as specified below:

```
AdditionalService Error ::= CHOICE {
  defineEcl           [0] DefineEventConditionList Error,
  addECLReference     [1] AddEventConditionListReference Error,
  removeECLReference  [2] RemoveEventConditionListReference Error,
  initiateUC          [3] InitiateUnitControl Error,
  startUC             [4] IMPLICIT StartUnitControl Error,
  stopUC              [5] IMPLICIT StopUnitControl Error,
  deleteUC            [6] IMPLICIT DeleteUnitControl Error,
  loadUCFromFile      [7] IMPLICIT LoadUnitControlFromFile Error
  }
```

7.6 End of module

The following END statement closes the module.

```
END
```

8 Standardized objects

There are no standardized objects specified by this part of ISO/IEC 9506. Recommendations for Variable names are specified in annex .

9 Conformance

Conformance to requirements of this Standard shall be at the level of support, or nonsupport, of individual services, service parameters, and service procedures extended by this part of ISO/IEC 9506. Conforming implementations shall report in the PICS which services and service options are supported.

9.1 Conformance classes

9.1.1 Definition of conformance classes

The following conformance classes are defined for this part of ISO/IEC 9056. An overview of the conformance classes is provided by Table 31.

Table 31 — Conformance classes

Conformance Class	Process Control Functions
1	Data Acquisition Parametric Control
2	Program Management
3	Unsolicited Data Acquisition Interlocked Control
4	Configured Data Acquisition Alarming Semaphore
5	Remote MMS Object Instantiation
6	Historian

NOTE — These conformance classes are established to be compatible with similar classes in other companion standards.

9.1.2 Services required for conformance classes

The required services for each conformance class are given in Table 32. Services are grouped by the relevant clause of ISO/IEC 9506 1 or by Clause 7 of this part of ISO/IEC 9506.

An "S" in the table indicates that a conforming system shall support this service as a server. An "X" indicates that a conforming system shall support this service as both a client and as a server.

Table 32 — Service requirements for conformance classes

Service Class	1	2	3	4	5	6
Initiate	S	S	X	X	X	S
Conclude	S	S	X	X	X	S
Abort	S	S	S	S	S	S
Cancel				S	S	
Reject	S	S	S	S	S	S
Status	S	S	X	X	X	S
UnsolicitedStatus			X	X	X	
GetNameList		S	S	S	S	S
Identify	S	S	S	S	S	S
Rename						
GetCapabilityList						
InitiateDownloadSequence		S	S	S	S	
DownloadSegment		S	S	S	S	
TerminateDownloadSequence		S	S	S	S	
InitiateUploadSequence		S	S	S	S	
UploadSegment		S	S	S	S	
TerminateUploadSequence		S	S	S	S	
RequestDomainDownload						
RequestDomainUpload						
LoadDomainContent						
StoreDomainContent						
DeleteDomain		S	S	S	S	
GetDomainAttributes		S	S	S	S	
CreateProgramInvocation		S	S	S	S	
DeleteProgramInvocation		S	S	S	S	
Start		S	S	S	S	
Stop		S	S	S	S	
Resume		S	S	S	S	
Reset		S	S	S	S	
Kill						
GetProgramInvocationAttributes		S	S	S	S	

Table 32 cont. — Service requirements for conformance classes

Service Class	1	2	3	4	5	6
Read	S	S	X	X	X	
Write	S	S	X	X	X	
InformationReport			X	X	X	
GetVariableAccessAttributes			S	S	S	
DefineNamedVariable					S	
DefineScatteredAccess						
GetScatteredAccessAttributes						
DeleteVariableAccess					S	
DefineNamedVariableList					S	
GetNamedVariableListAttributes					S	
DeleteNamedVariableList					S	
DefineNamedType						
GetNamedTypeAttributes						
DeleteNamedType						
TakeControl				S	S	
RelinquishControl				S	S	
DefineSemaphore					S	
DeleteSemaphore					S	
ReportSemaphoreStatus				S	S	
ReportPoolSemaphoreStatus						
ReportSemaphoreEntryStatus				S	S	
AttachToSemaphore						
Input						
Output						

Table 32 cont. — Service requirements for conformance classes

Service Class	1	2	3	4	5	6
DefineEventCondition					S	
DeleteEventCondition					S	
GetEventConditionAttributes					S	
ReportEventConditionStatus				S	S	
AlterEventConditionMonitoring				S	S	
TriggerEvent						
DefineEventAction					S	
DeleteEventAction					S	
GetEventActionAttributes					S	
ReportEventActionStatus				S	S	
DefineEventEnrollment					S	
DeleteEventEnrollment					S	
GetEventEnrollmentAttributes					S	
ReportEventEnrollmentStatus				S	S	
AlterEventEnrollment					S	
EventNotification				S	S	
AcknowledgeEventNotification				S	S	
GetAlarmSummary				S	S	
GetAlarmEnrollmentSummary						
AttachToEventCondition						
ReadJournal						S
WriteJournal						S
InitializeJournal						S
ReportJournalStatus						S
CreateJournal						
DeleteJournal						
ObtainFile						
DataExchange			X	X	X	
GetDataExchangeAttributes			X	X	X	

Table 32 cont. — Service requirements for conformance classes

Service Class	1	2	3	4	5	6
InitiateUnitControlLoad						
UnitControlLoadSegment						
UnitControlUpload						
StartUnitControl						
StopUnitControl						
CreateUnitControl						
AddToUnitControl						
RemoveFromUnitControl						
GetUnitControlAttributes						
LoadUnitControlFromFile						
StoreUnitControlToFile						
DeleteUnitControl						
DefineEventConditionList						
DeleteEventConditionList						
AddEventConditionListReference						
RemoveEventConditionListReference						
GetEventConditionListAttributes						
ReportEventConditionListStatus						
AlterEventConditionListMonitoring						

9.1.3 Parameter CBBs required for conformance classes

The required parameter CBBs for each conformance class are given in Table 33. Parameter CBBs are defined in ISO/IEC 9506 1 or in Clause 7 of this part of ISO/IEC 9506.

An "X" indicates that a conforming system shall support this parameter CBB. For the NEST parameter, the minimum acceptable value for a conforming system is shown.

Table 33 — Parameter requirements for conformance classes

Parameter CBB Class	1	2	3	4	5	6
STR1			X	X	X	
STR2			X	X	X	
VNAM	X	X	X	X	X	
VADR						
VALT				X	X	
VSCA						
TPY						
VLIS						
REAL						
AKEC						
CEI						
DES						
DEI						
RECL						
NEST	0	0	1	1	1	0

9.2 PICS Part One: Implementation information

The provisions of Subclause 18.2 of ISO/IEC 9506 2 shall apply without alteration. A conforming implementation shall complete the PICS part one of ISO/IEC 9506 2.

9.3 PICS part Two: Service CBBs

The provisions of Subclause 18.3 of ISO/IEC 9506 2 shall apply without alteration. A conforming implementation shall complete the PICS part two of ISO/IEC 9506 2. In addition, Table 34 shall be completed to provide additional PICS information relevant to this part of ISO/IEC 9506.

Table 34 — Additional service CBBs

Additional Service Conformance Building Blocks	Server, Client, or Both
InitiateUnitControlLoad	
UnitControlLoadSegment	
UnitControlUpload	
StartUnitControl	
StopUnitControl	
CreateUnitControl	
AddToUnitControl	
RemoveFromUnitControl	
GetUnitControlAttributes	
LoadUnitControlFromFile	
StoreUnitControlToFile	
DeleteUnitControl	
DefineEventConditionList	
DeleteEventConditionList	
AddEventConditionListReference	
RemoveEventConditionListReference	
GetEventConditionListAttributes	
ReportEventConditionListStatus	
AlterEventConditionListMonitoring	

9.4 PICS Part Three: Parameter CBBs

The provisions of Subclause 18.4 of ISO/IEC 9506 2 shall apply without alteration. A conforming implementation shall complete the PICS part three of ISO/IEC 9506 2. In addition, a conforming implementation shall complete Table 35 to indicate support for additional parameter CBBs.

Table 35 — Additional parameter CBBs

Additional Parameter Conformance Building Blocks	Supported
DES	
DEI	
RECL	

9.5 PICS Part Four: Local implementation values

The provisions of Subclause 18.5 of ISO/IEC 9506 2 shall apply without alteration. A conforming implementation shall complete the PICS part four of ISO/IEC 9506 2.

Annex A — Application Association model (Normative)

A.1 General

This annex specifies the structure of the Application Association object to be found in MMS. It is included as an annex until such time as ISO/IEC 9506 1 is amended to include this model.

The object model of the VMD in ISO/IEC 9506 1 is extended to include a List of Application Association references.

A.2 Application Association

The Application Association identifies a specific instance of communication of the VMD with an MMS client.

```
Object: Application Association
Key Attribute: Application Association Identifier
Attribute: AP title of MMS Client
Attribute: Authentication Unit employed (TRUE, FALSE)
Constraint: Authentication Unit employed = TRUE
  Attribute: Authentication Value
Attribute: Other ACSE parameters
Attribute: List of AA Specific named objects
Attribute: List of Transaction Objects
```

A.2.1 Application Association Identifier

This attribute identifies the application association. Since this attribute is never communicated, its form is a local matter.

A.2.2 AP title of MMS Client

This attribute, which is derived from the parameters of the A ASSOCIATE service, identifies the MMS client present on this association.

A.2.3 Authentication Unit employed

This attribute indicates whether (true) or not (false) the authentication unit of the ACSE was used in establishing this association. If this attribute is true, the following attribute also appears.

A.2.4 Authentication Value

This attribute is the value of the Authentication Value parameter of the A ASSOCIATE service as presented by the MMS Client.

A.2.5 Other ACSE parameters

This attribute contains the values of the other parameters of the A ASSOCIATE service as presented by the MMS Client.

A.2.6 List of AA Specific named objects

This attribute contains a list of all the named objects within the VMD that are declared to have AA specific scope and identify this Application Association.

A.2.7 List of Transaction Objects

This attribute contains a list of all the transaction objects in process on this association. This attribute has been moved from the VMD to the Application Association.

Annex B — Block concepts (Informative)

This annex defines blocks in process control systems for use in this part of ISO/IEC 9506, and compares and contrasts this usage to the usage of function blocks in programmable controllers.

B.1 Definition

A block is an instance of one or more functions, where a function is a primitive monitoring or control operation that cannot be further decomposed, and represents an element that may be used in an application for monitoring, or control, or both.

Example:
An example of a function is signal inversion. A second example is a square root extractor.

Each instance represents a unique entity within a control system that implements one or more functions. Functions may be described and categorized by function type.

B.2 Block classification

Blocks are classified as primitive or complex, depending on the number of functions implemented in the block. A primitive block has a single function, while a complex block has at least two functions. The terms "little block" and "big block" are sometimes applied to primitive and complex blocks. Some complex blocks may contain many functions.

Example:
A PID block may have several functions, such as signal conditioning, limiting, and alarming in addition to the PID function.

B.3 Block attributes

Each block, regardless of complexity, has attributes of input parameters, the block function or algorithm type, and output parameters. In complex blocks containing several functions, additional attributes of state and status parameters also exist. Taken together, the block parameters constitute the block database.

Example:
The signal inversion block has an input parameter of the signal to be inverted, an algorithm or function type that performs the inversion, and an output parameter of the inverted signal. A more complex block, such as a PID block, may contain several input parameters, several output parameters, and several state and status parameters.

Block input and output parameters may be simple, in which only the parameter value is provided, or may be structured, in which the parameter value is provided along with status and consistency information.

B.4 Block identification

Each block in a control system has an assigned name, or "tag," that is used for access purposes. The tag provides a way to identify a particular block, and represents an access path to the attributes of the block. Tag names are generally unique within a manufacturing facility.

If the use of the alphabet for block identification is consistent with the constraints of the Identifier type (see 7.6.2 of ISO/IEC 9506 2), the Domain and Program Invocation associated with the block should use the same name. However, often block names contain characters other than those specified in 7.6.2 of ISO/IEC 9506 2. In addition, sometimes block names begin with a digit, which is forbidden for Identifiers. In those cases, the following rules may be applied to develop an appropriate name for the Domain and Program Invocation from the block name.

a) Replace "-" with "_" (i.e., replace hyphens with underscores).

b) Replace "/" with "$".

c) If the first character is a digit, precede it with "$".

B.5 Block structures

Although there may be several input, output, and state/status parameters, there is not a nesting structure or mechanism defined that permits access to a block within a block. Functions internal to the block, such as limiting and alarming in the PID example, are treated as elements of the total function of the block, and all block attributes are treated as attributes of the (outermost) block. Each tag name identifies a single block, with a single (possibly complex) function.

NOTE — While it is possible and desirable to define complex functions in terms of a group of simple functions, once installed in a control system it seems simpler to deal with functions grouped into a single block as a single complex function.

B.6 Block installation

Blocks are installed into a control system through the process of configuration. Blocks are programmed with their execution instructions, sometimes using a language that is based on blocks such as the IEC Function Block programming language, and loaded into a control system. Once installed, blocks exist in the control system until they are explicitly removed or deleted.

B.7 Block security

Because blocks are potentially subject to control from more than one source, such as a system operator and a batch program, blocks are equipped with a mechanism to distinguish between and select from multiple control sources, and to ensure allocation of primary control to a single source. Certain pre-emption mechanisms are also required. The mechanism fulfilling this requirement is the Mode/State parameter.

B.8 Mode and State concepts

Modes and states are used to control and describe the operation of blocks that are capable of executing different phases of the overall algorithm.

Example:
A PID block is typically capable of allowing the output to be directly set by the operator or to set the output based on the value of the inputs and the execution of the algorithm.

> **NOTE —** In real process control systems, the precise definition of what constitutes a mode, versus what constitutes a state, varies from implementation to implementation. This annex is intended to span the scope of those aspects of block operation that are attributed to both modes and states, without drawing a specific distinction.

The descriptions of mode and state concepts that follow represent the principles on which modes and states are based. Not all blocks will necessarily implement all possible modes and states, nor support all of the following concepts. Actual mode and state schemes in use will vary from control system to control system.

B.8.1 Controlling entity

While blocks are essentially autonomous, in that they derive outputs from inputs whenever they execute, they remain fundamentally subject to commands from a higher level entity, termed the controlling entity, which may adjust inputs, outputs, or otherwise affect the block mode or state.

Example:
Examples of controlling entities include system operators, other blocks in a cascade structure, batch programs, supervisory computers, and expert systems.

B.8.2 Pre-emption

Pre-emption represents the ability of one controlling entity to forcibly wrest control of a block from another controlling entity.

Example:
An operator may conclude that a batch program is operating erratically and may place those blocks in control of hazardous processes into a safe, operator controlled mode, or state.

When pre emption occurs, control privilege is assumed by the entity exercising pre-emption. The pre-empted entity is no longer involved with block control unless listed as a participant in the priority of fallback scheme, described in B.8.5. Pre-emption privilege is not affected when pre-emption occurs, but may be modified through a service request that includes a requested change to pre-emption privilege. An act of pre-emption may additionally include concurrent changes to other aspects of a block's mode state.

B.8.3 Control privilege

Control privilege designates the entity currently in control of the block. The holder of control privilege holds a right to alter the mode state of the block. The holder of pre-emption privilege also holds this right. The holder of control privilege also holds the exclusive right to alter the values of input and output parameters at the block when the effect of such alteration is to cause a change in the behaviour of the block.

Example:
When a block is operating in a mode state where the output value is under direct control of a specific operator, the operator is considered to hold control privilege and holds the exclusive right to alter the value of the output parameter.

Changes to block parameters that do not cause a discernable change in block behaviour are permitted at any time, subject to the privilege of the entity requesting the change.

Certain aspects of control privilege may be delegated to another entity without releasing overall control privilege.

B.8.4 Delegated privilege

The controlling entity may choose to delegate privilege for an aspect of the block's control to another controlling entity. Such delegation does not remove the ultimate authority of the controlling entity with control privilege, but temporarily allows one or more additional controlling entities to adjust certain aspects of the block's mode or state.

Example:
Privilege may be delegated for purposes of tuning, alarming, or other block optimization.

If a block with delegated privilege has its control privilege pre-empted by another controlling entity, the delegated privileges remain. Delegated privileges are held until revoked by the entity holding control privilege.

B.8.5 Priority of fallback

Priority of fallback indicates the order in which control privilege will be assumed in the event that the controlling entity with control privilege fails. Priority of fallback does not apply to delegated privilege.

Example:
A batch program terminates abnormally. Control may revert, based on the priority of fallback, to a specific operator console.

B.8.6 Block function

Block function is a way of describing and specifying the behaviour of a block that is capable of exhibiting multiple behaviours over time.

Example:
PID blocks typically allow, under certain conditions, the output to be set directly, or the output to be derived by an algorithm that uses a setpoint as an input parameter, or other behaviours necessary for fully functioned control.

B.8.7 Activation

Active-Inactive describes whether (active) or not (inactive) the block function is actively operating. An inactive block reverts to a safe mode state.

B.8.8 Parameter source selectors

Parameter source selectors are analogous to switches. They are sometimes used at block input and output parameters to select a source of derivation for those parameter values.

Example:
An input source selector on a PID block setpoint parameter could select between the operator, a supervisory computer, a batch program, or a primary block utilized in a cascade structure. An output source selector on a PID block output parameter could select between the block algorithm, the block setpoint, and a supervisory computer.

An implication of the presence of parameter source selectors is the behaviour of updates to parameters from remote sources such as supervisory computers or operators. Updates to parameters will in general succeed, since an update involves writing a variable, but the effect of the update may or may not be seen depending on the mode state of the block.

Example:
An update to the setpoint of a block that is in a mode state where the output is under direct control of a supervisory computer will succeed in updating the setpoint parameter variable, but the effect at the block will not be discernable until the block is placed into a mode state in which the value of the setpoint parameter becomes significant.

NOTE — A logical view of a possible implementation of this scheme is that of an input array for each parameter on which a source selector is provided. An update of the "parameter" from a remote source such as an operator or a remote computer is directed to the appropriate location of the array based on the classification of the remote entity. At each execution, the block "reads" the correct input parameter based on the current mode state. In this logical view, reports of the current value of any block parameter do not use the input array, but read from the actual value in use at the time the report is requested.

B.8.9 Status

The concepts detailed in B.8.1 through B.8.8 may be utilized to command a specific block, or, through the use of a status reporting mechanism, to report the values set by the most recent command. The block status, when reported, also contains other information about the block and about recent actions affecting the block.

B.8.9.1 Initialization

Under certain conditions, including determination of "bad" signal at the setpoint, a block may temporarily force itself into an initialization mode state, in which the block logically executes backwards. In this mode state, the setpoint value is determined by calculating backwards from the output value. Existence in this mode state is not lasting, but may be detectable through status reports.

B.8.9.2 Windup

When the output of a block has reached its maximum or minimum and further changes to the setpoint will have no effect on the output of the block, the block is said to be "wound up." Windup can occur in both the high and low directions and may be reported when block status is reported.

B.8.9.3 Operator pre-empted

In certain application categories it is important to know if control of a block was assumed by an operator by using pre-emption. This condition may be reported as this status parameter.

Example:
In pharmaceutical manufacturing, it is a requirement to determine if an operator assumed control from a batch program.

B.9 Comparison with usage in Programmable Controllers

Table 36 illustrates comparisons and contrasts between application and MMS objects as utilized in process control and in programmable controllers.

Table 36 — Application objects and MMS objects in Process Control and Programmable Controllers

MMS object	Programmable Controller Object	Process Control Object
Domain	Function Type	
Domain	Function Block Type	
Domain	Program Type	Block Algorithm
Domain, Program Invocation	Program Instance	Block
Domain	Global Variables	
VMD-specific MMS Named Variables	Access Path to Global Variables	
Domain-Specific MMS Named Variables	Access Path to Program formal parameters and program global variables	Block Parameters
Domain, Controlling Program Invocation, Unit Control	Configuration	Set of Blocks
Unit Control, Controlling Program Invocation		Set of Blocks (pre-configured)
Domain, Unit Control, Controlled Program Invocation	Resource	Set of Blocks

Annex C — Use of this part of ISO/IEC 9506 for batch processing. (Informative)

NOTE — Terminology used in this annex is based on draft 2 of ISA dS88, available from ISA, 67 Alexander Drive, PO Box 12277, Research Triangle Park, NC 27709 USA. Later drafts may become available following publication of this part of ISO/IEC 9506.

This informative annex suggests how this part of ISO/IEC 9506 could be utilized in support of batch processing.

C.1 Relationship to Process Industries Reference model

Draft 2 of ISA dS88 provides a Process Industries Reference model. This part of ISO/IEC 9506 is believed to be applicable for communications at the Unit level described in this model and above. Below the Unit level, communications are envisioned as utilizing the Fieldbus. Future Linking Devices may someday provide a transparent path for communications between devices utilizing this part of ISO/IEC 9506 and the Fieldbus.

C.2 Recipe management

Control and working recipes may be treated as Domain content. As such, they may be downloaded and uploaded as needed to manage their development and installation in a working control system. The degree of granularity in terms of how Domains are utilized is an application issue. For example, recipes may be completely contained within a Domain or may be split into multiple Domains. For products that are made in different units using the same recipe, it may make sense to store a generalized recipe in one Domain and the unit descriptors in another domain.

C.3 Batch management

Executing batches (with their accompanying procedures operations, phases, control steps, and control instructions) may be treated as Blocks. Procedures may thus incorporate the features provided by Domain and Program Invocation objects and may additionally contain a mode/state. Recipes may be loaded as Domains and linked using the Program Invocation component of the Block object.

C.4 Historical records

Historical records may be treated as MMS journals. If the capabilities of the MMS journal are insufficient, Event Notifications may be sent directly to a higher level device that performs the logging directly.

C.5 Alarm notifications

Alarm notifications may be performed as MMS Event Notifications.

C.6 Batch End reports

Batch End reports may be triggered by the Event Condition associated with the completion of a Program Invocation. The Batch End report is probably best treated as a file and transmitted by means other than this part of ISO/IEC 9506.

C.7 Shared use resources

Where resources are shared and must be coordinated via communication services, MMS semaphores may be utilized.

Annex D — Block symbol definitions (Informative)

NOTE — It is recognized that standardization of specific block types is a necessary step towards a desirable goal of standardization of certain application functions. Additional work is required to attain this goal. Towards this end, the work contained in this annex has been extracted from the body of the text in order to better position it to move to a possible future IEC document that defines and standardizes blocks, block parameter types, and block types.

This annex specifies certain aspects of blocks. Specifically, certain parameters are specified that may be utilized to construct blocks with aspects of functionality defined by these parameters. Extension of the specified block types and functionality are possible through the use of additional block parameters specified in this annex or through the use of additional block parameters not specified in this annex and for which there is not duplication of functionality with block parameters specified in this annex.

The presence of a parameter in a block type implies that the functionality of the block parameter is included in the block. Extension of block parameter lists beyond those specified in this annex are optional.

To be included in the following list, a block parameter was considered to have met the following criteria:

a) The block parameter is commonly available in current process control systems.

b) The block parameter is commonly used by user software.

c) The block parameter is unambiguously defined.

D.1 Domain specific Named Variables

D.1.1 C_ACTION

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_ACTION" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = boolean

Attribute: Access Method

Semantic: Controller Action. Indicates the control action of the controller. When TRUE, an increased set point causes an increased output. When FALSE, an increase in the Process Variable causes an increase in controller output.

D.1.2 C_AGELIM

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_AGELIM" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Age Limit. Period of time in seconds to elapse before age limit bit is set in the QUAL parameter, when a Process Variable has not been measured.

D.1.3 C_ALMDB

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_ALMDB" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: The amount above or below the limit that the Process Variable must travel before the alarm clears.

Example:
A Process Variable that has become greater than C_PVHITP will stay in alarm until the variable drops to C_PVHITP less the dead band.

D.1.4 C_ALMST

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_ALMST" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = bit string 8

Attribute: Access Method

Semantic: Alarm Status variable. Defined as a bit string, with each bit representing (TRUE) the presence or (FALSE) the absence of an alarm. Defined values for the bit string are:

DEVHI	(0),
DEVLO	(1),
HIGH	(2),
LOW	(3),
HIGH HIGH	(4),
LOW LOW	(5),
RATE +	(6),
RATE -	(7)

D.1.5 C_AOUTPUT

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_AOUTPUT" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Analog output of a block or control entity. Expressed in percent of span.

D.1.6 C_APV

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_APV" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: An analog process variable. The measured process value in implied engineering units.

D.1.7 C_ASP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_ASP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Set Point. Desired value of the process variable in engineering units.

D.1.8 C_BIAS

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID,

itemID "C_BIAS" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Controller Bias. Quantity added to the calculated set point input after the ratio has been applied.

D.1.9 C_CGAIN

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_CGAIN" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Controller gain.

D.1.10 C_CRATE

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_CRATE" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Controller rate time in minutes.

D.1.11 C_CRESET

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_CRESET" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Controller reset time. Controller reset time in repeats per minute. A repeat is the error signal resulting from a steady state offset between C_APV and C_ASP.

D.1.12 C_DESC

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_DESC" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = visible string 32

Attribute: Access Method

Semantic: Textual Block description.

D.1.13 C_DEVHIAP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_DEVHIAP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Deviation High Alarm Point. A process event is generated when C_APV becomes greater than   C_ASP by the specified amount. The corresponding boolean variable representing the event condition is set to true.

D.1.14 C_DEVLOAP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_DEVLOAP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Deviation Low Alarm Point. A process event is generated when C_APV becomes less than C_ASP by the specified amount. The corresponding boolean variable representing the event condition is set to true.

D.1.15 C_DOUTPUT

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_DOUTPUT" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = boolean

Attribute: Access Method

Semantic: Digital output of a block. Expressed in boolean terms (TRUE, FALSE).

D.1.16 C_DPV

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_DPV" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = boolean

Attribute: Access Method

Semantic: A digital process variable. The measured process value in boolean terms (TRUE, FALSE).

D.1.17 C_DSP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_DSP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = boolean

Attribute: Access Method

Semantic: Digital Set Point.

D.1.18 C_EDESC

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_EDESC" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = visible string 127

Attribute: Access Method

Semantic: Extended Text Description. This parameter may be used to describe the more important internal features of the block that are not covered by standard parameter names, such as sensor type and controller type.

D.1.19 C_FBCLASS

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_FBCLASS" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = integer 8

Attribute: Access Method

Semantic: Block Class Identification. Specific values for this parameter may be specified in conjunction with standardized block types.

Analog Measurement	0,
Digital Measurement	1,
Symbolic Measurement	2,
Analog Output	3,
Digital Output	4,
Symbolic Output	5,
Regulatory	6,
Alarm	7,
Limit	8

D.1.20 C_IPV

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_IPV" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = integer 8

Attribute: Access Method

Semantic: An integer process variable. The measured process variable in counts.

D.1.21 C_ISP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_ISP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = integer 8

Attribute: Access Method

Semantic: An integer set point. Desired value in counts.

D.1.22 C_LMSTAT

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_LMSTAT" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = bit string 8

Attribute: Access Method

Semantic: Limit Status. Current status information that indicates that the set point or output is currently being limited.

SP HI	(0),
SP LO	(1),
OUT HI	(2),
OUT LO	(3),
SPRTLIM	(4),
SPARE	(5),
SPARE	(6),
SPARE	(7)

SP HI - indicates that the set point is limited in the high direction.

SP LO - indicates that the set point is limited in the low direction.

OP HI - indicates that the output is limited in the high direction.

OP LO - indicates that the output is limited in the low direction.

SPRTLIM - indicates that the rate of change of the set point is currently being rate limited.

SPARE - Reserved for future use.

D.1.23 C_MODE

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_MODE" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = octet string 128

Attribute: Access Method

Semantic: Block Mode structure. The variable implementing the mode state attribute of the Block

D.1.24 C_OBIAS

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_OBIAS" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Output Bias. A bias value added to the controller output after the block calculation. Expressed in terms of percent of output.

D.1.25 C_OPHILM

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_OPHILM" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Output high limit. Within a controller, the output is limited or clamped so as to not exceed a specified point. This limit is applied internally by the controller for restriction of internally generated changes. Expressed in terms of percent of output.

D.1.26 C_OPLOLM

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_OPLOLM" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{ format width 32, exponent width 8}

Attribute: Access Method

Semantic: Output low limit. Within a controller, the output is limited or clamped so as to not exceed a specified point. This limit is applied internally by the controller for restriction of internally generated changes. Expressed in terms of percent of output.

D.1.27 C_PERIOD

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_PERIOD" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Scanning period. Defines the time, in seconds, between execution/scans of the Block.

D.1.28 C_PROCTIM

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_PROCTIM" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = binary time TRUE

Attribute: Access Method

Semantic: Time the process variable was last measured (C_APV, C_DPV, or C_SPV).

D.1.29 C_PVHHAP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_PVHHAP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Analog process variable high high alarm point. When an analog process variable exceeds this value, the corresponding boolean variable representing the event condition is set to true.

D.1.30 C_PVHIAP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_PVHIAP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Analog process variable high alarm point. When an analog process variable exceeds this value, the corresponding boolean variable representing the event condition is set to true.

D.1.31 C_PVLLAP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_PVLLAP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Analog process variable low low alarm point. When an analog process variable is lower than this value, the corresponding boolean variable representing the event condition is set to true.

D.1.32 C_PVLOAP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_PVLOAP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Analog process variable low alarm point. When an analog process variable is lower than this value, the corresponding boolean variable representing the event condition is set to true.

D.1.33 C_PVRCNAP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_PVRCNAP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Negative Rate of Change point. Applied to the process variable, this causes the corresponding boolean variable representing the event condition to be set if the process variable changes by more than the specified rate. Expressed as a negative rate of increase in engineering units per second. The dead band does not apply to this alarm point.

D.1.34 C_PVRCPAP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_PVRCPAP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Positive Rate of Change point. Applied to the process variable, this causes the corresponding boolean variable representing the event condition to be set if the process variable changes by more than the specified rate. Expressed as a positive rate of increase in engineering units per second. The dead band does not apply to this alarm point.

D.1.35 C_QUAL

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_QUAL" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = bit string 16

Attribute: Access Method

Semantic: Quality of the process variable.

out of range	(0),
default value	(1),
inaccurate/uncalibrated	(2),
manually entered	(3),
test mode/maintenance	(4),
no data	(5),
hardware error	(6),
undefined error	(7),
not executing	(8),
age exceeded	(9),
spare	(10),
spare	(11),
spare	(12),
spare	(13),
spare	(14),
spare	(15)

out of range	The value is beyond normal sensor range.
default value	The value supplied is the default. The actual value itself is unavailable.
inaccurate/uncalibrated	The sensor or analog to digital conversion requires calibration. The value may not be accurate.
manually entered	The operator or engineer has entered the value.
test mode/maint	The block is being calibrated, tested, or otherwise maintained.
no data	No data is available for the block. This occurs at similar times to "default value" but when no default value is defined for the block.
hardware error	The hardware associated with the device hosting the block has failed. For example: fuse blown, scanner relays failed.
undefined error	There is a problem with the value that is not defined above.

not executing	The block is not executing.
age exceeded	Process variable may no longer be accurate due to time since last measurement.

D.1.36 C_RANGHI

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_RANGHI" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Range High. 100% engineering units value associated with the analog process variable. This value defines the normal operating range of the analog process variable.

D.1.37 C_RANGLO

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_RANGLO" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Range Low. 0% engineering units value associated with the analog process variable. This value defines the normal operating range of the analog process variable.

D.1.38 C_RATEG

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_RATEG" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Controller rate gain. The gain, dimensionless, for the rate term in a PID controller.

D.1.39 C_RATIO

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_RATIO" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Controller Ratio. Scale factor applied to the calculated set point input of the controller.

D.1.40 C_RPDELTA

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_RPDELTA" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Reporting Delta in Engineering Units. Change in Process Variable required to initiate a new value being sent.

D.1.41 C_SOUTPUT

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_SOUTPUT" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = visible string 16

Attribute: Access Method

Semantic: Symbolic output statement. The required state expressed in symbolic terms (OPEN, CLOSED, STOPPED).

D.1.42 C_SPHILM

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_SPHILM" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Set point high limit. Within a controller, the set point is limited or clamped so as to not exceed the specified point. This limit is applied internally by the controller for restriction of changes generated internally.

D.1.43 C_SPLOLM

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_SPLOLM" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Set point low limit. Within a controller, the set point is limited or clamped so as to not be set below a specified point. This limit is applied internally by the controller for restriction of changes generated internally.

D.1.44 C_SPRTLM

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_SPRTLM" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Set point rate limit. Within a controller, the set point rate of movement is limited or clamped so as to not exceed a specified rate of change. Expressed in engineering units per second. This limit is applied internally by the controller for restriction of changes generated internally.

D.1.45 C_SPV

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_SPV" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = visible string 16

Attribute: Access Method

Semantic: A symbolic process variable. The measured process value, in an explicit textual format (ON, OFF STARTED, RUNNING, etc.).

D.1.46 C_SSP

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_SSP" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = visible string 16

Attribute: Access Method

Semantic: Symbolic Set Point. Desired value of the process variable in words. (EXAMPLES: ON, OFF STARTED, CLOSED).

D.1.47 C_TARGET

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_TARGET" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Target set point. This parameter may be used when a new set point is desired, but a step change to it is undesirable. When TARGET is changed, the server ramps the set point towards the target value at an internally configured rate.

D.1.48 C_TMAX

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_TMAX" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = floating point

{format width 32, exponent width 8}

Attribute: Access Method

Semantic: Maximum Time without report. Maximum amount of elapsed time in seconds for the process variable to go unreported when the process variable change has not exceeded C_RPDELTA.

D.1.49 C_UNITS

Object:Named Variable

Key Attribute: Variable Name = domain specific {

domainID ,

itemID "C_UNITS" }

Attribute: MMS Deletable = FALSE

Attribute: Type Description = visible string 8

Attribute: Access Method

Semantic: Engineering units. Engineering units associated with an analog process variable, for example, meters/second or degrees Celsius. If the engineering units are not available, the value is all blanks.

ISBN: 1-55617-519-1