ICS 25.040.40
J 07

中华人民共和国国家标准

GB/T 16720.2—2005/ISO 9506-2:2003
代替 GB/T 16720.2—1996

工业自动化系统 制造报文规范 第2部分:协议规范

**Industrial automation systems—
Manufacturing message specification—
Part 2:Protocol specification**

(ISO 9506-2:2003,IDT)

2005-01-24 发布　　2005-06-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

前　言

本部分等同采用国际标准 ISO 9506-2:2003《工业自动化系统　制造报文规范　第 2 部分:协议规范》。ISO 9506:2003 在《工业自动化系统　制造报文规范》的总标题下,由两部分组成:

第 1 部分:服务定义;

第 2 部分:协议规范。

《工业自动化系统　制造报文规范》是一个系列标准,国际标准有过几个版本。最早的版本是 1990 年由 ISO、IEC 双编号发布的 ISO/IEC 9506。我国已经将有关部分等效或等同转化为国家标准,见下列目录:

——GB/T 16720.1—1996　工业自动化系统　制造报文规范　第 1 部分:服务定义(eqv ISO 9506-1:1990);

——GB/T 16720.2—1996　工业自动化系统　制造报文规范　第 2 部分:协议规范(eqv ISO/IEC 9506-2:1990);

——GB/T 16720.3—1996　工业自动化系统　制造报文规范　第 3 部分:机器人伴同标准(eqv ISO/IEC 9506-3:1991);

——GB/T 16720.4—1998　工业自动化系统　制造报文规范　第 4 部分:数值控制用伴同标准(eqv ISO/IEC 9506-4:1992);

——GB/T 16721—1996　工业自动化系统　制造报文规范　协议子集规范(eqv ISO/TR 13345:1994);

——GB/T 16979.1—1997　工业自动化系统　制造报文规范　第 1 部分:服务定义　补充件 1:数据交换(idt ISO/IEC 9506-1:1990/Amd.1:1993);

——GB/T 16979.2—1997　工业自动化系统　制造报文规范　第 2 部分:协议规范　补充件 1:数据交换(idt ISO/IEC 9506-2:1990/Amd.1:1993)。

本部分的技术内容和组成结构与 ISO 9506-2:2003 相一致,在编写格式上符合我国国家标准 GB/T 1.1—2000《标准化工作导则　第 1 部分:标准的结构和编写规则》。只是根据我国国家标准的制定要求,做了如下编辑性的改动:

a) 大写的英文缩写保留英文原名,去掉了 ISO 前言。

b) 对正文中的有关英文参数、类型、名称、函数等,为了便于阅读和易于使用,并与原国际标准保持一致;同时,又考虑到要便于用户了解其英文所表示的含意,在本部分中,均引用英文,而在第一次出现时,将中文译名括起来放在英文原文之后。

c) 将“本国际标准”和 ISO 9506 改为“GB/T 16720”。将 ISO 9506-1 改为 GB/T 16720.1;将 ISO 9506-2 改为 GB/T 16720.2;将 ISO/IEC 9506-1:1990 改为 GB/T 16720.1—1996;将 ISO/IEC 9506-2:1990 改为 GB/T 16720.2—1996;将 ISO/IEC 9506-3 改为 GB/T 16720.3 ;将 ISO/IEC 9506-4 改为 GB/T 16720.4;将 ISO/IEC TR 13345 改为 GB/T 16721。

d) 将规范性引用文件中已转化为国家标准的国际标准编号改为国家标准编号,相应的国家标准采用的国际标准版本号和采用程度放在标准名称后的括号内,未转化的国际标准保留。

e) 将 ISO 9506-2:2003 文本后面的“索引”编排为附录 G(资料性附录):中英文对照表。

本部分的附录 A、附录 B、附录 C 是规范性附录,附录 D、附录 E、附录 F、附录 G 是资料性附录。

本部分为推荐性国家标准。

本部分由中国机械工业联合会提出。

本部分由全国工业自动化系统与集成标准化技术委员会(SAC/TC 159)归口。

本部分由北京机械工业自动化所负责起草、北京四方继保自动化有限公司参加起草。

主要起草人:魏文娟、郝淑芬、任雁铭、邬学礼、许瑾等。

本部分所代替标准的历次版本发布情况为:

GB/T 16720.2—1996 和 GB/T 16721—1996。

引　言

本部分为各种制造和过程控制设备提供广泛的服务。它被编制得不仅可自身单独使用，而且可配合伴同标准使用。这些伴同标准描述这些服务的子集在特殊类型设备上的应用。

由制造报文规范(MMS)提供的服务的范围从简单到极其复杂。它并不期望所有的这些服务都能被所有的设备支持。所支持的服务子集在某些情况下受伴同标准限制，而在所有情况下，均可由执行者加以限制。在选择所支持的服务子集时，需考虑的重要特性包括：

a) 服务对设备的适用性；

b) 服务和需求的复杂性；

c) 通过网络提供特定类服务的复杂性与设备复杂性的比较。

安全性考虑

在保密或安全紧要应用中实施 MMS 时，需要采用 OSI 安全性体系结构。本国际标准为鉴别(口令)和访问控制提供简单的工具，需要较高安全度的系统则必须考虑一些超出本部分范围的特性。本部分不为非认定(non-repudiation)提供方便。

服务和需求的复杂性

一些 MMS 服务非常复杂，应该考虑更先进的功能。而在非常简单的应用中使用的设备通常不需要这样先进的功能，因此，它们不必支持这样的 MMS 服务。

关键词

应用互操作	Application Interworking
应用层协议	Application Layer Protocol
信息处理系统	Information Processing Systems
制造通信网络	Manufacturing Communication Network
制造报文规范	Manufacturing Message Specification
数字控制系统	Numerical Control Systems
开放系统互连	Open Systems Interconnection
OSI 参考模型	OSI Reference Model
过程控制系统	Process Control Systems
可编程控制器	Programmable Controller
可编程设备	Programmable Device
机器人控制系统	Robotics Control Systems
虚拟制造设备	Virtual Manufacturing Device

概要

本部分是为便于信息处理系统互连而制定的成套标准之一。它作为相对于其他标准的应用服务元素(ASE)被开放系统互连基本参考模型(GB/T 9387)列入开放系统互连环境的应用层中。

开放系统互连的目标是在互联标准之外，用最低限度的技术协定，使信息处理系统能实现如下互连：

a) 来自不同的设备制造厂；

b) 在不同的管理方式下；

c) 不同的复杂程度；

d) 设备的不同生产年代。

目的

本部分的目的是定义制造报文规范所提供的服务。它与制造报文规范服务定义(GB/T 16720.1)的应用领域密切相关,并处于其中,它利用通信系统提供的服务来传输它的 PDU。

构建 MMS 协议为的是能够定义协议子集。为使制造报文规范广泛提供给不同的应用,在本部分之范围内做一些变动和有选择地使用是必不可少的。这样,最低限度的一致性实施不适用于所有可能的情况。因此,重要的是,利用提供的选择语句或必需的用于指定设备或使用目的的语句来限制对本部分的所有引用。

注:本部分的服务是通用的,并有伴同标准引用,而每一种这些伴同标准都直接面向一类较特定的应用。本部分的服务也可以单独使用(不使用伴同标准)。

值得注意的是,当有序协议序列的数目很大时,利用当前技术验证某个实施在所有环境下均正确执行本部分中定义的协议是不可能的。但借助于测试来确定某个实施在一个有代表性的示例环境中正确执行协议的可靠程序是可能的。

出版

本部分与 GB/T 16720.2—1996 有区别,它更正了与 ASN.1 类型定义和建模结构相关的一些协议错误,以及在正文中的一些印刷错误。

本部分与 GB/T 16720.2—1996 的区别如下:

a) 在 GB/T 16721—1996 中规定 MMS 协议子集的那些内容已包含在本部分中。

b) 修正 1 和修正 2 中的所有内容已合并到正文及技术勘误表中。

c) 用于 GB/T 16720.1—1996 的格式化对象模型为本部分中规定的协议提供类型定义。

d) 在已发布的伴同标准(GB/T 16720.3,GB/T 16720.4 及 ISO/IEC 9506-6)中提供的服务和协议已合并到基本部分中。

通过这样的合并,对于每个伴同标准不再需要单独的抽象语法。尽管利用其他的抽象语法仍然可能逆向兼容,但是,所有伴同标准现在均可以在基本部分的单一抽象语法下进行操作。单独定义GB/T 16720.2第一版第 19 章中的模型已不再需要,因而,该章已取消。

e) MMS 的通信需求已概念化了,所以,根据支持这些需求所必需的抽象服务集来描述 MMS。这个抽象服务集与整套 OSI 通信协议所提供的服务之间的关系在附录中规定。这样,只要提供与这些抽象服务等效的服务,就有可能利用 MMS 正确地跨越另外的通信系统(例如压缩栈的实现)进行操作。

f) 放宽了对作为标识符的字符的限制,允许标识符用数字字符开头,进而,可以完全由数字字符组成。

g) 可见串(visibleString)的许多(不是全部)值已被新产品 MMS 串(MMSString)取代。MMS 串提供一种选择,它可使用取自 ISO 10646 的任意字符串。类似地,这些较通用的串也可用来作为标识符。为了协商对这些较通用串的使用,增加了新参数 CBB。

h) 在程序调用管理一章中介绍了一种新的服务:重新配置程序调用。该服务提供的技术能动态改变运行程序调用的构成范围。

i) 在有名变量和有名类型的对象模型中增加了一个新的域,该域用于描述与有名变量或有名类型相关的语义。它或者是预定义,或者,它的值就是在 DefineNamed Variable 或 DefineNamedType 服务中用于构造有名类型而赋予的名字。只要协商好 Sem(新参数 CBB),就可利用 GetvariableAccessAttributes 或 GetNamedType Attributes 服务将此域报告输出。

j) 为压缩各章,正文内容已重新组织。

k) 正文中已删掉了实数据类型。

l) 将分散访问从正文中删除,移至附录中。

m) 按照 GB/T 16262 中的建议,协议中的所有 EXTERNAL 值以替换为 CHOICE{EXTERNAL,EMBEDDED PDV}。

n) 第一版的 PICS 已被提供配置和初始化信息的一章所取代。这一章为 VMD 及下属对象的一些域(比较少)提供初始化说明,并为其他域(当执行程序支持时)的初始化值提供列表输出。增加了一个新附录(附录 B),它提供一个 ASN.1 模块,适用于该表中的信息的通信。

协议

由于利用了 ASN.1 对象建模技术,所以,协议存在于 3 个分开的模块之中。其中一个是包含在 GB/T 16720.1 中的对象模型部分,另两个则在本部分中定义,它描述所有有效 PDUs 的内容和结构。尽管在某些情况下 ASN.1 的表达形式看上去不同,但通过 GB/T 16720 第一版而产生的 PDUs 与利用这一版而产生的 PDUs 仍然是相同的。为此,本版本继续用主版本号 1 来标识。(变更次版本号,以反映所有对文本的增改)。

有两个例外应注意。

a) 现在,由伴同标准定义的语法扩充用新参数 CBBs 来标识,替代单个的抽象语法。因此,对于每次使用 MMS 来调用伴同标准工具,在启动 PDU 中要作改动。然而,如果不使用伴同标准工具,启动 PDU(InitiatePDU)仍然和第一版所定义的相同。

b) 在修改访问控制(ChangeAccessControl)服务中增加 了一些小的修改,以便于使它与获取名字表(GetNameList)服务及更名(Rename)服务中的相应协议相匹配。

c) 利用 PER(GB/T 16263)的 PDUs 编码与用 GB/T 16720—1996 第一版生成的 PDUs 不可能完全兼容。这是因为用包含某个字(type)的 CHICE 来替代这个字时,用 PER 会导致不同的编码。而对于这两种情况,BER 编码则是相同的。这样,如果 PDUs 包含某个 EXTERNAL(对应于上述第 m)项)元素,那么,它们将被替换为导致不同 PER 编码的 CHICE。

ASN.1 模块

在 GB/T 16720 中定义的 ASN.1 模块可以从 ISO TC 184 SC4 秘书处以计算机可读的形式获得。这些模块有两种格式:一种是已出版的格式,另一种则是带有 IF-ENDIF 括号的格式。

为获得这些文件,网址是:

〈http://forums.nema.org:8080/~iso_tc184_sc5〉

工业自动化系统
制造报文规范
第2部分:协议规范

1 范围

制造报文规范是一个工业自动化应用层的标准,以支持计算机集成制造(CIM)环境中可编程设备的双向报文通信。

1.1 规定

本部分规定:

a) 单个协议的过程,用于将数据和控制信息从一个应用实体传输到MMS-上下文中的一个同层应用实体。

b) 在MMS上下文中通信时,应用实体使用的服务的选取方法。

c) 用于传输数据和控制信息的制造报文规范协议数据单元的结构。

1.2 过程

过程按以下几方来定义:

a) 通过交换制造报文应用协议数据单元进行的同层应用实体之间的交互。

b) 在同一系统中,通过交换MMS原语进行的MMS-提供者与MMS-用户之间的交互。

c) MMS-提供者与底层通信系统所提供的抽象服务之间进行的交互。

1.3 适用性

这些过程适用于在OSI参考模型的应用层中,支持MMS系统之间的通信实例,这些系统需要在开放系统互连环境中的互连能力。

1.4 一致性

本部分还对执行这些过程的系统规定了一致性要求。但它不包含用于验证是否符合这些要求的测试。

2 规范性引用文件

下列文件中的条款通过GB/T 16720的本部分的引用而成为本部分的条款。凡是注日期的引用文件,其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本部分,然而,鼓励根据本部分达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件,其最新版本适用于本部分。

GB/T 1988—1998 信息技术 信息交换用七位编码字符集(eqv ISO/IEC 646:1991)

GB/T 9387.1—1998 信息技术 开放系统互连 基本参考模型 第1部分:基本模型(idt ISO/IEC 7498-1:1994)

GB/T 9387.2—1995 信息处理系统 开放系统互连 基本参考模型 第2部分:安全体系结构(idt ISO 7498-2:1989)

GB/T 9387.3—1995 信息处理系统 开放系统互连 基本参考模型 第3部分:命名与编址(idt ISO 7498-3:1989)

GB/T 15695—1995 信息处理系统 开放系统互连 面向连接的表示服务定义(idt ISO 8822:1988)

GB/T 16687—1996 信息处理系统 开放系统互连 联系控制服务元素协议规范(idt ISO 8650:1988)

GB/T 16688—1996 信息处理系统 开放系统互连 联系控制服务元素服务定义(idt ISO/IEC 8649:1988)

GB/T 16720.1 工业自动化系统 制造报文规范 第1部分:服务定义(idt ISO 9506-1)

GB/T 17176—1997 信息技术 开放系统互连 应用层结构(idt ISO/IEC 9545:1994)

GB/T 17967—2000 信息技术 开放系统互连 基本参考模型 OSI服务定义约定(idt ISO/IEC 10731:1994)

ISO/IEC 8824-1:1998 信息技术 抽象语法记法1(ASN.1) 基本记法规范

ISO/IEC 8824-2:1998 信息技术 抽象语法记法1(ASN.1) 信息对象规范

ISO/IEC 8825-1:1998 信息技术 ASN.1编码规则:基本编码规则(BER)、典型编码规则(CER)和特异编码规则(DER)的规范

ISO/IEC 8825-2:1998 信息技术 ASN.1编码规则:打包编码规则(PER)的规范

ISO/IEC 10646-1:2000 信息技术 通用多八位编码字符集(UCS) 第1部分:体系结构和基本多语种平面

ANSI/IEEE 754:1985 二进制浮点运算的IEEE标准

3 术语和定义

本部分采用下列术语及定义。

注:本章中的定义使用第4章定义的缩略语。

3.1 参考模型定义

本部分以开放系统互连基本参考模型GB/T 9387制定的概念为基础,并采用该标准中定义的下列术语:

a) 应用实体(application-entity);
b) 应用过程(application-process);
c) 应用服务元素(application Service element);
d) 开放系统(open system);
e) (N)-协议((N)-protocol);
f) (N)-协议数据单元((N)-protocol-data-unit);
g) (N)-服务访问点((N)- Service-access-point);
h) (N)-层((N)-layer);
i) 系统(system)。

3.2 服务约定定义

当下述术语应用于制造报文规范时,本部分采用OSI服务定义约定(GB/T 17967)中定义的下列术语:

a) 确认(confirm);
b) 指示(indication);
c) 原语(primitive);
d) 请求(request);
e) 应答(response);
f) 服务原语(service primitive);
g) 服务提供器(service provider);
h) 服务用户(service user)。

3.3 抽象语法记法定义

本部分采用抽象语法记法 1(ASN.1)规范(GB/T 16262.1)中定义的下列术语：

a) 值(value)；
b) 类型(type)；
c) 简单类型(simple type)；
d) 结构类型(structure type)；
e) 分量类型(component type)；
f) 标记(tag)；
g) 加标记(tagging)；
h) 类型(或值)引用名(type (or value)reference name)；
i) 字符串类型(character string type)；
j) 布尔类型(boolean type)；
k) 真(true)；
l) 假(false)；
m) 整型(interger type)；
n) 位串型 (bitstring type)；
o) 八位位组串型(octetstring type)；
p) 空类型(null type)；
q) 序列类型(Sequence type)；
r) 后续类型(Sequence-of type)
s) 标记类型(tagged type)；
t) 选择类型(choice type)；
u) 选取类型(selection type)；
v) 实型(real type)
w) 对象标识符类型(object identifier type)；
x) 模块(module)；
y) 产生式(production)；
z) ASN.1 编码规则；
aa) ASN.1 字符集；
ab) 外部类型(external type)。

3.4 其他定义

本部分还采用下列定义：

3.4.1

应用关联专用 AA-specific

用于描述一个对象的修饰语，表示该对象名的范围限定为单一的应用关联(即该对象名仅在定义该对象的相关应用关联上引用)。

3.4.2

属性 Attribute

具有确定意义的数据元素以及它可能的取值集合的说明。

3.4.3

MMS 受叫户 Called MMS-user

发出 Initiate.response 服务原语的 MMS 用户。

3.4.4

MMS 呼叫户　Calling MMS-user

发出 Initiate. request 服务原语的 MMS 用户。

3.4.5

客户　Client

为了某一特定目的，通过服务请求实例使用 VMD 的同层通信实体。

3.4.6

一致性构造块　conformance building block（CBB）

描述 MMS 一致性需求的基本单元。

3.4.7

数据　data

已赋予或可赋予意义的一种表达（如字符）。

3.4.8

域　domain

表达用于特定目的的一个 VMD 的能力子集的抽象对象。

3.4.9

域专用　Domain-specific

用于描述一个对象的修饰语，表示该对象名的范围限定为单个域（即，该对象名可以在引用这个域的 VMD 所建立的所有应用关联上引用）。

3.4.10

下载　download

通过给 MMS 用户装载数据来传递域及其下属对象的内容的传送过程。

3.4.11

事件管理　event management

对事件条件、事件活动、事件登录及事件条件列表的管理。

3.4.12

文件　file

有明确有名的、并有公用属性集的信息集合。

3.4.13

文件操作　file operation

开放系统之间文件的传递，部分文件内容的检查、修改和更换，或者对文件及其属性的管理。

3.4.14

文件库　file store

驻留在一个特定开放系统中的一批有组织的文件，包括它们的属性和名称。

3.4.15

信息　information

数据及其所传达含义的组合。

3.4.16

无效 PDU　invalid PDU

在结构上或（和）在意义上不遵循本部分要求的 PDU。

3.4.17

日志　journal

在检索时可逻辑排序的、被记录的、带时标的事件变迁、变量数据及/或注释的集合。

3.4.18

本地事务 local matter

在制造报文规范中系统作出的涉及其行为的判定。而不受 GB/T 16720 要求的限制。

3.4.19

制造报文协议机 manufacturing massage protocol machine (MMPM)

实现本部分所指定的过程的抽象机。

3.4.20

MMS-上下文 MMS-context

在应用关联生存期内使用的 MMS 服务元素的说明及通信语义。

3.4.21

MMS 提供者 MMS-provider

通过交换 MMS PDU,从概念上讲,提供 MMS 服务的应用实体的那一部分。

3.4.22

MMS 用户 MMS-user

从概念上讲,调用制造报文规范的应用过程的那一部分。

3.4.23

被监控事件 monitored event

被检测到的一个事件的条件状态的变化。

3.4.24

网络触发事件 network-triggered event

由客户明确请求而引发的事件。

3.4.25

操作员站 operator station

表达与 VMD 相关联的设备的抽象对象,这些设备提供与操作员的输入/输出交互。

3.4.26

预定义对象 predefined object

通过使用不同于 MMS 服务的一些机制来说明的对象。

3.4.27

程序调用 program invocation

由域集组成的、表示一个动态元素的抽象对象,该动态元素紧密地对应于多任务环境中的一个执行线程。

3.4.28

协议错误 protocol error

不遵循本部分要求的 PDU。

3.4.29

接收 MMPM receiving MMPM

接收 MMS PDU 的 MMPM。

3.4.30

MMS 接收户 receiving MMS-user

接收指示服务原语或确认服务原语的 MMS 用户。

3.4.31

远程设备控制和监控 remote device control and monitoring

对安装在服务请求应答方的设备的操作和状态检查。

3.4.32

MMS 请求户 requesting MMS-user

对某个服务发出请求服务原语的 MMS 用户。

3.4.33

MMS 应答户 responding MMS-user

对某个服务发出应答服务原语的 MMS 用户。

3.4.34

信标 semaphore

与逻辑或物理资源相关联的概念锁，它只允许锁的占有者访问这些资源。

3.4.35

信标管理 semaphore management

对信标的控制。

3.4.36

发送 MMPM sending MMPM

发送 MMS PDU 的 MMPM。

3.4.37

MMS 发送户 sending MMS-user

发出请求服务原语或应答服务原语的 MMS 用户。

3.4.38

服务器 server

对一个特定服务请求实例起 VMD 代理作用的同层通信实体。

3.4.39

标准化对象 standardized object

在 GB/T 16720.1 中或在 MMS 伴同标准中定义的对象实例。

3.4.40

类型 type

可以通过变量值进行传递的一批值的抽象描述。

3.4.41

上载 upload

通过从远程用户装载数据来传递域内容及其下属对象的过程，用这种方式允许连续下载。

3.4.42

有效 PDU valid PDU

在结构及意义上遵守本部分的 PDU。

3.4.43

变量 variable

通过唯一名称或描述引用的一个或多个数据元素。

3.4.44

变量访问 variable access

对 VMD 中定义的变量或变量的分量的检查或修改。

3.4.45

虚拟制造设备 virtual manufacturing device(VMD)

真实制造设备上特定资源和功能集的抽象表达，以及该抽象表达到真实制造设备的物理和功能方面的映射。

3.4.46

VMD 专用 VMD-specific

用于描述一个对象的修饰语，该对象的名称范围限定为单个 VMD(即，该名称可以为与这个 VMD 建立的所有应用关联引用)。

4 缩略语

AA	应用关联
ACSE	关联控制服务元素
AE	应用实体
AP	应用过程
APDU	应用协议数据单元
ASE	应用服务元素
ASN.1	抽象语法记法 1
CBB	一致性构造块
CIS	配置与初始化说明
FRSM	文件读状态机
MMPM	制造报文协议机
MMS	制造报文规范
OSI	开放系统互联
PDU	协议数据单元
ULSM	上载状态机
VMD	虚拟制造设备

5 约定

5.1 服务约定

本部分采用 OSI 服务定义约定(GB/T 17967)中的陈述性约定。模型定义 MMS-用户和 MMS-提供者之间的交互。在 MMS-用户和 MMS-提供者之间，利用可传递参数的服务原语进行信息传输。

5.2 数制基

本部分若无另加说明，则所有数制均采用十进制表示。

5.3 记法

本部分采用 ISO/IEC 8824(ASN.1 规范)中定义的抽象语法记法。为了与 ASN.1 标准的含义和要求保持一致，所有的类型引用符均以大写字母开头，而所有的值引用则以小写字母开头。

5.4 支持产生式

在本部分不同章中引入的支持产生式，如果只是在一个地方采用，则在引用它们的地方加以说明。当支持产生式在不同的地方被多次引用时，它们被集中在最相关章节的末尾加以说明。在任何情况下，在本部分的结尾处均可查到带页号的产生式索引。

5.5 传递参数

各种 MMS 服务的参数通过服务请求 PDU 从请求原语传递到指示原语，或者，通过服务应答 PDU，从应答原语传递到确认原语，无需 MMS-提供者对这些参数施加其他的动作。

5.5.1 传递请求参数

由类型引用名标识的类型应具有来自服务请求原语的同名参数，并且，当发出该请求原语时，它将作为同名的参数出现在服务指示原语中。在请求原语、指示原语和请求 PDU 中，该参数的值在语义上是相同的。

如果该参数是任选的，并且，在请求服务原语中被略去，那么，它也不出现在请求 PDU 中。如果可选参数不出现在请求 PDU 中，那么，它也不应出现在指示服务原语中。

如果一个参数在请求 PDU 中有缺省值，并且，该值在请求服务原语中给出，那么，该参数可以不出现在请求 PDU 中。

如果一个参数在请求 PDU 中有缺省值，并且，这个参数不出现在请求 PDU 中，那么，这个缺省值应在指示服务原语中规定。

5.5.2 传递应答参数

由类型引用名标识的类型应具有来自服务应答原语的同名参数，并且，当发出该原语时，它应作为同名参数出现在服务确认原语中。在应答原语、确认原语和应答 PDU 中，该参数的值在语义上是相同的。

如果该参数是任选的，并且，从应答服务原语中略去，那么，它不应出现在应答 PDU 中。如果一个可选参数不出现在应答 PDU 中，那么，它也不应出现在确认服务原语中。

如果一个参数在应答 PDU 中有缺省值，并且，该缺省值在应答服务原语中给出，那么，这个参数可以不出现在应答 PDU 中。

如果一个参数在应答 PDU 中有缺省值，并且，该参数不出现在应答 PDU 中，那么，这个缺省值应在确认原语中规定。

5.5.3 参数中的枚举值

对于服务描述中具有枚举值的那些参数，为相应协议参数规定的值应取自是包含此参数的服务原语的同名的值(参见 5.5)。在服务原语、结果 PDU 以及通过接收服务原语而产生的服务原语中传递的值在语义应是相同的。

注：在本部分中，通过在服务原语和协议中使用相同的名字来标识这些值之间的对应关系。在服务规范中，这种值用大写字母说明。在协议规范中，名字的大小写的选择按照协议中用法，未用大写字母的名字要加以注释满足 ASN.1 的语法要求。

5.6 否定确认

在处理 MMS 应答户提出的服务请求出现错误时，大部分确认型 MMS 服务提供否定确认。这种否定确认用服务应答原语中的 Result(－)参数和“ ErrorType”参数标识。Result(－)参数和 ErrorType 参数应出现在确认服务原语中，他们与应答原语中的那些参数在语义上是相同的。

用于否定确认的抽象语法是该服务的带“error”域的 ErrorPDU，这个“error”域从应答服务原语的“Problem”参数导出而来。

5.7 服务请求的修饰符

MMS 服务允许修饰符与服务请求实例一起使用。

在使用修饰符的服务请求实例中，在 RequestPDU 中指定的修饰符与请求服务原语中指定的修饰符在语义上等价，并且顺序相同。指示原语应包含一张修饰符表，表中的修饰符与 RequestPDU 中的修饰符语义上等价，顺序相同。

5.8 错误的说明

对于本部分正文中给出的每个服务，因使用这些服务而导致的错误并未与该服务的协议一起给出，而是在单独的一章中说明。

5.9 MMS 呼叫户和 MMS 受叫户

本部分使用术语 MMS 呼叫户和 MMS 受叫户。MMS 呼叫户是发送 Initiate.request 服务原语的 MMS 用户，MMS 受叫户是发送 Initiate.response 服务原语的 MMS 用户。

注：在 MMS 中使用术语“受叫”(Called)与 OSI 中术语的一般用法不同。MMS 使用术语“受叫”相当于 OSI 使用术语“应答”(responding)，介绍这一区别是为了避免与下面给出的 MMS 请求/应答户相混淆。

5.10 MMS 发送户和接收户以及 MMPM

本部分使用术语 MMS 发送户和 MMS 接收户。MMS 发送户是发送请求或应答服务原语的 MMS

用户。MMS 接收户是接收指示服务原语或确认服务原语的 MMS 用户。

注：在完成确认型 MMS 服务的过程中，两个 MMS 用户既是发送者，同时又是接收者。第一个 MMS 用户发送请求并接收确认，同时，第二个 MMS 用户接收指示，并发送应答。注意这一点是非常重要的。

本部分使用术语发送 MMPM 和接收 MMPM。发送 MMPM 是发送一个 MMS PDU 的 MMPM，接收 MMPM 是接收一个 MMS PDU 的 MMPM。

5.11 MMS 请求户和 MMS 应答户

本部分使用术语 MMS 请求户和 MMS 应答户。MMS 请求户是为一项服务发送请求服务原语的 MMS-用户，而 MMS 应答户则是为此发送应答服务原语的 MMS 用户。

注：重要的是应注意，在这里，术语 MMS 应答户的使用不同于 ACSE 中和其他标准中术语应答实体的使用。在那些标准中，该术语用来引用响应连接请求的实体。

5.12 服务的客户和服务器

为描述 MMS VMD 模型，本部分采用术语客户和服务器。服务器被定义为对一个特定服务请求实例起 VMD 作用的同层通信实体。客户是为某些特定目的，通过服务请求实例使用 VMD 的同层通信实体。VMD 模型首先是用于描述服务器的活动，从而描述客户可使用的命令及应答。在应用关联的生存期内，实终端系统可以接受客户角色或服务器角色，或同时接受二者。

5.13 ASN.1 定义

本部分第 7 到 23 章中提供的 ASN.1 定义是 ASN.1 模块“ISO-9506-MMS-1”的一部分。本部分附录 A 中提供的 ASN.1 定义是 ASN.1 模块“MMS-Environment-1”的一部分。本部分附录 B 中提供的 ASN.1 定义是 ASN.1 模块“MMS-SCI-Module-1”的一部分。本部分附录 C 和附录 E 中提供的 ASN.1 定义是 ASN.1 模块“ISO-9506-1A”的一部分。为了使文档易于阅读，将开始和结束语句省略，这两个语句用于标明所提供的每一 ASN.1 定义是其相应模块的一部分。所提供的每个 ASN.1 定义在其开头隐含语句：ModuleName DEFINITIONS::=BEGIN。在其结尾包含关键字“END”。此处，ModuleName 是 ASN.1 模块的名字，而该定义是此模块的构成部分。

注：ISO-9506-MMS-1 表示由本部分提供的 MMS 核心抽象语法的主版本号 1。

5.14 协议子集记法

本部分推荐的记法采用嵌入 ASN.1 的预处理程序语言的形式。在概念上，它与 C 语言的预处理程序非常类似。

在该记法中只有三个命令：

——IF(<自变量表>)；

——ELSE；

——ENDIF。

IF 命令要有一个自变量表(括于圆括弧中)，这些自变量是一致性构造块的名字，或者是服务或参数。必须要有一个或多个自变量。如果出现一个以上的自变量，它们之间用一个或多个空格分隔。如果作为 MMS 启动交换的结果，支持相应的服务或参数构造块，那么，被当作布尔变量处理的自变量取值为“真”。如果只有一个自变量，并且，当且仅当支持这样有名的一致性构造块时，从该 IF 语句以下，直至 ELSE 语句，或者(如果没有 ELSE 语句)直至相匹配的 ENDIF 语句之间的所有行应被包含在所产生的 ASN.1 定义中。如果有一个以上的自变量，那么，当自变量表中的任何一个一致性构造块被支持时(这可以看作是一致性构造块的“逻辑或”功能)，IF 语句之后的行应被包含在 ASN.1 定义中。

IF 语句可以任意深度地嵌套。

IF(X)
IF(Y)

的作用是当且仅当 X 和 Y 都为“真”时，即一致性构造块 X 和 Y 都包括时(这可以看作是一致性构造块的“逻辑与”功能)，应包含这些命令之后的行。

ELSE 语句允许当一致性构造块不为“真”时包含 ASN.1 语句。它的用法类似于编程语言中

ELSE 的标准用法。

ENDIF 语句用于标明 IF 语句或 ELSE 语句范围的结束。每个 IF 语句必须有一个与之配对的 ENDIF 语句。

5.15 有效协议的确定

对于服务和参数 CBBs 的任意特定组合，其协议的有效作用由下列步骤确定：

a) 对于由启动交换说明的或协商的每个服务 CBB 和参数 CBB，相应自变量设置为“真”。

b) 对本部分中规定的所有 ASN.1 模块进行处理。对每个 IF 语句，计算自变量。

   i) 如果自变量中的某个元素为“真”，则将 IF 语句之后的语句保留，直至相匹配的 ENDIF 语句或 ELSE 语句(如果出现的话)之前的语句保留。废弃 ELSE 语句之后的所有语句，直至相匹配的 ENDIF 语句之间的所有语句。

   ii) 如果自变量中的所有元素都为“假”，那么，废弃其后的直至相匹配的 ELSE 或 ENDIF 之前的所有语句。如果存在 ELSE 语句，则保留该语句之后，直至相匹配的 ENDIF 之前的语句。

   iii) 废弃 IF 语句、与其相匹配的 ENDIF 语句，以及 ELSE 语句(如果存在的话)。结果应是一个没有 IF、ELSE 和 ENDIF 的 ASN.1 模块。

c) 在每个产生式中，如果出现逗号后紧跟一个右括号，则用一个右括号替代，即，将这样的逗号删除。

d) 形成仅包含第 1 产生式(即第 7 章中说明的产生式 MMSpdu)的产生式 ASN.1 工作模块。

e) 将工作模块中引用的、还未包含在该模块中的产生式加到 ASN.1 工作模块中。

f) 重复步骤 e)，直至没有新的产生式可添加。

最终的 ASN.1 模块是对这个 CBBs 组合有效的模块。接收一个与该模块不相符的 PDU 会导致拒绝)。

6 协议过程元素

本章描述与发送 MMPDUs 和接收 MMPDUs 相关的协议过程元素，以及它们在 MMS 用户到 MMS 提供者的边界上与服务原语事件的关系。

6.1 描述的约定

本章的图使用标准状态图描述机制。下面简要介绍这种机制。所有状态图都是从 MMS 提供者的观察角度给出的。

每个状态用一个方框表示。方框内给出状态名。每个箭头表示一个状态的转换进或转换出。箭头指向输出状态，它是作为转换结果而进入的一种状态。

每个转换用引起该转换的输入活动和根据转换产生输出活动来标注。输入在输出之上给出，并用实水平线与输出相隔。

加“+”号的服务原语表示该服务原语包含 Result(+)参数。加“-”号的服务原语表示该服务原语包含 Result(-)参数。

6.2 进入和退出 MMS 环境

启动、结束和异常中止服务提供进入和退出 MMS 环境的机制。这些服务的模型(它描述可允许的事件序列)在 GB/T 16720.1 第 8 章描述。

6.3 MMS 环境中的操作

一旦进入 MMS 环境，随时都会有许多待完成的服务。GB/T 16720 对每一个这样的服务请求实例分别描述其状态图。

注：本部分的另一些章节定义了对可允许的服务原语序列的附加限制，它将进一步约束 MMS 用户。

6.3.1 确认的 MMS 服务

本章描述可以在 MMS 环境中调用的、所有确认服务的状态转换。该服务集由利用 Confirmed-RequestPDU 而申请的所有服务组成。

图 1 和图 2 所示的状态转换图可适用于这些服务，并且，分别可应用于每个服务请求的各个实例。在任一给定时刻，可以有多个并发的服务请求实例，应遵循本部分另一章中规定的排序规则。

与单个 MMS 确认服务实例相关联的所有 PDUs（这些 PDUs 是 Confirmed-RequestPDU、Confirmed-ResponsePDU、Confirmed-ErrorPDU、Cancel-RequestPDU、Cancel-ResponsePDU、Cancel-ErrorPDU 以及 RejectPDU）应在同一个表示上下文中发送。

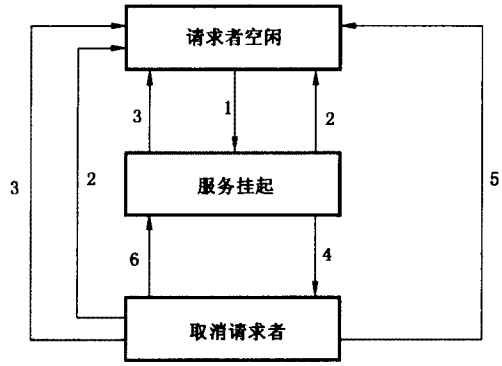

图 1 从服务请求者的角度看确认服务请求

转化：

1- $\dfrac{\text{x. request}}{\text{confirmed-RequestPDU(x)}}$

2- $\dfrac{\text{Confirmed-ResponsePDU(x)}}{\text{x. confirm+}}$

3- $\dfrac{\text{Confirmed-ErrorPDU(x)}}{\text{x. confirm−}}$

4- $\dfrac{\text{Cancel. request}}{\text{Cancel-RequestPDU}}$

5- $\dfrac{\text{Cancel-ResponsePDU and Confirmed-ErrorPDU(x)}}{\text{Cancel. confirm+ and x. confirm−}}$

6- $\dfrac{\text{Cancel-ErrorPDU}}{\text{Cancel. Confirm−}}$

6.3.1.1 服务请求者

在 6.3.1 的转换 5 中，接收 Cancel-ResponsePDU 与接收 Confirmed-ErrorPDU(x)的顺序是没有关系的。

本条款从服务请求者的角度描述一个确认 MMS 服务请求的进行步骤。发出服务请求原语之前，服务被认为处于“请求者空闲”状态。一旦接收到对某个 MMS 确认服务的请求原语，MMS 提供者发送一个 Confirmed-RequestPDU（指定调用 ID，该 ID 在应用关联上单值地标识这个服务请求实例），并进入“服务挂起请求者”状态。

一旦接收到 Confirmed-ResponsePDU（指定预先请求的服务以及用于指定服务实例的调用 ID），MMS 提供者向 MMS 用户发出包含 Result(+)参数的确认服务原语（指定服务类型和预先请求的调用 ID），然后，发生进入“请求者空闲”状态的状态转换。

一旦接收到 Confirmed-ErrorPDU（指定预先请求的服务和指定服务实例的调用 ID），MMS 提供者向 MMS 用户发出包含 Result(−)参数的确认服务原语（指明服务类型和预先请求的调用 ID），然后，发生进入“请求者空闲”状态的状态转换。

一旦接收到 MMS 用户的取消服务原语，MMS 提供者发送一个 Cancel-RequestPDU，它包含要取消的服务请求的调用 ID(该信息是从取消请求原语参数中获得的)。然后，进入“由请求者取消”状态。

当接收到 4 种可能的输入活动的任一种时，退出“由请求者取消”状态。

如果接收了 Cancel-ErrorPDU(它指定与取消服务请求的正常实例相匹配的调用 ID)，MMS 提供者向 MMS 用户发出包含 Result(－)参数的取消确认服务原语，然后，返回到“服务挂起请求者”状态。在这种情况下，该取消请求被认为失败。

在取消请求成功的情况，发生下列事件：

a) 接收一个 Cancel-ResponsePDU，它的调用 ID 与取消服务请求的正常实例相匹配；

b) 接收一个 Confirmed-ErrorPDU，它指定被取消的服务的类型，以及与被取消服务相匹配的调用 ID；

c) MMS 提供者向 MMS 用户发出包含 Result(＋) 参数的取消确认服务原语，并对被取消的服务发出包含 Result(－)参数的确认服务原语(并说明取消的原因)；

d) MMS 提供者转换到“请求者空闲”状态。

如果接收到一个 Confirmed-ResponsePDU，它指定了被取消服务的类型和与这个被取消服务相匹配的调用 ID，MMS 提供者对正在进行取消处理的服务发出包含 Result(＋)参数的确认服务原语。在这种情况下，该取消请求被认为是失败的，并接收到用于这个取消服务调用的 Cancel-ErrorPDU。

注：通常，当双路同步对话中的两个 MMS 用户同步地发出对被要取消服务的 Confirmed-ResponsePDU 及 Cancel-RequestPDU 时，就发生上述情况。

如果接收到一个 Confirmed-ErrorPDU，它指定了被取消服务的服务类型(该服务的调用 ID 与被取消的服务相匹配)，同时，错误原因不是错误类别 SERVICE-PREEMPT 和错误代码 CANCEL，则 MMS 提供者对正在进行取消处理的服务发出包含 Result(－) 参数的确认服务原语。在这种情况下，取消请求被认为失败，并接收到用于这个取消服务调用的 Cancel-ErrorPDU。

注：通常，当双路同步对话中的两个 MMS 用户同步地发出对被要取消服务的 Confirmed-ErrorPDU 及 Cancel-RequestPDU 时，发生上述情况。

对错误取消的处理在 6.4 中描述。

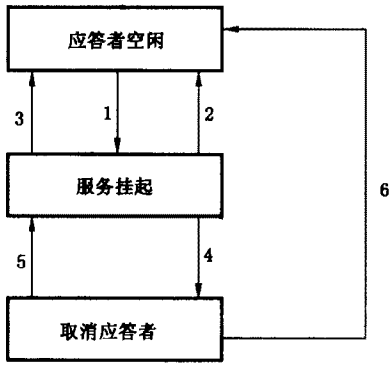

图 2 从服务应答者的角度看确认服务请求

转换：

1- $\dfrac{\text{Confirmed-RequestPDU(x)}}{\text{x. indication}}$

2- $\dfrac{\text{x. response+}}{\text{Confirmed-ResponsePDU(x)}}$

3- $\dfrac{\text{x. response−}}{\text{Confirmed-Error PDU(x)}}$

4- $\dfrac{\text{Cancel. Request PDU}}{\text{Cancel. indication}}$

5- $\frac{\text{Cancel. response}-}{\text{Cancel-ErrorPDU}}$

6- $\frac{\text{Cancel. response}+\text{and x. response}-}{\text{Cancel-ResponsePDU and Confirmed-ErrorPDU(x)}}$

6.3.1.2 服务应答者

在图 2 的转换 6 中，Cancel. response+及 x. response−服务原语的发送顺序是没有关系的。

图 2 从服务应答者的角度描述一个确认 MMS 服务请求的进行步骤。在接收服务的 Confirmed-RequestPDU 之前，服务被认为处于“应答者空闲”状态。一旦接收到上面标识的某个已确认的服务的 Confirmed-RequestPDU 时，MMS 提供者发出指示原语(指定所请求的特定服务和指定服务实例的调用 ID)，并进入“服务挂起应答者”状态。

一旦接收到包含 Result(+) 参数的应答服务原语(指定预先指明的服务和指定服务实例的调用 ID)，MMS 提供者发送一个 Confirmed-ResponsePDU(指定服务类型和调用 ID，它们取自应答原语)。然后，进入状态转换“应答者空闲”状态。

一旦接收到包含 Result(−) 参数的应答服务原语(指定预先指明的服务和指定服务实例的调用 ID)，MMS 提供者发送一个 Confirmed-ErrorPDU(指定服务类型和调用 ID，它们取自应答原语)。状态转换进入“应答者空闲”状态。

一旦接收到 Cancel-RequestPDU(指定与服务实例相匹配的调用 ID)，MMS 提供者发出一个取消指示服务原语，指定被取消的服务请求的调用 ID(该信息取自 Cancel-RequestPDU 参数)。然后，进入“取消服务应答者”状态。

注：当接收到一个 Cancel-RequestPDU，它的调用 ID 与任何一个待完成的服务实例不相匹配时，所发生的活动在 6.4 中说明。

当接收到两种可能的输入活动中的任何一个时，退出“取消服务应答者”状态。这些在下两段描述。

当取消请求在 MMS 应答户一方取得成功时，发生下列事件序列：

a) MMS 应答户向 MMS 提供者发出一个包含参数 Result(+)的取消应答，指定匹配服务实例的调用 ID，同时，对被取消的服务发出包含参数 Result(−)(指定错误类别 SERVICE-PREEMPT 和错误代码 CANCEL)的应答服务原语；

b) MMS 提供者发送一个 Cancel-ResponsePDU 和一个 Confirmed-ErrorPDU，指定被取消的服务实例(以及错误类别 SERVICE-PREEMPT 和错误代码 CANCEL)；

c) MMS 提供者返回到“应答者空闲”状态。

MMS 用户在不发出包含参数 Result(−)(指明错误类别 SERVICE-PREEMPT 和错误代码 CANCEL)的应答服务原语时，不得发出包含参数 Result(+)的取消应答服务原语。反之，MMS 用户在不发出包含参数 Result(+)的取消应答服务原语时，不得发出包含参数 Result(−)(指定错误类别 SERVICE-PREEMPT 和错误代码 CANCEL)的应答服务原语。因此，这两个事件逻辑上是同时发生的。

如果接收到包含 Result(−) 参数的取消应答(指定匹配服务实例的调用 ID)，MMS 提供者发送一个 Cancel-ErrorPDU，并返回到“服务挂起”状态。在这种情况下，取消请求被认为是失败的。

注：对错误的取消请求和无效 PDU 的处理在 6.4 中描述。

6.3.2 无确认的 MMS 服务

本章描述无确认 MMS 服务的操作。这个服务集所定义的服务构成第 7 章中所定义的 UnconfirmedService 选择的服务。

图 3 和图 4 给出的状态转换图可适用于上述每个服务，并且，被分别应用于每一个服务请求的各个实例。

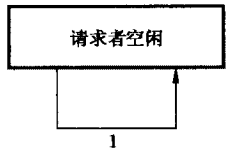

图 3 从服务请求者的角度看无确认服务请求

转换：

$$1- \quad \frac{\text{y. request}}{\text{Unconfirmed-PDU(y)}}$$

6.3.2.1 服务请求者

图 3 从服务请求者的角度描述一个无确认 MMS 服务的进行步骤。发出服务请求原语之前，服务被认为是处于“请求者空闲”状态。一旦接收到对上述某个无确认服务的请求原语，MMS 提供者则发送一个 UnconfirmedPDU(指明所请求的特定服务)，并转回到“请求者空闲”状态。

对于无确认的 MMS 服务，无需接收应答 PDU 和错误 PDU。因此，取消一个无确认的 MMS 服务是不可能的。

图 4 从服务应答者的角度看无确认服务请求

转换：

$$1- \quad \frac{\text{Unconfirmed-PDU(Y)}}{\text{y. indication}}$$

6.3.2.2 服务应答者

图 4 从服务应答者的角度描述一个无确认的 MMS 服务的进行步骤。接收 UnconfirmedPDU 之前，服务被认为处于“应答者空闲”状态。一旦接收到对上述某个无确认服务的请求原语，MMS 提供者发出指示服务原语(根据所接收到的 UnconfirmedPDU 中的信息，指定所请求的特定服务)，并且，从“应答者空闲”状态转换到“应答者空闲”(即转换到同一状态)。

对于无确认的 MMS 服务，MMS 用户可以不发出任何应答原语。因此，取消一个无确认的服务是不可能的。

6.3.3 取消服务

当取消服务是一个确认服务时，对它的操作方式不同于其他确认服务的操作方式。当调用取消服务时，不建立新的状态机。更确切地说，是要被取消的服务的状态机在起作用。取消服务请求不能通过另一调用取消服务来取消。在取消服务请求中指定的调用 ID 可以不是另一个调用取消服务的 ID，因为取消服务仅对构成第 7 章中所定义的 ConfirmedServiceRequest 选择的那些服务进行操作。

在任一给定时刻，对于任一给定的服务请求实例，只有一个取消服务调用是待完成的。调用取消服务不影响在任一给定时刻处于待完成状态的服务请求的个数限制，服务请求的个数是由启动服务商定的。并不对取消服务请求计数，以确定此限制是否达到。

取消服务的操作已在前面描述 MMS 确认服务的服务请求者和服务应答者的章节中描述。

6.4 错误条件的处理

一旦接收到一个无效 PDU，MMS 提供者就向 MMS 用户发出一条拒绝指示原语，指明所检测到的错误。并且，向发送这个无效 PDU 的系统发送一个 RejectPDU。在这一情况下，不发生状态改变。如果确定这个无效 PDU 是一个无效的 RejectPDU，那么，就不发送 RejectPDU。

一旦接收到试图取消一个未知服务请求的 Cancel-requestPDU 时，例如，当指定的调用 ID 不是一个等待完成的确认服务时，MMS 提供者将向取消请求的发送者发送一个 Cancel-ErrorPDU。在这种情况下，这个错误的取消意图就不通告 MMS 用户。

注：对于已请求取消的服务，有可能由两个通信用户同时发出 Cancel-requestPDU 和 Confirmed-ResponsePDU 或 Confirmed-ErrorPDU。这样，一方认为服务已完成，而另一方则认为在等待取消。在这种情况下，取消请求失败，服务正常完成。

一旦接收到 Cancel-ErrorPDU，此时，由被取消服务的调用 ID 所引用的状态机处于“请求空闲”状态，MMS 提供者向 MMS 用户发出取消确认服务原语。

注：当被取消服务的 Confirmed-ResponsePDU 或 Confirmed-ErrorPDU 送出对该服务的 Cancel-RequestPDU 时，发生上述情况。

6.5 拒绝服务和拒绝 PDU

拒绝服务用于将已发生的协议错误通告 MMS 用户。这个服务的操作在 GB/T 16720.1 的第 7 章描述。

注：MMS 用户在接收到拒绝指示服务原语后所进行的活动是本地事务。重要的是必须注意，由于请求、应答或错误有可能被拒绝，其结果是：在会话中的两个 MMS 用户对待要完成事务对象（参见 GB/T 16720.1 第 7 章）的状态可能会有不同的理解。在任意时刻，异常中止服务均可被 MMS 用户用来终止 MMS 环境和应用关联。

7 MMS PDU（MMS 协议数据单元）

本章描述用于操作 MMS 协议的 PDUs。这些 PDUs 向底层服务的映射在第 24 章描述。MMS 服务向这些 PDUs 的映射在第 8 至第 23 章描述。

```
ISO-9506-MMS-1 { iso standard 9506 part(2) mms-abstract-syntax-version1(1) }
DEFINITIONS ::= BEGIN
    EXPORTS AlternateAccess,
    AttachToEventCondition,
    AttachToSemaphore,
    ConfirmedServiceRequest,
    Data,
    EE-State,
    FileName,
    Identifier,
    Integer8,
    Integer32,
    MMSString,
    MMS255String,
    ObjectName,
    TimeOfDay,
    TypeSpecification,
    Unsigned32,
    Unsigned8,
    VariableSpecification;
    IMPORTS ApplicationReference,
    Authentication-value FROM
    MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1 (4) }
    ObtainFile-Request,
```

ObtainFile-Response,
ObtainFile-Error,
FileOpen-Request,
FileOpen-Response,
FileRead-Request,
FileRead-Response,
FileClose-Request,
FileClose-Response,
FileRename-Request,
FileRename-Response,
FileRename-Error,
FileDelete-Request,
FileDelete-Response,
FileDirectory-Request,
FileDirectory-Response,
DefineScatteredAccess-Request,
DefineScatteredAccess-Response,
ScatteredAccessDescription,
GetScatteredAccessAttributes-Request,
GetScatteredAccessAttributes-Response FROM
ISO-9506-MMS-1A { iso standard 9506 part(2) mms-annex-version1(3) }
AccessCondition,
AdditionalCBBOptions,
AdditionalSupportOptions,
Address,
AlarmAckRule,
Control-State,
DomainState,
EC-State,
EC-Class,
EE-Duration,
EE-Class,
EventTime,
Journal-Variable,
LogicalStatus,
Modifier,
normalPriority,
normalSeverity,
ParameterSupportOptions,
PhysicalStatus,
Priority,
ProgramInvocationState,
Running-Mode,

```
    ServiceSupportOptions,
    Severity,
    Transitions,
    TypeDescription,
    ULState,
    VMDState
    FROM MMS-Object-Module-1
    { iso standard 9506 part(1) mms-object-model-version1(2) };
MMSpdu ::= CHOICE {
    confirmed-RequestPDU        [0] IMPLICIT Confirmed-RequestPDU,
    confirmed-ResponsePDU       [1] IMPLICIT Confirmed-ResponsePDU,
    confirmed-ErrorPDU          [2] IMPLICIT Confirmed-ErrorPDU,
IF ( unsolicitedStatus informationReport eventNotification )
    unconfirmed-PDU             [3] IMPLICIT Unconfirmed-PDU,
ELSE
    unconfirmed-PDU             [3] IMPLICIT NULL,
ENDIF
    rejectPDU                   [4] IMPLICIT RejectPDU,
IF (cancel)
    cancel-RequestPDU           [5] IMPLICIT Cancel-RequestPDU,
    cancel-ResponsePDU          [6] IMPLICIT Cancel-ResponsePDU,
    cancel-ErrorPDU             [7] IMPLICIT Cancel-ErrorPDU,
ELSE
    cancel-RequestPDU           [5] IMPLICIT NULL,
    cancel-ResponsePDU          [6] IMPLICIT NULL,
    cancel-ErrorPDU             [7] IMPLICIT NULL,
ENDIF
    initiate-RequestPDU         [8] IMPLICIT Initiate-RequestPDU,
    initiate-ResponsePDU        [9] IMPLICIT Initiate-ResponsePDU,
    initiate-ErrorPDU           [10] IMPLICIT Initiate-ErrorPDU,
    conclude-RequestPDU         [11] IMPLICIT Conclude-RequestPDU,
    conclude-ResponsePDU        [12] IMPLICIT Conclude-ResponsePDU,
    conclude-ErrorPDU           [13] IMPLICIT Conclude-ErrorPDU
    }
```

在 MMS 中有 14 类 PDUs，Initiate-RequestPDU、Initiate-ResponsePDU、Initiate-ErrorPDU Conclude-RequestPDU、Conclude-ResponsePDU、Conclude-ErrorPDU、RejectPDU、Cancel-RequestPDU、Cancel-ResponsePDU 和 Cancel-ErrorPDU 在第 8 章定义，其余的 PDU 类型在 7.1 至 7.4 中定义。

7.1 Confirmed-Request PDU(确认—请求 PDU)

```
  Confirmed-RequestPDU ::= SEQUENCE {
    invokeID                        Unsigned32,
IF (attachToEventCondition attachToSemaphore )
    listOfModifiers                 SEQUENCE OF Modifier OPTIONAL,
```

```
ENDIF
    service                         ConfirmedServiceRequest,
...
IF ( csr cspi ),
    service-ext                     [79] Request-Detail OPTIONAL
ENDIF
                        —shall not be transmitted if value is the value
                        —of a tagged type derived from NULL
}
```

Confirmed-RequestPDU 是包含 4 个元素的序列，这 4 个元素是：一个无符号整数、一个可选的修饰符表、一个 ConfirmedServiceRequest 和一个 Request-detail。

invokeID 是一个 32 位的无符号整数，它唯一地标识来自给定应用关联上一个特定 MMS 用户的所有待完成的确认服务请求中的一个服务请求。在任一时刻，对于任意给定的 invokeID，至多只有一个来自某个应用关联上的特定 MMS 用户的待完成的服务请求与之对应。invokeID 的值应是 MMS 用户在请求服务原语（参见 GB/T 16720.1 第 5 章）中提供的值。在 Confirmed-ResponsePDU 和 Confirmed-ErrorPDU 中提供的 invokeID 使得 MMS 提供者和 MMS 用户将这些 PDUs 与相应的服务请求联系起来。

ListOfModifiers 用于指定执行服务请求的修饰符。在 ListOfModifiers 中给出的一个修饰符应该在该 ListOfModifiers 中下一个修饰符执行之前或者在服务请求执行前被成功地执行。因此，修饰符在表中的顺序是很重要的。如果没有给出 ListOfModifiers，那么，一旦接收该请求，则可以立即无先决条件地开始执行服务请求。

ConfirmedServiceRequest 用于指定一个确认服务及其自变量。该参数在 7.1.1 中进一步说明。

修饰符的作用由第 6 章中提供的服务调用状态机设定。修饰符的定义可以在本部分随后章节的协议描述中找到。

7.1.1 ConfirmedServiceRequest（确认服务请求）

```
    ConfirmedServiceRequest ::= CHOICE {
IF ( status )
    status
      [0] IMPLICIT Status-Request
ELSE
    status
      [0] IMPLICIT NULL
ENDIF
IF ( getNameList )
,   getNameList
      [1] IMPLICIT GetNameList-Request
ELSE
,   getNameList
      [1] IMPLICIT NULL
ENDIF
IF ( identify )
,   identify
```

```
        [2] IMPLICIT Identify-Request
ELSE
,   identify
        [2] IMPLICIT NULL
ENDIF
IF ( rename )
,   rename
        [3] IMPLICIT Rename-Request
ELSE
,   rename
        [3] IMPLICIT NULL
ENDIF
IF ( read )
,   read
        [4] IMPLICIT Read-Request
ELSE
,   read
        [4] IMPLICIT NULL
ENDIF
IF ( write )
,   write
        [5] IMPLICIT Write-Request
ELSE
,   write
        [5] IMPLICIT NULL
ENDIF
IF ( vnam vadr )
IF ( getVariableAccessAttributes )
,   getVariableAccessAttributes
        [6] GetVariableAccessAttributes-Request
ELSE
,   getVariableAccessAttributes
        [6] IMPLICIT NULL
ENDIF
ELSE
,   getVariableAccessAttributes
        [6] IMPLICIT NULL
ENDIF
IF ( vnam )
IF ( vadr )
IF ( defineNamedVariable )
,   defineNamedVariable
        [7] IMPLICIT DefineNamedVariable-Request
```

```
ELSE
,   defineNamedVariable
        [7] IMPLICIT NULL
ENDIF
ELSE
,   defineNamedVariable
        [7] IMPLICIT NULL
ENDIF
ELSE
,   defineNamedVariable
        [7] IMPLICIT NULL
ENDIF
IF ( vsca )
  —[8] is reserved for a service defined in Annex E
IF ( defineScatteredAccess )
,   defineScatteredAccess
        [8] IMPLICIT DefineScatteredAccess-Request
ELSE
,   defineScatteredAccess
        [8] IMPLICIT NULL
ENDIF
  —[9] is reserved for a service defined in Annex E
IF ( getScatteredAccessAttributes )
,   getScatteredAccessAttributes
        [9] GetScatteredAccessAttributes-Request
ELSE
,   getScatteredAccessAttributes
        [9] IMPLICIT NULL
ENDIF
ELSE
,   defineScatteredAccess
        [8] IMPLICIT NULL,
    getScatteredAccessAttributes
        [9] IMPLICIT NULL
ENDIF
IF ( vnam )
IF ( deleteVariableAccess )
,   deleteVariableAccess
        [10] IMPLICIT DeleteVariableAccess-Request
ELSE
,   deleteVariableAccess
        [10] IMPLICIT NULL
ENDIF
```

```
ELSE
,   deleteVariableAccess
        [10] IMPLICIT NULL
ENDIF
IF ( vlis )
IF ( vnam )
IF ( defineNamedVariableList )
,   defineNamedVariableList
        [11] IMPLICIT DefineNamedVariableList-Request
ELSE
,   defineNamedVariableList
        [11] IMPLICIT NULL
ENDIF
IF ( getNamedVariableListAttributes )
,   getNamedVariableListAttributes
        [12] GetNamedVariableListAttributes-Request
ELSE
,   getNamedVariableListAttributes
        [12] IMPLICIT NULL
ENDIF
IF ( deleteNamedVariableList )
,   deleteNamedVariableList
        [13] IMPLICIT DeleteNamedVariableList-Request
ELSE
,   deleteNamedVariableList
        [13] IMPLICIT NULL
ENDIF
ELSE
,   defineNamedVariableList
        [11] IMPLICIT NULL,
    getNamedVariableListAttribute
        [12] IMPLICIT NULL,
    deleteNamedVariableList
        [13] IMPLICIT NULL
ENDIF
ELSE
,   defineNamedVariableList
        [11] IMPLICIT NULL,
    getNamedVariableListAttributes
        [12] IMPLICIT NULL,
    deleteNamedVariableList
        [13] IMPLICIT NULL
ENDIF
```

```
ENDIF
IF ( vnam )
IF ( defineNamedType )
,  defineNamedType
       [14] IMPLICIT DefineNamedType-Request
ELSE
,  defineNamedType
       [14] IMPLICIT NULL
ENDIF
IF ( getNamedTypeAttributes )
,  getNamedTypeAttributes
       [15] GetNamedTypeAttributes-Request
ELSE
,  getNamedTypeAttributes
       [15] IMPLICIT NULL
ENDIF
IF ( deleteNamedType )
,  deleteNamedType
       [16] IMPLICIT DeleteNamedType-Request
ELSE
,  deleteNamedType
       [16] IMPLICIT NULL
ENDIF
ELSE
,  defineNamedType
       [14] IMPLICIT NULL,
    getNamedTypeAttributes
       [15] IMPLICIT NULL,
    deleteNamedType
       [16] IMPLICIT NULL
ENDIF
IF ( input )
,  input
       [17] IMPLICIT Input-Request
ELSE
,  input
       [17] IMPLICIT NULL
ENDIF
IF ( output )
,  output
       [18] IMPLICIT Output-Request
ELSE
,  output
```

```
        [18] IMPLICIT NULL
ENDIF
IF ( takeControl )
,   takeControl
        [19] IMPLICIT TakeControl-Request
ELSE
,   takeControl
        [19] IMPLICIT NULL
ENDIF
IF ( relinquishControl )
,   relinquishControl
        [20] IMPLICIT RelinquishControl-Request
ELSE
,   relinquishControl
        [20] IMPLICIT NULL
ENDIF
IF ( defineSemaphore )
,   defineSemaphore
        [21] IMPLICIT DefineSemaphore-Request
ELSE
,   defineSemaphore
        [21] IMPLICIT NULL
ENDIF
IF ( deleteSemaphore )
,   deleteSemaphore
        [22] DeleteSemaphore-Request
ELSE
,   deleteSemaphore
        [22] IMPLICIT NULL
ENDIF
IF ( reportSemaphoreStatus )
,   reportSemaphoreStatus
        [23] ReportSemaphoreStatus-Request
ELSE
,   reportSemaphoreStatus
        [23] IMPLICIT NULL
ENDIF
IF ( reportPoolSemaphoreStatus )
,   reportPoolSemaphoreStatus
        [24] IMPLICIT ReportPoolSemaphoreStatus-Request
ELSE
,   reportPoolSemaphoreStatus
        [24] IMPLICIT NULL
```

```
ENDIF
IF ( reportSemaphoreEntryStatus )
,  reportSemaphoreEntryStatus
       [25] IMPLICIT ReportSemaphoreEntryStatus-Request
ELSE
,  reportSemaphoreEntryStatus
       [25] IMPLICIT NULL
ENDIF
IF ( initiateDownloadSequence )
,  initiateDownloadSequence
       [26] IMPLICIT InitiateDownloadSequence-Request,
    downloadSegment
       [27] IMPLICIT DownloadSegment-Request,
    terminateDownloadSequence
       [28] IMPLICIT TerminateDownloadSequence-Request
ELSE
,  initiateDownloadSequence
       [26] IMPLICIT NULL,
    downloadSegment
       [27] IMPLICIT NULL,
    terminateDownloadSequence
       [28] IMPLICIT NULL
ENDIF
IF ( initiateUploadSequence )
,  initiateUploadSequence
       [29] IMPLICIT InitiateUploadSequence-Request,
    uploadSegment
       [30] IMPLICIT UploadSegment-Request,
    terminateUploadSequence
       [31] IMPLICIT TerminateUploadSequence-Request
ELSE
,  initiateUploadSequence
       [29] IMPLICIT NULL,
    uploadSegment
       [30] IMPLICIT NULL,
    terminateUploadSequence
       [31] IMPLICIT NULL
ENDIF
IF ( requestDomainDownload )
,  requestDomainDownload
       [32] IMPLICIT RequestDomainDownload-Request
ELSE
,  requestDomainDownload
```

```
        [32] IMPLICIT NULL
ENDIF
IF ( requestDomainUpload )
,   requestDomainUpload
        [33] IMPLICIT RequestDomainUpload-Request
ELSE
,   requestDomainUpload
        [33] IMPLICIT NULL
ENDIF
IF ( loadDomainContent )
,   loadDomainContent
        [34] IMPLICIT LoadDomainContent-Request
ELSE
,   loadDomainContent
        [34] IMPLICIT NULL
ENDIF
IF ( storeDomainContent )
,   storeDomainContent
        [35] IMPLICIT StoreDomainContent-Request
ELSE
,    storeDomainContent
        [35] IMPLICIT NULL
ENDIF
IF ( deleteDomain )
,   deleteDomain
        [36] IMPLICIT DeleteDomain-Request
ELSE
,   deleteDomain
        [36] IMPLICIT NULL
ENDIF
IF ( getDomainAttributes )
,   getDomainAttributes
        [37] IMPLICIT GetDomainAttributes-Request
ELSE
,   getDomainAttributes
        [37] IMPLICIT NULL
ENDIF
IF ( createProgramInvocation )
,   createProgramInvocation
        [38] IMPLICIT CreateProgramInvocation-Request
ELSE
,   createProgramInvocation
        [38] IMPLICIT NULL
```

```
ENDIF
IF ( deleteProgramInvocation )
, deleteProgramInvocation
      [39] IMPLICIT DeleteProgramInvocation-Request
ELSE
, deleteProgramInvocation
      [39] IMPLICIT NULL
ENDIF
IF ( start )
, start
      [40] IMPLICIT Start-Request
ELSE
, start
      [40] IMPLICIT NULL
ENDIF
IF ( stop )
, stop
      [41] IMPLICIT Stop-Request
ELSE
, stop
      [41] IMPLICIT NULL
ENDIF
IF ( resume )
, resume
      [42] IMPLICIT Resume-Request
ELSE
, resume
      [42] IMPLICIT NULL
ENDIF
IF ( reset )
, reset
      [43] IMPLICIT Reset-Request
ELSE
, reset
      [43] IMPLICIT NULL
ENDIF
IF ( kill )
, kill
      [44] IMPLICIT Kill-Request
ELSE
, kill
      [44] IMPLICIT NULL
ENDIF
```

```
IF ( getProgramInvocationAttributes )
, getProgramInvocationAttributes
      [45] IMPLICIT GetProgramInvocationAttributes-Request
ELSE
, getProgramInvocationAttributes
      [45] IMPLICIT NULL
ENDIF
IF ( obtainFile )
, obtainFile
      [46] IMPLICIT ObtainFile-Request
ELSE
, obtainFile
      [46] IMPLICIT NULL
ENDIF
IF ( defineEventCondition )
, defineEventCondition
      [47] IMPLICIT DefineEventCondition-Request
ELSE
, defineEventCondition
      [47] IMPLICIT NULL
ENDIF
IF ( deleteEventCondition )
, deleteEventCondition
      [48] DeleteEventCondition-Request
ELSE
, deleteEventCondition
      [48] IMPLICIT NULL
ENDIF
IF ( getEventConditionAttributes )
, getEventConditionAttributes
      [49] GetEventConditionAttributes-Request
ELSE
, getEventConditionAttributes
      [49] IMPLICIT NULL
ENDIF
IF ( reportEventConditionStatus )
, reportEventConditionStatus
      [50] ReportEventConditionStatus-Request
ELSE
, reportEventConditionStatus
      [50] IMPLICIT NULL
ENDIF
IF ( alterEventConditionMonitoring )
```

```
, alterEventConditionMonitoring
      [51] IMPLICIT AlterEventConditionMonitoring-Request
ELSE
, alterEventConditionMonitoring
      [51] IMPLICIT NULL
ENDIF
IF ( triggerEvent )
, triggerEvent
      [52] IMPLICIT TriggerEvent-Request
ELSE
, triggerEvent
      [52] IMPLICIT NULL
ENDIF
IF ( defineEventAction )
, defineEventAction
      [53] IMPLICIT DefineEventAction-Request
ELSE
, defineEventAction
      [53] IMPLICIT NULL
ENDIF
IF ( deleteEventAction )
, deleteEventAction
      [54] DeleteEventAction-Request
ELSE
, deleteEventAction
      [54] IMPLICIT NULL
ENDIF
IF ( getEventActionAttributes )
, getEventActionAttributes
      [55] GetEventActionAttributes-Request
ELSE
, getEventActionAttributes
      [55] IMPLICIT NULL
ENDIF
IF ( reportEventActionStatus )
, reportEventActionStatus
      [56] ReportEventActionStatus-Request
ELSE
, reportEventActionStatus
      [56] IMPLICIT NULL
ENDIF
IF ( defineEventEnrollment )
, defineEventEnrollment
```

```
        [57] IMPLICIT DefineEventEnrollment-Request
ELSE
,   defineEventEnrollment
        [57] IMPLICIT NULL
ENDIF
IF ( deleteEventEnrollment )
,   deleteEventEnrollment
        [58] DeleteEventEnrollment-Request
ELSE
,   deleteEventEnrollment
        [58] IMPLICIT NULL
ENDIF
IF ( alterEventEnrollment )
,   alterEventEnrollment
        [59] IMPLICIT AlterEventEnrollment-Request
ELSE
,   alterEventEnrollment
        [59] IMPLICIT NULL
ENDIF
IF ( reportEventEnrollmentStatus )
,   reportEventEnrollmentStatus
        [60] ReportEventEnrollmentStatus-Request
ELSE
,   reportEventEnrollmentStatus
        [60] IMPLICIT NULL
ENDIF
IF ( getEventEnrollmentAttributes )
,   getEventEnrollmentAttributes
        [61] IMPLICIT GetEventEnrollmentAttributes-Request
ELSE
,    getEventEnrollmentAttributes
        [61] IMPLICIT NULL
ENDIF
IF ( acknowledgeEventNotification )
,   acknowledgeEventNotification
        [62] IMPLICIT AcknowledgeEventNotification-Request
ELSE
,   acknowledgeEventNotification
        [62] IMPLICIT NULL
ENDIF
IF ( getAlarmSummary )
,   getAlarmSummary
        [63] IMPLICIT GetAlarmSummary-Request
```

```
ELSE
,   getAlarmSummary
        [63] IMPLICIT NULL
ENDIF
IF ( getAlarmEnrollmentSummary )
,   getAlarmEnrollmentSummary
        [64] IMPLICIT GetAlarmEnrollmentSummary-Request
ELSE
,   getAlarmEnrollmentSummary
        [64] IMPLICIT NULL
ENDIF
IF ( readJournal )
,   readJournal
        [65] IMPLICIT ReadJournal-Request
ELSE
,   readJournal
        [65] IMPLICIT NULL
ENDIF
IF ( writeJournal )
,   writeJournal
        [66] IMPLICIT WriteJournal-Request
ELSE
,   writeJournal
        [66] IMPLICIT NULL
ENDIF
IF ( initializeJournal )
,   initializeJournal
        [67] IMPLICIT InitializeJournal-Request
ELSE
,   initializeJournal
        [67] IMPLICIT NULL
ENDIF
IF ( reportJournalStatus )
,   reportJournalStatus
        [68] ReportJournalStatus-Request
ELSE
,   reportJournalStatus
        [68] IMPLICIT NULL
ENDIF
IF ( createJournal )
,   createJournal
        [69] IMPLICIT CreateJournal-Request
ELSE
```

```
, createJournal
        [69] IMPLICIT NULL
ENDIF
IF ( deleteJournal )
, deleteJournal
        [70] IMPLICIT DeleteJournal-Request
ELSE
, deleteJournal
        [70] IMPLICIT NULL
ENDIF
IF ( getCapabilityList )
, getCapabilityList
        [71] IMPLICIT GetCapabilityList-Request
ELSE
, getCapabilityList
        [71] IMPLICIT NULL
ENDIF
        —choices [72] through [77] are reserved for use by services
        —defined in annex D
IF ( fileOpen )
, fileOpen
        [72] IMPLICIT FileOpen-Request
ELSE
,  fileOpen
        [72] IMPLICIT NULL
ENDIF
IF ( fileRead )
, fileRead
        [73] IMPLICIT FileRead-Request
ELSE
,  fileRead
        [73] IMPLICIT NULL
ENDIF
IF ( fileClose )
,  fileClose
        [74] IMPLICIT FileClose-Request
ELSE
, fileClose
        [74] IMPLICIT NULL
ENDIF
IF ( fileRename )
, fileRename
        [75] IMPLICIT FileRename-Request
```

```
ELSE
,   fileRename
        [75] IMPLICIT NULL
ENDIF
IF ( fileDelete )
,    fileDelete
        [76] IMPLICIT FileDelete-Request
ELSE
,   fileDelete
        [76] IMPLICIT NULL
ENDIF
IF ( fileDirectory )
,   fileDirectory
        [77] IMPLICIT FileDirectory-Request
ELSE
,    fileDirectory
        [77] IMPLICIT NULL
ENDIF
, ...
IF ( csr cspi )
,   additionalService
        [78] AdditionalService-Request
ENDIF
        —choice [79] is reserved
IF ( getDataExchangeAttributes )
,   getDataExchangeAttributes
        [80] GetDataExchangeAttributes-Request
                —Shall not appear in minor version 1
ENDIF
IF ( exchangeData )
,   exchangeData
        [81] IMPLICIT ExchangeData-Request
                —Shall not appear in minor version 1
ENDIF
IF ( defineAccessControlList )
,   defineAccessControlList
        [82] IMPLICIT DefineAccessControlList-Request
                —Shall not appear in minor version 1 or 2
ENDIF
IF ( getAccessControlListAttributes )
,   getAccessControlListAttributes
        [83] GetAccessControlListAttributes-Request
                —Shall not appear in minor version 1 or 2
```

```
ENDIF
IF ( reportAccessControlledObjects )
,   reportAccessControlledObjects
        [84] IMPLICIT ReportAccessControlledObjects-Request
                —Shall not appear in minor version 1 or 2
ENDIF
IF ( deleteAccessControlList )
,    deleteAccessControlList
        [85] IMPLICIT DeleteAccessControlList-Request
                —Shall not appear in minor version 1 or 2
ENDIF
IF ( changeAccessControl )
,   changeAccessControl
        [86] IMPLICIT ChangeAccessControl-Request
                —Shall not appear in minor version 1 or 2
ENDIF
        , ...
        }
```

ConfirmedServiceRequest 类型应指明服务的类型以及该服务的自变量。所提供的上下文标记指明这个服务类型。通过 ConfirmedServiceRequest 产生式所引用的类型定义，每个专用服务的定义规定了服务的自变量的格式。在 ConfirmedServiceRequest 选择中的每个服务是确认服务。

7.1.2 AdditionalService-Request(附加服务—请求)

```
  AdditionalService-Request ::= CHOICE {
IF ( csr )
IF ( vMDStop )
    vMDStop
      [0] IMPLICIT VMDStop-Request
ELSE
    vMDStop
      [0] IMPLICIT NULL
ENDIF
IF ( vMDReset )
,   vMDReset
      [1] IMPLICIT VMDReset-Request
ELSE
,   vMDReset
      [1] IMPLICIT NULL
ENDIF
IF ( select )
,   select
      [2] IMPLICIT Select-Request
ELSE
,   select
```

```
        [2] IMPLICIT NULL
ENDIF
IF ( alterProgramInvocationAttributes )
,   alterPI
        [3] IMPLICIT AlterProgramInvocationAttributes-Request
ELSE
,   alterPI
        [3] IMPLICIT NULL
ENDIF
ELSE
,   vMDStop
        [0] IMPLICIT NULL,
    vMDReset
        [1] IMPLICIT NULL,
    select
        [2] IMPLICIT NULL,
    alterPI
        [3] IMPLICIT NULL
ENDIF
IF ( cspi )
IF ( initiateUnitControlLoad )
,   initiateUCLoad
        [4] IMPLICIT InitiateUnitControlLoad-Request
ELSE
,   initiateUCLoad
        [4] IMPLICIT NULL
ENDIF
IF ( unitControlLoadSegment )
,   uCLoad
        [5] IMPLICIT UnitControlLoadSegment-Request
ELSE
,   uCLoad
        [5] IMPLICIT NULL
ENDIF
IF ( unitControlUpload )
,   uCUpload
        [6] IMPLICIT UnitControlUpload-Request
ELSE
,   uCUpload
        [6] IMPLICIT NULL
ENDIF
IF ( startUnitControl )
,   startUC
```

```
        [7] IMPLICIT StartUnitControl-Request
ELSE
,   startUC
        [7] IMPLICIT NULL
ENDIF
IF ( stopUnitControl )
,    stopUC
        [8] IMPLICIT StopUnitControl-Request
ELSE
,    stopUC
        [8] IMPLICIT NULL
ENDIF
IF ( createUnitControl )
,   createUC
        [9] IMPLICIT CreateUnitControl-Request
ELSE
,   createUC
        [9] IMPLICIT NULL
ENDIF
IF ( addToUnitControl )
,   addToUC
        [10] IMPLICIT AddToUnitControl-Request
ELSE
,   addToUC
        [10] IMPLICIT NULL
ENDIF
IF ( removeFromUnitControl )
,   removeFromUC
        [11] IMPLICIT RemoveFromUnitControl-Request
ELSE
,    removeFromUC
        [11] IMPLICIT NULL
ENDIF
IF ( getUnitControlAttributes )
,   getUCAttributes
        [12] IMPLICIT GetUnitControlAttributes-Request
ELSE
,   getUCAttributes
        [12] IMPLICIT NULL
ENDIF
IF ( loadUnitControlFromFile )
,    loadUCFromFile
        [13] IMPLICIT LoadUnitControlFromFile-Request
```

```
ELSE
,   loadUCFromFile
        [13] IMPLICIT NULL
ENDIF
IF ( storeUnitControlToFile )
,   storeUCToFile
        [14] IMPLICIT StoreUnitControlToFile-Request
ELSE
,   storeUCToFile
        [14] IMPLICIT NULL
ENDIF
IF ( deleteUnitControl )
,   deleteUC
        [15] IMPLICIT DeleteUnitControl-Request
ELSE
,   deleteUC
        [15] IMPLICIT NULL
ENDIF
IF ( defineEventConditionList )
,   defineECL
        [16] DefineEventConditionList-Request
ELSE
,   defineECL
        [16] IMPLICIT NULL
ENDIF
IF ( deleteEventConditionList )
,   deleteECL
        [17] DeleteEventConditionList-Request
ELSE
,   deleteECL
        [17] IMPLICIT NULL
ENDIF
IF ( addEventConditionListReference )
,   addECLReference
        [18] IMPLICIT AddEventConditionListReference-Request
ELSE
,   addECLReference
        [18] IMPLICIT NULL
ENDIF

IF ( removeEventConditionListReference )
,   removeECLReference
        [19] IMPLICIT RemoveEventConditionListReference-Request
```

```
ELSE
,  removeECLReference
       [19] IMPLICIT NULL
ENDIF
IF ( getEventConditionListAttributes )
,  getECLAttributes
       [20] GetEventConditionListAttributes-Request
ELSE
,  getECLAttributes
       [20] IMPLICIT NULL
ENDIF
IF ( reportEventConditionListStatus )
,   reportECLStatus
       [21] IMPLICIT ReportEventConditionListStatus-Request
ELSE
,  reportECLStatus
       [21] IMPLICIT NULL
ENDIF
IF ( alterEventConditionListMonitoring )
,  alterECLMonitoring
       [22] IMPLICIT AlterEventConditionListMonitoring-Request
ELSE
,  alterECLMonitoring
       [22] IMPLICIT NULL
ENDIF
ELSE
,  initiateUCLoad
       [4] IMPLICIT NULL,
    uCLoad
       [5] IMPLICIT NULL,
    uCUpload
       [6] IMPLICIT NULL,
    startUC
       [7] IMPLICIT NULL,
    stopUC
       [8] IMPLICIT NULL,
    createUC
       [9] IMPLICIT NULL,
    addToUC
       [10] IMPLICIT NULL,
    removeFromUC
       [11] IMPLICIT NULL,
    getUCAttributes
```

```
      [12] IMPLICIT NULL,
    loadUCFromFile
      [13] IMPLICIT NULL,
    storeUCToFile
      [14] IMPLICIT NULL,
    deleteUC
      [15] IMPLICIT NULL,
    defineECL
      [16] IMPLICIT NULL,
    deleteECL
      [17] IMPLICIT NULL,
    addECLReference
      [18] IMPLICIT NULL,
    removeECLReference
      [19] IMPLICIT NULL,
    getECLAttributes
      [20] IMPLICIT NULL,
    reportECLStatus
      [21] IMPLICIT NULL,
    alterECLMonitoring
      [22] IMPLICIT NULL
ENDIF
      }
```

7.1.3 Request-Detail(请求—细目)

```
  Request-Detail ::= CHOICE {
          —this choice shall be selected if the tag value of the
          —ConfirmedServiceRequest does not match any of the tags below
    otherRequests       NULL
IF ( createProgramInvocation )
,   createProgramInvocation
      [38] IMPLICIT CS-CreateProgramInvocation-Request
ELSE
,   createProgramInvocation
      [38] IMPLICIT NULL
ENDIF
IF ( start )
,   start
      [40] IMPLICIT CS-Start-Request
ELSE
,   start
      [40] IMPLICIT NULL
ENDIF
IF ( resume )
```

```
, resume
      [42] IMPLICIT CS-Resume-Request
ELSE
, resume
      [42] IMPLICIT NULL
ENDIF
IF ( defineEventCondition )
, defineEventCondition
      [47] IMPLICIT CS-DefineEventCondition-Request
ELSE
, defineEventCondition
      [47] IMPLICIT NULL
ENDIF
IF ( alterEventConditionMonitoring )
,  alterEventConditionMonitoring
      [51] IMPLICIT CS-AlterEventConditionMonitoring-Request
ELSE
, alterEventConditionMonitoring
      [51] IMPLICIT NULL
ENDIF
IF ( defineEventEnrollment )
, defineEventEnrollment
      [57] IMPLICIT CS-DefineEventEnrollment-Request
ELSE
, defineEventEnrollment
      [57] IMPLICIT NULL
ENDIF
IF ( alterEventEnrollment )
, alterEventEnrollment
      [59] IMPLICIT CS-AlterEventEnrollment-Request
ELSE
, alterEventEnrollment
      [59] IMPLICIT NULL
ENDIF
      }
```

7.2 Unconfirmed PDU(无确认 PDU)

```
  Unconfirmed-PDU ::= SEQUENCE {
    service              UnconfirmedService,
    ...
IF ( cspi )
, service-ext            [79] Unconfirmed-Detail OPTIONAL
ENDIF
                              --shall not be transmitted if value is the value
```

```
                    —of a tagged type derived from NULL
    }
```

UnconfirmedPDU 应是一个包含 UnconfirmedSercice 和 Unconfirmed-Detail 的序列。

UnconfirmedService 用于指明一个无确认的服务及其自变量。

7.2.1 UnconfirmedService(无确认服务)

```
  UnconfirmedService ::= CHOICE {
IF (informationReport )
    informationReport
      [0] IMPLICIT InformationReport
ELSE
    informationReport
      [0] IMPLICIT NULL
ENDIF
IF ( unsolicitedStatus )
,   unsolicitedStatus
      [1] IMPLICIT UnsolicitedStatus
ELSE
,   unsolicitedStatus
      [1] IMPLICIT NULL
ENDIF
IF ( eventNotification )
,   eventNotification
      [2] IMPLICIT EventNotification
ELSE
,   eventNotification
      [2] IMPLICIT NULL
ENDIF
    }
```

UnconfirmedService 类型指明服务的类型及该服务的自变量。提供的上下文标记指定这个服务类型。通过 UnconfirmedServiceRequest 产生式所引用的类型定义,每个服务的定义规定了服务的自变量格式。UnconfirmedService 选择中的每个服务都是无确认的服务。

7.2.2 Unconfirmed-Detail(无确认—细目)

```
  Unconfirmed-Detail ::= CHOICE {
            —this choice shall be selected if the tag value of the
            —UnconfirmedService does not match any of the tags below
    otherRequests NULL
IF ( cspi )
,   eventNotification
      [2] IMPLICIT CS-EventNotification
ENDIF
    }
```

7.3 Confirmed-Response PDU(确认—应答 PDU)

```
  Confirmed-ResponsePDU ::= SEQUENCE {
```

```
    invokeID                Unsigned32,
    service                 ConfirmedServiceResponse,
    ...
IF ( csr cspi ),
    service-ext             [79] Response-Detail OPTIONAL
ENDIF
                            —shall not be transmitted if value is the value
                            —of a tagged type derived from NULL
    }
```

Confirmed-ResponsePDU 是包含三个元素的序列。这三个元素是:一个无符号整数、一个 ConfirmedServiceResponse 和一个 Response-Detail。

invoke ID 是一个 32 位的无符号整数,它唯一地标识来自一个应用关联上的一个特定 MMS 用户的所有待完成的确认服务请求中的一个服务请求。在任一时刻,对于任意给定的 invoke ID,至多只可能是一个来自一个应用关联上的特定 MMS 用户的待完成的服务请求。invoke ID 的值应该是 MMS 用户在应答服务原语(参见 GB/T 16720.1 第 5 章)中提供的值。并且,它指定了引起该服务执行的请求实例。在这个 PDU 中的 invokeID 使得 MMS 提供者和 MMS 用户将这个 PDU 与相应的服务请求联系起来。

ConfirmedServiceResponse 用于指定一个确认服务和对这个确认服务的应答。该参数在 7.3.1 中描述。

7.3.1 ConfirmedServiceResponse(确认服务应答)

```
  ConfirmedServiceResponse ::= CHOICE {
IF ( status )
    Status
      [0] IMPLICIT Status-Response
ELSE
    status
      [0] IMPLICIT RejectPDU
ENDIF
IF ( getNameList )
,   getNameList
      [1] IMPLICIT GetNameList-Response
ELSE
,   getNameList
      [1] IMPLICIT RejectPDU
ENDIF
IF ( identify )
,   identify
      [2] IMPLICIT Identify-Response
ELSE
,   identify
      [2] IMPLICIT RejectPDU
ENDIF
```

```
IF ( rename )
,    rename
        [3] IMPLICIT Rename-Response
ELSE
,    rename
        [3] IMPLICIT RejectPDU
ENDIF
IF ( read )
,    read
        [4] IMPLICIT Read-Response
ELSE
,    read
        [4] IMPLICIT RejectPDU
ENDIF
IF ( write )
,    write
        [5] IMPLICIT Write-Response
ELSE
,    write
        [5] IMPLICIT RejectPDU
ENDIF
IF ( vnam vadr )
IF ( getVariableAccessAttributes )
,    getVariableAccessAttributes
        [6] IMPLICIT GetVariableAccessAttributes-Response
ELSE
,    getVariableAccessAttributes
        [6] IMPLICIT RejectPDU
ENDIF
ELSE
,    getVariableAccessAttributes
        [6] IMPLICIT RejectPDU
ENDIF
IF ( vnam )
IF ( vadr )
IF ( defineNamedVariable )
,    defineNamedVariable
        [7] IMPLICIT DefineNamedVariable-Response
ELSE
,    defineNamedVariable
        [7] IMPLICIT RejectPDU
ENDIF
ELSE
```

```
,    defineNamedVariable
       [7] IMPLICIT RejectPDU
ENDIF
ELSE
,    defineNamedVariable
       [7] IMPLICIT RejectPDU
ENDIF
IF ( vsca )
       —choice [8] is reserved for a service defined in Annex E
IF ( defineScatteredAccess )
,    defineScatteredAccess
       [8] IMPLICIT DefineScatteredAccess-Response
ELSE
,    defineScatteredAccess
       [8] IMPLICIT RejectPDU
ENDIF
       —choice [9] is reserved for a service defined in Annex E
IF ( getScatteredAccessAttributes )
,    getScatteredAccessAttributes
       [9] IMPLICIT GetScatteredAccessAttributes-Response
ELSE
,    getScatteredAccessAttributes
       [9] IMPLICIT RejectPDU
ENDIF
ELSE
,    defineScatteredAccess
       [8] IMPLICIT RejectPDU,
     getScatteredAccessAttributes
       [9] IMPLICIT RejectPDU

ENDIF
IF ( vnam )
IF ( deleteVariableAccess )
,    deleteVariableAccess
       [10] IMPLICIT DeleteVariableAccess-Response
ELSE
,    deleteVariableAccess
       [10] IMPLICIT RejectPDU
ENDIF
ELSE
,    deleteVariableAccess
       [10] IMPLICIT RejectPDU
ENDIF
```

```
IF ( vlis )
IF ( vnam )
IF ( defineNamedVariableList )
,  defineNamedVariableList
       [11] IMPLICIT DefineNamedVariableList-Response
ELSE
,    defineNamedVariableList
       [11] IMPLICIT RejectPDU
ENDIF
IF ( getNamedVariableListAttributes )
,    getNamedVariableListAttributes
       [12] IMPLICIT GetNamedVariableListAttributes-Response
ELSE
,  getNamedVariableListAttributes
       [12] IMPLICIT RejectPDU
ENDIF
IF ( deleteNamedVariableList )
,  deleteNamedVariableList
       [13] IMPLICIT DeleteNamedVariableList-Response
ELSE
,  deleteNamedVariableList
       [13] IMPLICIT RejectPDU
ENDIF
ELSE
,  defineNamedVariableList
       [11] IMPLICIT RejectPDU,
    getNamedVariableListAttributes
       [12] IMPLICIT RejectPDU,
    deleteNamedVariableList
       [13] IMPLICIT RejectPDU
ENDIF
ELSE
,  defineNamedVariableList
       [11] IMPLICIT RejectPDU,
    getNamedVariableListAttributes
       [12] IMPLICIT RejectPDU,
    deleteNamedVariableList
       [13] IMPLICIT RejectPDU
ENDIF
IF ( vnam )
IF ( defineNamedType )
,  defineNamedType
       [14] IMPLICIT DefineNamedType-Response
```

```
ELSE
,  defineNamedType
       [14] IMPLICIT RejectPDU
ENDIF
IF ( getNamedTypeAttributes )
,  getNamedTypeAttributes
       [15] IMPLICIT GetNamedTypeAttributes-Response
ELSE
,  getNamedTypeAttributes
       [15] IMPLICIT RejectPDU
ENDIF
IF ( deleteNamedType )
,    deleteNamedType
       [16] IMPLICIT DeleteNamedType-Response
ELSE
,    deleteNamedType
       [16] IMPLICIT RejectPDU
ENDIF
ELSE
,  defineNamedType
       [14] IMPLICIT RejectPDU,
     getNamedTypeAttributes
       [15] IMPLICIT RejectPDU,
     deleteNamedType
       [16] IMPLICIT RejectPDU
ENDIF
IF ( input )
,    input
       [17] IMPLICIT Input-Response
ELSE
,    input
       [17] IMPLICIT RejectPDU
ENDIF
IF ( output )
,  output
       [18] IMPLICIT Output-Response
ELSE
,  output
       [18] IMPLICIT RejectPDU
ENDIF
IF ( takeControl )
,  takeControl
       [19] TakeControl-Response
```

```
ELSE
,    takeControl
        [19] IMPLICIT RejectPDU
ENDIF
IF ( relinquishControl )
,    relinquishControl
        [20] IMPLICIT RelinquishControl-Response
ELSE
,    relinquishControl
        [20] IMPLICIT RejectPDU
ENDIF
IF ( defineSemaphore )
,    defineSemaphore
        [21] IMPLICIT DefineSemaphore-Response
ELSE
,    defineSemaphore
        [21] IMPLICIT RejectPDU
ENDIF
IF ( deleteSemaphore )
,    deleteSemaphore
        [22] IMPLICIT DeleteSemaphore-Response
ELSE
,    deleteSemaphore
        [22] IMPLICIT RejectPDU
ENDIF
IF ( reportSemaphoreStatus )
,    reportSemaphoreStatus
        [23] IMPLICIT ReportSemaphoreStatus-Response
ELSE
,    reportSemaphoreStatus
        [23] IMPLICIT RejectPDU
ENDIF
IF ( reportPoolSemaphoreStatus )
,    reportPoolSemaphoreStatus
        [24] IMPLICIT ReportPoolSemaphoreStatus-Response
ELSE
,    reportPoolSemaphoreStatus
        [24] IMPLICIT RejectPDU
ENDIF
IF ( reportSemaphoreEntryStatus )
,    reportSemaphoreEntryStatus
        [25] IMPLICIT ReportSemaphoreEntryStatus-Response
ELSE
```

```
, reportSemaphoreEntryStatus
      [25] IMPLICIT RejectPDU
ENDIF
IF ( initiateDownloadSequence )
, initiateDownloadSequence
      [26] IMPLICIT InitiateDownloadSequence-Response,
    downloadSegment
      [27] IMPLICIT DownloadSegment-Response,
    terminateDownloadSequence
      [28] IMPLICIT TerminateDownloadSequence-Response
ELSE
, initiateDownloadSequence
      [26] IMPLICIT RejectPDU,
    downloadSegment
      [27] IMPLICIT RejectPDU,
    terminateDownloadSequence
      [28] IMPLICIT RejectPDU
ENDIF
IF ( initiateUploadSequence )
, initiateUploadSequence
      [29] IMPLICIT InitiateUploadSequence-Response,
    uploadSegment
      [30] IMPLICIT UploadSegment-Response,
    terminateUploadSequence
      [31] IMPLICIT TerminateUploadSequence-Response
ELSE
, initiateUploadSequence
      [29] IMPLICIT RejectPDU,
    uploadSegment
      [30] IMPLICIT RejectPDU,
    terminateUploadSequence
      [31] IMPLICIT RejectPDU
ENDIF
IF ( requestDomainDownload )
,   requestDomainDownload
      [32] IMPLICIT RequestDomainDownload-Response
ELSE
,   requestDomainDownload
      [32] IMPLICIT RejectPDU
ENDIF
IF ( requestDomainUpload )
,   requestDomainUpload
      [33] IMPLICIT RequestDomainUpload-Response
```

```
ELSE
,   requestDomainUpload
        [33] IMPLICIT RejectPDU
ENDIF
IF ( loadDomainContent )
,   loadDomainContent
        [34] IMPLICIT LoadDomainContent-Response
ELSE
,   loadDomainContent
        [34] IMPLICIT RejectPDU
ENDIF
IF ( storeDomainContent )
,   storeDomainContent
        [35] IMPLICIT StoreDomainContent-Response
ELSE
,    storeDomainContent
        [35] IMPLICIT RejectPDU
ENDIF
IF ( deleteDomain )
,   deleteDomain
        [36] IMPLICIT DeleteDomain-Response
ELSE
,   deleteDomain
        [36] IMPLICIT RejectPDU
ENDIF
IF ( getDomainAttributes )
,   getDomainAttributes
        [37] IMPLICIT GetDomainAttributes-Response
ELSE
,   getDomainAttributes
        [37] IMPLICIT RejectPDU
ENDIF
IF ( createProgramInvocation )
,   createProgramInvocation
        [38] IMPLICIT CreateProgramInvocation-Response
ELSE
,   createProgramInvocation
        [38] IMPLICIT RejectPDU
ENDIF
IF ( deleteProgramInvocation )
,   deleteProgramInvocation
        [39] IMPLICIT DeleteProgramInvocation-Response
ELSE
```

```
, deleteProgramInvocation
      [39] IMPLICIT RejectPDU
ENDIF
IF ( start )
, start
      [40] IMPLICIT Start-Response
ELSE
, start
      [40] IMPLICIT RejectPDU
ENDIF
IF ( stop )
, stop
      [41] IMPLICIT Stop-Response
ELSE
, stop
      [41] IMPLICIT RejectPDU
ENDIF
IF ( resume )
, resume
      [42] IMPLICIT Resume-Response
ELSE
, resume
      [42] IMPLICIT RejectPDU
ENDIF
IF ( reset )
, reset
      [43] IMPLICIT Reset-Response
ELSE
, reset
      [43] IMPLICIT RejectPDU
ENDIF
IF ( kill )
, kill
      [44] IMPLICIT Kill-Response
ELSE
, kill
      [44] IMPLICIT RejectPDU
ENDIF
IF ( getProgramInvocationAttributes )
, getProgramInvocationAttributes
      [45] IMPLICIT GetProgramInvocationAttributes-Respo
ELSE
, getProgramInvocationAttributes
```

```
        [45] IMPLICIT RejectPDU
ENDIF
IF ( obtainFile )
,   obtainFile
        [46] IMPLICIT ObtainFile-Response
ELSE
,   obtainFile
        [46] IMPLICIT RejectPDU
ENDIF
IF ( defineEventCondition )
,    defineEventCondition
        [47] IMPLICIT DefineEventCondition-Response
ELSE
,   defineEventCondition
        [47] IMPLICIT RejectPDU
ENDIF
IF ( deleteEventCondition )
,   deleteEventCondition
        [48] IMPLICIT DeleteEventCondition-Response
ELSE
,   deleteEventCondition
        [48] IMPLICIT RejectPDU
ENDIF
IF ( getEventConditionAttributes )
,   getEventConditionAttributes
        [49] IMPLICIT GetEventConditionAttributes-Response
ELSE
,   getEventConditionAttributes
        [49] IMPLICIT RejectPDU
ENDIF
IF ( reportEventConditionStatus )
,   reportEventConditionStatus
        [50] IMPLICIT ReportEventConditionStatus-Response
ELSE
,    reportEventConditionStatus
        [50] IMPLICIT RejectPDU
ENDIF
IF ( alterEventConditionMonitoring )
,   alterEventConditionMonitoring
        [51] IMPLICIT AlterEventConditionMonitoring-Response
ELSE
,   alterEventConditionMonitoring
        [51] IMPLICIT RejectPDU
```

```
ENDIF
IF ( triggerEvent )
,  triggerEvent
       [52] IMPLICIT TriggerEvent-Response
ELSE
,    triggerEvent
       [52] IMPLICIT RejectPDU
ENDIF
IF ( defineEventAction )
,  defineEventAction
       [53] IMPLICIT DefineEventAction-Response
ELSE
,    defineEventAction
       [53] IMPLICIT RejectPDU
ENDIF
IF ( deleteEventAction )
,  deleteEventAction
       [54] IMPLICIT DeleteEventAction-Response
ELSE
,  deleteEventAction
       [54] IMPLICIT RejectPDU
ENDIF
IF ( getEventActionAttributes )
,  getEventActionAttributes
       [55] IMPLICIT GetEventActionAttributes-Response
ELSE
,  getEventActionAttributes
       [55] IMPLICIT RejectPDU
ENDIF
IF ( reportEventActionStatus )
,    reportEventActionStatus
       [56] IMPLICIT ReportEventActionStatus-Response
ELSE
,  reportEventActionStatus
       [56] IMPLICIT RejectPDU
ENDIF
IF ( defineEventEnrollment )
,    defineEventEnrollment
       [57] IMPLICIT DefineEventEnrollment-Response
ELSE
,  defineEventEnrollment
       [57] IMPLICIT RejectPDU
ENDIF
```

```
IF ( deleteEventEnrollment )
,  deleteEventEnrollment
       [58] IMPLICIT DeleteEventEnrollment-Response
ELSE
,    deleteEventEnrollment
       [58] IMPLICIT RejectPDU
ENDIF
IF ( alterEventEnrollment )
,  alterEventEnrollment
       [59] IMPLICIT AlterEventEnrollment-Response
ELSE
,    alterEventEnrollment
       [59] IMPLICIT RejectPDU
ENDIF
IF ( reportEventEnrollmentStatus )
,    reportEventEnrollmentStatus
       [60] IMPLICIT ReportEventEnrollmentStatus-Response
ELSE
,  reportEventEnrollmentStatus
       [60] IMPLICIT RejectPDU
ENDIF
IF ( getEventEnrollmentAttributes )
,  getEventEnrollmentAttributes
       [61] IMPLICIT GetEventEnrollmentAttributes-Response
ELSE
,  getEventEnrollmentAttributes
       [61] IMPLICIT RejectPDU
ENDIF
IF ( acknowledgeEventNotification )
,    acknowledgeEventNotification
       [62] IMPLICIT AcknowledgeEventNotification-Response
ELSE
,  acknowledgeEventNotification
       [62] IMPLICIT RejectPDU
ENDIF
IF ( getAlarmSummary )
,  getAlarmSummary
       [63] IMPLICIT GetAlarmSummary-Response
ELSE
,  getAlarmSummary
       [63] IMPLICIT RejectPDU
ENDIF
IF ( getAlarmEnrollmentSummary )
```

```
, getAlarmEnrollmentSummary
      [64] IMPLICIT GetAlarmEnrollmentSummary-Response
ELSE
, getAlarmEnrollmentSummary
      [64] IMPLICIT RejectPDU
ENDIF
IF ( readJournal )
, readJournal
      [65] IMPLICIT ReadJournal-Response
ELSE
,  readJournal
      [65] IMPLICIT RejectPDU
ENDIF
IF ( writeJournal )
, writeJournal
      [66] IMPLICIT WriteJournal-Response
ELSE
, writeJournal
      [66] IMPLICIT RejectPDU
ENDIF
IF ( initializeJournal )
, initializeJournal
      [67] IMPLICIT InitializeJournal-Response
ELSE
,  initializeJournal
      [67] IMPLICIT RejectPDU
ENDIF
IF ( reportJournalStatus )
, reportJournalStatus
      [68] IMPLICIT ReportJournalStatus-Response
ELSE
, reportJournalStatus
      [68] IMPLICIT RejectPDU
ENDIF
IF ( createJournal )
,  createJournal
      [69] IMPLICIT CreateJournal-Response
ELSE
, createJournal
      [69] IMPLICIT RejectPDU
ENDIF
IF ( deleteJournal )
,  deleteJournal
```

```
        [70] IMPLICIT DeleteJournal-Response
ELSE
,   deleteJournal
        [70] IMPLICIT RejectPDU
ENDIF
IF ( getCapabilityList )
,   getCapabilityList
        [71] IMPLICIT GetCapabilityList-Response
ELSE
,   getCapabilityList
        [71] IMPLICIT RejectPDU
ENDIF
        —choices [72] through [77] are reserved for use by services
        —defined in annex D
IF ( fileOpen )
,    fileOpen
        [72] IMPLICIT FileOpen-Response
ELSE
,   fileOpen
        [72] IMPLICIT RejectPDU
ENDIF
IF ( fileRead )
,   fileRead
        [73] IMPLICIT FileRead-Response
ELSE
,    fileRead
        [73] IMPLICIT RejectPDU
ENDIF
IF ( fileClose )
,   fileClose
        [74] IMPLICIT FileClose-Response
ELSE
,    fileClose
        [74] IMPLICIT RejectPDU
ENDIF
IF ( fileRename )
,    fileRename
        [75] IMPLICIT FileRename-Response
ELSE
,   fileRename
        [75] IMPLICIT RejectPDU
ENDIF
IF ( fileDelete )
```

```
, fileDelete
        [76] IMPLICIT FileDelete-Response
ELSE
, fileDelete
        [76] IMPLICIT RejectPDU
ENDIF
IF ( fileDirectory )
, fileDirectory
        [77] IMPLICIT FileDirectory-Response
ELSE
, fileDirectory
        [77] IMPLICIT RejectPDU
ENDIF
, ...
IF ( csr cspi )
, additionalService
        [78] AdditionalService-Response
        —choice [79] is reserved
IF ( getDataExchangeAttributes ),
, getDataExchangeAttributes
        [80] GetDataExchangeAttributes-Response
        —Shall not appear in minor version 1
ENDIF
IF ( exchangeData ),
, exchangeData
        [81] IMPLICIT ExchangeData-Response
        —Shall not appear in minor version 1
ENDIF
IF ( defineAccessControlList ),
, defineAccessControlList
        [82] IMPLICIT DefineAccessControlList-Response
        —Shall not appear in minor version 1 or 2
ENDIF
IF ( getAccessControlListAttributes ),
,  getAccessControlListAttributes
        [83] IMPLICIT GetAccessControlListAttributes-Response
        —Shall not appear in minor version 1 or 2
ENDIF
IF ( reportAccessControlledObjects ),
, reportAccessControlledObjects
        [84] IMPLICIT ReportAccessControlledObjects-Response
        —Shall not appear in minor version 1 or 2
ENDIF
```

```
IF ( deleteAccessControlList ),
,   deleteAccessControlList
        [85] IMPLICIT DeleteAccessControlList-Response
        —Shall not appear in minor version 1 or 2
ENDIF
IF ( changeAccessControl ),
,   changeAccessControl
        [86] IMPLICIT ChangeAccessControl-Response
        —Shall not appear in minor version 1 or 2
ENDIF
,   ...
    }
```

ConfirmedServiceResponse 类型指明该服务的类型及应答。提供的上下文标记指明这个服务类型。通过 ConfirmedServiceResponse 产生式所引用的类型定义，每个服务的定义规定了该服务的应答格式。

7.3.2 AdditionalService - Response(附加服务—应答)

```
  AdditionalService-Response ::= CHOICE {
IF ( csr )
IF ( vMDStop )
    vMDStop
      [0] IMPLICIT VMDStop-Response
ELSE
    vMDStop
      [0] IMPLICIT RejectPDU
ENDIF
IF ( vMDReset )
,   vMDReset
      [1] IMPLICIT VMDReset-Response
ELSE
,   vMDReset
      [1] IMPLICIT RejectPDU
ENDIF
IF ( select )
,   select
      [2] IMPLICIT Select-Response
ELSE
,   select
      [2] IMPLICIT RejectPDU
ENDIF
IF ( alterProgramInvocationAttributes )
,   alterPI
      [3] IMPLICIT AlterProgramInvocationAttributes-Response
ELSE
```

```
,    alterPI
       [3] IMPLICIT RejectPDU
ENDIF
ELSE
,    vMDStop
       [0] IMPLICIT RejectPDU,
     vMDReset
       [1] IMPLICIT RejectPDU,
     select
       [2] IMPLICIT RejectPDU,
     alterPI
       [3] IMPLICIT RejectPDU
ENDIF
IF ( cspi )
IF ( initiateUnitControlLoad )
,    initiateUCLoad
       [4] IMPLICIT InitiateUnitControlLoad-Response
ELSE
,    initiateUCLoad
       [4] IMPLICIT RejectPDU
ENDIF
IF ( unitControlLoadSegment )
,    uCLoad
       [5] IMPLICIT UnitControlLoadSegment-Response
ELSE
,    uCLoad
       [5] IMPLICIT RejectPDU
ENDIF
IF ( unitControlUpload )
,    uCUpload
       [6] IMPLICIT UnitControlUpload-Response
ELSE
,    uCUpload
       [6] IMPLICIT RejectPDU
ENDIF
IF ( startUnitControl )
,    startUC
       [7] IMPLICIT StartUnitControl-Response
ELSE
,    startUC
       [7] IMPLICIT RejectPDU
ENDIF
IF ( stopUnitControl )
```

```
,   stopUC
        [8] IMPLICIT StopUnitControl-Response
ELSE
,   stopUC
        [8] IMPLICIT RejectPDU
ENDIF
IF ( createUnitControl )
,   createUC
        [9] IMPLICIT CreateUnitControl-Response
ELSE
,   createUC
        [9] IMPLICIT RejectPDU
ENDIF
IF ( addToUnitControl )
,   addToUC
        [10] IMPLICIT AddToUnitControl-Response
ELSE
,   addToUC
        [10] IMPLICIT RejectPDU
ENDIF
IF ( removeFromUnitControl )
,   removeFromUC
        [11] IMPLICIT RemoveFromUnitControl-Response
ELSE
,   removeFromUC
        [11] IMPLICIT RejectPDU
ENDIF
IF ( getUnitControlAttributes )
,   getUCAttributes
        [12] IMPLICIT GetUnitControlAttributes-Response
ELSE
,   getUCAttributes
        [12] IMPLICIT RejectPDU
ENDIF
IF ( loadUnitControlFromFile )
,   loadUCFromFile
        [13] IMPLICIT LoadUnitControlFromFile-Response
ELSE
,   loadUCFromFile
        [13] IMPLICIT RejectPDU
ENDIF
IF ( storeUnitControlToFile )
,   storeUCToFile
```

```
        [14] IMPLICIT StoreUnitControlToFile-Response
ELSE
,   storeUCToFile
        [14] IMPLICIT RejectPDU
ENDIF
IF ( deleteUnitControl )
,   deleteUC
        [15] IMPLICIT DeleteUnitControl-Response
ELSE
,   deleteUC
        [15] IMPLICIT RejectPDU
ENDIF
IF ( defineEventConditionList )
,   defineECL
        [16] IMPLICIT DefineEventConditionList-Response
ELSE
,   defineECL
        [16] IMPLICIT RejectPDU
ENDIF
IF ( deleteEventConditionList )
,   deleteECL
        [17] IMPLICIT DeleteEventConditionList-Response
ELSE
,   deleteECL
        [17] IMPLICIT RejectPDU
ENDIF
IF ( addEventConditionListReference )
,   addECLReference
        [18] IMPLICIT AddEventConditionListReference-Response
ELSE
,   addECLReference
        [18] IMPLICIT RejectPDU
ENDIF
IF ( removeEventConditionListReference )
,   removeECLReference
        [19] IMPLICIT RemoveEventConditionListReference-Response
ELSE
,    removeECLReference
        [19] IMPLICIT RejectPDU
ENDIF
IF ( getEventConditionListAttributes )
,   getECLAttributes
        [20] IMPLICIT GetEventConditionListAttributes-Response
```

```
ELSE
,   getECLAttributes
        [20] IMPLICIT RejectPDU
ENDIF
IF ( reportEventConditionListStatus )
,   reportECLStatus
        [21] IMPLICIT ReportEventConditionListStatus-Response
ELSE
,   reportECLStatus
        [21] IMPLICIT RejectPDU
ENDIF
IF ( alterEventConditionListMonitoring )
,   alterECLMonitoring
        [22] IMPLICIT AlterEventConditionListMonitoring-Response
ELSE
,   alterECLMonitoring
        [22] IMPLICIT RejectPDU
ENDIF
ELSE
,   initiateUCLoad
        [4] IMPLICIT RejectPDU,
    uCLoad
        [5] IMPLICIT RejectPDU,
    uCUpload
        [6] IMPLICIT RejectPDU,
    startUC
        [7] IMPLICIT RejectPDU,
    stopUC
        [8] IMPLICIT RejectPDU,
    createUC
        [9] IMPLICIT RejectPDU,
    addToUC
        [10] IMPLICIT RejectPDU,
    removeFromUC
        [11] IMPLICIT RejectPDU,
    getUCAttributes
        [12] IMPLICIT RejectPDU,
    loadUCFromFile
        [13] IMPLICIT RejectPDU,
    storeUCToFile
        [14] IMPLICIT RejectPDU,
    deleteUC
        [15] IMPLICIT RejectPDU,
```

```
    defineECL
      [16] IMPLICIT RejectPDU,
    deleteECL
      [17] IMPLICIT RejectPDU,
    addECLReference
      [18] IMPLICIT RejectPDU,
    removeECLReference
      [19] IMPLICIT RejectPDU,
    getECLAttributes
      [20] IMPLICIT RejectPDU,
    reportECLStatus
      [21] IMPLICIT RejectPDU,
    alterECLMonitoring
      [22] IMPLICIT RejectPDU
ENDIF
    }
```

7.3.3 Response-Detail(应答—细目)

```
  Response-Detail ::= CHOICE {
            —this choice shall be selected if the tag value of the
            —ConfirmedServiceResponse does not match any of the tags below
    otherRequests                NULL
IF ( status )
,   status
      [0] CS-Status-Response
ENDIF
IF ( getProgramInvocationAttributes )
,  getProgramInvocationAttributes
      [45] IMPLICIT CS-GetProgramInvocationAttributes-Response
ENDIF
IF ( getEventConditionAttributes )
,  getEventConditionAttributes
      [49] IMPLICIT CS-GetEventConditionAttributes-Response
ENDIF
    }
```

7.4 Confirmed-Error PDU(确认—错误 PDU)

```
  Confirmed-ErrorPDU ::= SEQUENCE {
    invokeID              [0] IMPLICIT Unsigned32,
IF ( attachToEventCondition attachToSemaphore )
    modifierPosition      [1] IMPLICIT Unsigned32 OPTIONAL,
ENDIF
    serviceError          [2] IMPLICIT ServiceError
    }
```

Confirmed-ErrorPDU 是包含三个元素的序列,这三个元素是:一个无符号整数、一个可选的无符号

整数，以及一个 ServiceError。

invokeID 是一个 32 位的无符号整数，它唯一地标识来自一个应用关联上的一个特定 MMS 用户的所有待完成的确认服务请求中的一个服务请求。在任一时刻，对于任意给定的 invokeID，至多只可能是一个来自一个应用关联上的特定 MMS 用户的待完成的服务请求。invokeID 的值应该是 MMS 用户在应答服务原语(参见 GB/T 16720.1 第 5 章)中提供的值。并且，它指定了引起该服务执行的请求实例。在这个 PDU 中的 invokeID 使 MMS 提供者和 MMS 用户将这个 PDU 与相应的服务请求联系起来。

modifierPosition 是一个 32 位的无符号整数，它在由该 invokeID 指定的 Confirmed-RequestPDU 中的 ListOfModifiers 序列所指定的所有修饰符中，唯一地标识一个修饰符。该参数是从应答服务原语(参见 GB/T 16720.1 第 24 章)的 Service error 参数中的 Modifier Position 子参数导出出来的。

ServiceError 用于指明确认服务的修饰符、或确认服务的错误类型和错误代码。Service Error 参数在 7.4.1 中描述。

7.4.1 ServiceError(服务错误)

```
ServiceError ::= SEQUENCE {
    ErrorClass              [0] CHOICE {
    vmd-state                   [0] IMPLICIT INTEGER {
        other                       (0),
        vmd-state-conflict          (1),
        vmd-operational-problem     (2),
        domain-transfer-problem     (3),
        state-machine-id-invalid    (4)
    } (0..4),
    application-reference       [1] IMPLICIT INTEGER {
        other                       (0),
        application-unreachable     (1),
        connection-lost             (2),
        application-reference-invalid (3),
        context-unsupported         (4)
    } (0..4),
    definition                  [2] IMPLICIT INTEGER {
        other                       (0),
        object-undefined            (1),
        invalid-address             (2),
        type-unsupported            (3),
        type-inconsistent           (4),
        object-exists               (5),
        object-attribute-inconsistent (6)
    } (0..6),
    resource                    [3] IMPLICIT INTEGER {
        other                       (0),
        memory-unavailable          (1),
        processor-resource-unavailable (2),
        mass-storage-unavailable    (3),
```

```
        capability-unavailable                  (4),
        capability-unknown                      (5)
    } (0..5),
    service                     [4] IMPLICIT INTEGER {
        other                                   (0),
        primitives-out-of-sequence              (1),
        object-state-conflict                   (2),
                    —Value 3 reserved for further definition
        continuation-invalid                    (4),
        object-constraint-conflict              (5)
    } (0..5),
    service-preempt             [5] IMPLICIT INTEGER {
        other                                   (0),
        timeout                                 (1),
        deadlock                                (2),
        cancel                                  (3)
    } (0..3),
    time-resolution             [6] IMPLICIT INTEGER {
        other                                   (0),
        unsupportable-time-resolution           (1)
    } (0..1),
    access                      [7] IMPLICIT INTEGER {
        other                                   (0),
        object-access-unsupported               (1),
        object-non-existent                     (2),
        object-access-denied                    (3),
        object-invalidated                      (4)
    } (0..4),
    initiate                    [8] IMPLICIT INTEGER {
        other                                   (0),
                —Values 1 and 2 are reserved for further definition
        max-services-outstanding-calling-insufficient   (3),
        max-services-outstanding-called-insufficient    (4),
        service-CBB-insufficient                (5),
        parameter-CBB-insufficient              (6),
        nesting-level-insufficient              (7)
    } (0..7),
    conclude                    [9] IMPLICIT INTEGER {
        other                                   (0),
        further-communication-required          (1)
    } (0..1)
IF ( cancel )
,   cancel                      [10] IMPLICIT INTEGER {
```

```
        other                           (0),
        invoke-id-unknown               (1),
        cancel-not-possible             (2)
    } (0..2)
ELSE
,   cancel              [10] IMPLICIT NULL
ENDIF
IF ( fileOpen fileClose fileRead fileRename fileDelete fileDirectory obtainFile )
,   file                [11] IMPLICIT INTEGER {
        other                           (0),
        filename-ambiguous              (1),
        file-busy                       (2),
        filename-syntax-error           (3),
        content-type-invalid            (4),
        position-invalid                (5),
        file-access-denied              (6),
        file-non-existent               (7),
        duplicate-filename              (8),
        insufficient-space-in-filestore (9)
    } (0..9)
ELSE
,   file                [11] IMPLICIT NULL
ENDIF
,   others              [12] IMPLICIT INTEGER
    } ,
    additionalCode          [1] IMPLICIT INTEGER OPTIONAL,
    additionalDescription   [2] IMPLICIT VisibleString OPTIONAL,
    serviceSpecificInfo     [3] CHOICE {
IF ( obtainFile )
    obtainFile              [0] IMPLICIT ObtainFile-Error
ELSE
    obtainFile              [0] IMPLICIT NULL
ENDIF
IF ( start )
,   start                   [1] IMPLICIT Start-Error
ELSE
,   start                   [1] IMPLICIT NULL
ENDIF
IF ( stop )
,   stop                    [2] IMPLICIT Stop-Error
ELSE
,   stop                    [2] IMPLICIT NULL
ENDIF
```

```
IF ( resume )
,   resume                  [3] IMPLICIT Resume-Error
ELSE
,   resume                  [3] IMPLICIT NULL
ENDIF
IF ( reset )
,   reset                   [4] IMPLICIT Reset-Error
ELSE
,   reset                   [4] IMPLICIT NULL
ENDIF
IF ( deleteVariableAccess )
,   deleteVariableAccess    [5] IMPLICIT DeleteVariableAccess-Error
ELSE
,   deleteVariableAccess    [5] IMPLICIT NULL
ENDIF
IF ( deleteNamedVariableList )
,   deleteNamedVariableList [6] IMPLICIT DeleteNamedVariableList-Error
ELSE
,   deleteNamedVariableList [6] IMPLICIT NULL
ENDIF
IF ( deleteNamedType )
,   deleteNamedType         [7] IMPLICIT DeleteNamedType-Error
ELSE
,   deleteNamedType             [7] IMPLICIT NULL
ENDIF
IF ( defineEventEnrollment )
,   defineEventEnrollment-Error [8] DefineEventEnrollment-Error
ELSE
,   defineEventEnrollment-Error  [8] IMPLICIT NULL
ENDIF
    —[9] Reserved for use by annex D
IF ( fileRename )
,   fileRename                   [9] IMPLICIT FileRename-Error
ELSE
,   fileRename                   [9] IMPLICIT NULL
ENDIF
IF ( csr cspi )
,   additionalService           [10] AdditionalService-Error
ELSE
,   additionalService           [10] IMPLICIT NULL
ENDIF
IF ( changeAccessControl )
,   changeAccessControl         [11] IMPLICIT ChangeAccessControl-Error
```

```
ELSE
    ChangeAccessControl              [11] IMPLICIT NULL
ENDIF
    } OPTIONAL
  }
```

ServiceError 类型指明错误的类别和代码，它可以提供局部定义的错误代码和错误报文信息，并且为那些在出错时需要传递补充信息的服务提供服务专用信息。errorClass、additionalCode 以及 additionalDescription 参数的值是根据本部分 5.5 中的约定和 GB/T 16720.1 第 24 章中的定义导出出来的。

选取 errorClass CHOICE 是根据应答原语中指定的 ErrorType 参数的 errorClass 子参数来作出的，而对所选的 errorClass 的值的选取则是根据应答服务原语中指定的 ErrorType 参数的 Error Code 子参数来作出的。additionalCode 和 additionalDescription 参数则从具有相同名称的 ErrorType 参数的子参数中导出出来。

如果在 Confirmed-ErrorPDU 中给出了 modifierPosition 参数，那么，serviceSpecific Information 选项就不应出现。

如果在 Confirmed-ErrorPDU 中未给出 modifierPosition 参数，那么，serviceSpecific Information 选项应该从被指定为特定服务的 Result(一)参数的子参数的另一些参数(如果这样的服务专用子参数存在的话)中导出出来。

注：如果一个修饰符导致返回一个 Confirmed-ErrorPDU，那么，就不指定服务专用信息。在处理一个确认服务请求期间，因出现错误而导致 Confirmed-ErrorPDU 返回时，服务专用信息被返回给那些服务，它们提供的信息与相应服务过程所要求的一样。

7.4.2 AdditionalService-Error(附加服务—错误)

```
  AdditionalService-Error ::= CHOICE {
IF ( defineEventConditionList )
    defineEcl                [0] DefineEventConditionList-Error
ELSE
    defineEcl                [0] IMPLICIT NULL
ENDIF
IF ( addEventConditionListReference )
,   addECLReference          [1] AddEventConditionListReference-Error
ELSE
,   addECLReference          [1] IMPLICIT NULL
ENDIF
IF ( removeEventConditionListReference )
,   removeECLReference       [2] RemoveEventConditionListReference-Error
ELSE
,   removeECLReference       [2] IMPLICIT NULL
ENDIF
IF ( initiateUnitControlLoad )
,   initiateUC               [3] InitiateUnitControl-Error
ELSE
,   initiateUC               [3] IMPLICIT NULL
ENDIF
IF ( startUnitControl )
```

```
, startUC                    [4] IMPLICIT StartUnitControl-Error
ELSE
, startUC                    [4] IMPLICIT NULL
ENDIF
IF ( stopUnitControl )
, stopUC                     [5] IMPLICIT StopUnitControl-Error
ELSE
, stopUC                     [5] IMPLICIT NULL
ENDIF
IF ( deleteUnitControl )
, deleteUC                   [6] DeleteUnitControl-Error
ELSE
, deleteUC                   [6] IMPLICIT NULL
ENDIF
IF ( loadUnitControlFromFile )
, loadUCFromFile             [7] LoadUnitControlFromFile-Error
ELSE
, loadUCFromFile             [7] IMPLICIT NULL
ENDIF
    }
```

7.5 公用的 MMS 类型

本章定义几个在本部分其他章节中引用的类型。

7.5.1 TimeOfDay

TimeOfDay ::= OCTET STRING (SIZE(4|6))

TimeOfDay 类型是一个八位位组串(OCTET STRING),TimeOfDay 类型的值可以包含 4 个或 6 个八位位组。第一种格式用当前日期午夜起的毫秒数来指定时间(日期不包含在内),而第二种格式包含时间和日期(表示为 1984 年 1 月 1 日起的相对日期)。在这两种格式中,前 4 个 8 位位组包含的值指明从当前日期午夜起的毫秒数。时间域的值可以通过给这些八位位组编号而导出出来。编号从最后一个 8 位位组的最低有效位(作为第 0 位)开始,到第 1 个 8 位位组的最高有效位(作为第 31 位)结束。每一位被赋予一个数值 2^{**N},其中,N 是在此编号序列中一个位的位置。将被赋予 1 的那些位的数值相加,就可以得到时间值。第 28 位至 31 位永远设置为 0。

TimeOfDay 类型的 8 位位组规定如下。如果该类型的值包含日期(6 个 8 位位组的内容),它应表示(利用 ASN.1 位串记法)为:

'0000ttttttttttttttttttttttttttttddddddddddddddddd'B

如果该类型的值不包含日期(4 个八位位组内容的长度),那么,后两个 8 位位组("dd…d")被省略。在上面给出的位串表示中,"tt…t"是报告当天的相对毫秒数,午夜被表示为 0。而"dd…d"是相对日期,1984 年 1 月 1 日为 0。所有值都是二进制值。

在位串中,一个子域值的最高有效位出现在前,其后,位的有效性随之降低。

实现 Time Of Day 类型的系统应在协议执行一致性语句(参见第 18 章)中规定"tt…t"子域的粒度。

7.5.2 标识符和整数类型

本部分使用类型"Identifier"、"Integer8"、"Integer16"、"Integer32"、"unsigned8"、"unsigned16"以及"unsigned32"。这些类型定义如下:

```
maxIdentifier INTEGER ::= 32
```

```
Identifier ::=
IF (char)
        UTF8String (SIZE(1..maxIdentifier))
ELSE
        VisibleString ( FROM
            ("A"|"a"|"B"|"b"|"C"|"c"|"D"|"d"|"E"|"e"|"F"|"f"|
            "G"|"g"|"H"|"h"|"I"|"i"|"J"|"j"|"K"|"k"|"L"|"l"|
            "M"|"m"|"N"|"n"|"O"|"o"|"P"|"p"|"Q"|"q"|"R"|"r"|
            "S"|"s"|"T"|"t"|"U"|"u"|"V"|"v"|"W"|"w"|"X"|"x"|
            "Y"|"y"|"Z"|"z"|"$"|"_"|"0"|"1"|"2"|"3"|"4"|"5"|
            "6"|"7"|"8"|"9") ) (SIZE(1..maxIdentifier))
ENDIF
  Integer8 ::= INTEGER(-128..127)    --range-128 <= i <= 127
  Integer16 ::= INTEGER(-32768..32767)      --range-32,768 <= i <= 32,767
  Integer32 ::= INTEGER(-2147483648..2147483647)          --range-2**31 <= i <=
2**31-1
  Unsigned8 ::= INTEGER(0..127)     --range 0 <= i <= 127
  Unsigned16 ::= INTEGER(0..32767)          --range 0 <= i <= 32767
  Unsigned32 ::= INTEGER(0..2147483647)                    --range 0 <= i <= 2**31-1
```

MMS 根据标识符产生式定义各种类型的名字(如变量名、类型名等)。标识符长度限定为 32 字符这些字符或者选自 VisibleString 类型定义的字符集(当不支持 CharCBB 时),或者选自 UTF8string 定义的字符集(如果支持 CharCBB)。标识符对大小写是敏感的。

在本部分中,Integer8、Integer16、Integer32、unsigned8、unsigned16 以及 unsigned32 用于表达有范围限制的整数,它们可表达的最大值和最小值在其类型说明之后的注释中规定。

7.5.3 ObjectName

```
  ObjectName ::= CHOICE {
    vmd-specific              [0] IMPLICIT Identifier,
    domain-specific           [1] IMPLICIT SEQUENCE {
        domainID                  Identifier,
        itemID                    Identifier
        },
    aa-specific               [2] IMPLICIT Identifier
    }
```

ObjectName 参数应根据本部分 5.5 中提供的规则,并利用 GB/T 16720.1 第 7 章中的 Object Name 服务参数导出出来

7.5.4 ObjectClass

```
  ObjectClass ::= CHOICE {
    basicObjectClass        [0] IMPLICIT INTEGER {
IF ( vnam )
        namedVariable               (0)
ENDIF
                --value 1 is reserved for definition in Annex E
IF ( vsca )
```

```
,       scatteredAccess             (1)
ENDIF
IF ( vlis )
,       namedVariableList           (2)
ENDIF
IF ( vnam )
,       namedType                   (3)
ENDIF
,       semaphore                   (4),
        eventCondition              (5),
        eventAction                 (6),
        eventEnrollment             (7),
        journal                     (8),
        domain                      (9),
        programInvocation           (10),
        operatorStation             (11),
        dataExchange                (12),
    —Shall not appear in minor version 1
        accessControlList           (13)
    —Shall not appear in minor version 1 or 2
        } (0..13),
    ...
IF ( cspi )
,   csObjectClass       [1] IMPLICIT INTEGER {
        eventConditionList          (0),
        unitControl                 (1) } (0..1)
ENDIF
    }
```

ObjectClass 参数应根据本部分 5.5 中提供的规则，并利用 GB/T 16720.1 第 7 章中 ObjectClass 服务参数的定义导出出来。

7.5.5 ApplicationReference

ApplicationReferenee 类型的格式取决于所使用的通信系统。适用于 ISO 通信的定义请参见附录 A。

7.5.6 MMSString

MMSString 用于存储字符集中用户定义的、适合于他们使用的字符串。它的类型定义如下：

```
  MMSString ::=
IF ( char )
        UTF8String
ELSE
        VisibleString
ENDIF
  MMS255String ::=
IF ( char )
```

```
        UTF8String (SIZE(1..255))
ELSE
        VisibleString (SIZE(1..255))
ENDIF
```

如果未指定 Char，则 MMSstring 提供英语中使用的通用的 94 个字符集。如果指定了 Char，则 MMSstring 允许 UNICODE 的全部规定。

7.5.7 FileName

```
filename ::= SEQUENCE OF GraphicString
```

FileName 类型由图形串序列组成。在文件名的图形串序列中，元素语义的确定是本地事务。为了遵守 GB/T 16720.1 和 GB/T 16720.2 对 FileName 长度和合法字符的规定，系统施加的任何限制应该在系统配置和初始化语句(参见第 25 章)中规定。作为一种最低限度，对于每一个使用本章中定义的 FileName 类型的实施，应支持包含单个元素的 filename，该元素由以字母开头的 1 至 8 个大写字母或数字组成。

注：GB/T 16720.1 及 GB/T 16720.2 不对 FileName 的组成成分定义任何解释。这些成分提供对启动者和应答者透明的有名机制。在虚拟文件存储中定义的成分与这些成分在实际系统中的分配之间的关系由本地执行决定。执行可以将逻辑成分的结构映射到 FileName 的成分，或者，也可以将已有的 FileName 语法映射到只有一个成分名的 FileName。在选择对实际文件的访问路径时，执行可以影响 MMS FileName 的成分，但是，这种选择对于通过 MMS 进行互连的目的来说，其本身并不是可见的。

8 环境和通用管理协议

8.1 引言

本章描述组成环境和通用管理服务的服务的 PDUs，并为实现下列服务规定必须的协议：

Initiate(启动)

Conclude(结束)

Abort(异常中止)

Cancel(取消)

Reject(拒绝)

8.2 Initiate(启动)

启动服务请求、启动服务应答以及启动服务错误的抽象语法分别由 Initiate-RequestPDU 类型、Initiate-ResponsePDU 类型及 Initiate-ErrorPDU 类型规定。下面是对这些类型的规定，其后各节给出它们的描述。对于本章中未给出的显式导出的所有参数，它们的导出在 5.5 中描述。出于向上兼容的目的，作为 Initiate-RequestPDU、Initiate-ResponsePDU 或 Initiate-ErrorPDU 的序列元素而接收的附加有效 ASN.1 标记值将被忽略。

```
Initiate-RequestPDU ::= SEQUENCE {
    localDetailCalling                      [0] IMPLICIT Integer32 OPTIONAL,
    proposedMaxServOutstandingCalling       [1] IMPLICIT Integer16,
    proposedMaxServOutstandingCalled        [2] IMPLICIT Integer16,
    proposedDataStructureNestingLevel       [3] IMPLICIT Integer8 OPTIONAL,
    initRequestDetail                       [4] IMPLICIT SEQUENCE {
        proposedVersionNumber               [0] IMPLICIT Integer16,
        proposedParameterCBB                [1] IMPLICIT ParameterSupportOptions,
        servicesSupportedCalling            [2] IMPLICIT ServiceSupportOptions ,
        ...
IF (csr cspi)
```

```
,       additionalSupportedCalling              [3] IMPLICIT AdditionalSupportOptions
ENDIF
IF (cspi)
,       additionalCbbSupportedCalling           [4] IMPLICIT AdditionalCBBOptions,
        privilegeClassIdentityCalling           [5] IMPLICIT VisibleString
ENDIF
        }
    }
  Initiate-ResponsePDU ::= SEQUENCE {
    localDetailCalled                           [0] IMPLICIT Integer32 OPTIONAL,
    negotiatedMaxServOutstandingCalling         [1] IMPLICIT Integer16,
    negotiatedMaxServOutstandingCalled          [2] IMPLICIT Integer16,
    negotiatedDataStructureNestingLevel         [3] IMPLICIT Integer8 OPTIONAL,
    initResponseDetail                          [4] IMPLICIT SEQUENCE {
        negotiatedVersionNumber                 [0] IMPLICIT Integer16,
        negotiatedParameterCBB                  [1] IMPLICIT
  ParameterSupportOption,
        servicesSupportedCalled                 [2] IMPLICIT ServiceSupportOptions,
        ...
IF (csr cspi)
,       additionalSupportedCalled               [3] IMPLICIT
AdditionalSupportOptions
ENDIF
IF (cspi )
,       additionalCbbSupportedCalled            [4] IMPLICIT AdditionalCBBOptions,
        privilegeClassIdentityCalled            [5] IMPLICIT VisibleString
ENDIF
        }
    }
  Initiate-ErrorPDU ::= ServiceError
```

8.2.1 Initiate-RequestPDU(启动—请求 PDU)

启动服务请求的抽象语法是 Initiate-RequestPDU。

8.2.2 Initiate-ResponsePDU(启动—应答 PDU)

启动服务应答的抽象语法是 Initiate-ResponsePDU。

8.2.2 Initiate-ErrorPDU(启动—错误 PDU)

启动服务错误的抽象语法是 Initiate-ErrorPDU

8.3 Conclude(结束)

结束服务请求、结束服务应答以及结束服务错误的抽象语法分别由 Conclude-RequestPDU 类型、Conclude-ResponsePDU 类型以及 Conclude-ErrorPDU 类型规定。下面是对这些类型的规定，其后各节给出它们的描述。对于本章中未给出的显式导出的所有参数，它们的导出在 5.5 中描述。

```
Conclude-RequestPDU ::= NULL
Conclude-ResponsePDU ::= NULL
Conclude-ErrorPDU ::= ServiceError
```

8.3.1 **Conclude-RequestPDU(结束—请求 PDU)**

结束服务请求的抽象语法是 Conclude-RequestPDU。

8.3.2 **Conclude-ResponsePDU(结束—应答 PDU)**

结束服务应答的抽象语法是 Conclude-ResponsePDU。

8.3.3 **Conclude-ErrorPDU(结束—错误 PDU)**

结束服务错误的抽象语法是 Conclude-ErrorPDU。

8.4 **Abort(异常中止)**

异常中止服务被直接映射到 M-U-ABORT 服务(参见第 24 章)。

8.5 **Cancel(取消)**

取消服务请求、取消服务应答、取消服务错误的抽象语法分别由 Cancel-RequestPDU、Cancel-ResponsePDU 及 Cancel-ErrorPDU 规定。下面是对这些类型的规定,其后各节给出它们的描述。对于本章中未给出显式导出的所有参数,它们的导出在 5.5 中描述。

```
Cancel-RequestPDU ::= Unsigned32—originalInvokeID
Cancel-ResponsePDU ::= Unsigned32—originalInvokeID
Cancel-ErrorPDU ::= SEQUENCE {
    originalInvokeID            [0] IMPLICIT Unsigned32,
    serviceError                [1] IMPLICIT ServiceError
    }
```

8.5.1 **Cancel-RequestPDU(取消—请求 PDU)**

取消服务请求的抽象语法是 Cancel-RequestPDU。

8.5.2 **Cancel-ResponsePDU(取消—应答 PDU)**

取消服务应答的抽象语法是 Cancel-ResponsePDU。

8.5.3 **Cancel-ErrorPDU(取消—错误 PDU)**

取消服务错误的抽象语法是 Cancel-ErrorPDU。

8.6 **Reject(拒绝)**

拒绝服务的抽象语法由 RejectPDU 规定。对于本章中未给出显式导出的所有参数,它们的导出在 5.5 中描述。

```
RejectPDU ::= SEQUENCE {
    originalInvokeID            [0] IMPLICIT Unsigned32 OPTIONAL,
    rejectReason CHOICE {
        confirmed-requestPDU        [1] IMPLICIT INTEGER {
            other                       (0),
            unrecognized-service        (1),
            unrecognized-modifier       (2),
            invalid-invokeID            (3),
            invalid-argument            (4),
            invalid-modifier            (5),
            max-serv-outstanding-exceeded   (6),
                    —Value 7 reserved for further definition
            max-recursion-exceeded      (8),
            value-out-of-range          (9)
            } (0..9),
        confirmed-responsePDU       [2] IMPLICIT INTEGER {
```

```
                other                       (0),
                unrecognized-service        (1),
                invalid-invokeID            (2),
                invalid-result              (3),
                        —Value 4 reserved for further definition
                max-recursion-exceeded      (5),
                value-out-of-range          (6)
                } (0..6),
        confirmed-errorPDU          [3] IMPLICIT INTEGER {
                other                       (0),
                unrecognized-service        (1),
                invalid-invokeID            (2),
                invalid-serviceError        (3),
                value-out-of-range          (4)
                } (0..4),
        unconfirmedPDU              [4] IMPLICIT INTEGER {
                other                       (0),
                unrecognized-service        (1),
                invalid-argument            (2),
                max-recursion-exceeded      (3),
                value-out-of-range          (4)
                } (0..4),
        pdu-error                   [5] IMPLICIT INTEGER {
                unknown-pdu-type            (0),
                invalid-pdu                 (1),
                illegal-acse-mapping        (2)
                },
IF ( cancel )
        cancel-requestPDU           [6] IMPLICIT INTEGER {
                other                       (0),
                invalid-invokeID            (1)
                } (0..1),
        cancel-responsePDU          [7] IMPLICIT INTEGER {
                other                       (0),
                invalid-invokeID            (1)
                } (0..1),
        cancel-errorPDU             [8] IMPLICIT INTEGER {
                other                       (0),
                invalid-invokeID            (1),
                invalid-serviceError        (2),
                value-out-of-range          (3)
                } (0..3),
ELSE
```

```
        cancel-requestPDU           [6] IMPLICIT NULL,
        cancel-responsePDU          [7] IMPLICIT NULL,
        cancel-errorPDU             [8] IMPLICIT NULL,
ENDIF
        conclude-requestPDU         [9] IMPLICIT INTEGER {
            other                       (0),
            invalid-argument            (1)
            } (0..1),
        conclude-responsePDU        [10] IMPLICIT INTEGER {
            other                       (0),
            invalid-result              (1)
            } (0..1),
        conclude-errorPDU           [11] IMPLICIT INTEGER {
            other                       (0),
            invalid-serviceError        (1),
            value-out-of-range          (2)
            } (0..2)
          }
    }
```

拒绝服务的抽象语法是 RejectPDU。拒绝原因(rejectReason)域是从 RejectPDU 类型和服务说明中的 Reject Code(拒绝代码)参数中导出出来的。所取选择应该与注解中标明的服务参数中的 Reject-PDU 类型相匹配,而所取选择的值应该与注解中标明的服务参数中的拒绝代码的值相匹配。

9 条件服务应答协议

9.1 引言

本章描述的协议是实现 GB/T 16720.1 第 9 章定义的服务所必须的。它们包括:

DefineAccessControlList(定义访问控制表);

GetAccessControlListAttributes(获取访问控制表属性);

ReportAccessControlledObjects(报告访问控制对象);

DeleteAccessControlList(删除访问控制表);

ChangeAccessControl(修改访问控制)。

9.2 访问条件

Access Condition 参数的抽象语义是 AccessCondition 类型。该类型已在 GB/T 16720.1 的 9.1.2 中定义。

9.3 DefineAccessControlList(定义访问控制表)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 defineAccessControlList 选择的抽象语法分别由 DefineAccessControlList-Request 类型及 DefineAccessControlList-Response 类型规定。下面是对这些类型的规定,其后各节给出它们的描述。对于本章中未给出显式导出的所有参数,它们的导出在 5.5 中描述。

```
DefineAccessControlList-Request ::= SEQUENCE {
    accessControlListName [0] IMPLICIT Identifier,
    accessControlListElements [1] IMPLICIT SEQUENCE {
        readAccessCondition [0] AccessCondition OPTIONAL,
```

```
        storeAccessCondition [1] AccessCondition OPTIONAL,
        writeAccessCondition [2] AccessCondition OPTIONAL,
        loadAccessCondition [3] AccessCondition OPTIONAL,
        executeAccessCondition [4] AccessCondition OPTIONAL,
        deleteAccessCondition [5] AccessCondition OPTIONAL,
        editAccessCondition [6] AccessCondition OPTIONAL
        }
    }
  DefineAccessControlList-Response ::= NULL
```

9.3.1 DefineAccessControlList-Request

ConfirmedServiceRequest 类型的 defineAccessControlList-Request 选择的抽象语法是 DefineAccessControlList-Request。

9.3.1.1 accessControlListElements

accessControlListElements 域是 DefineAccessControlList. request 原语的 ListofAccess ControlElement(访问控制元素表)参数,它应作为 DefineAccessControlList. indication 的访问控制元素表参数出现。对于 Service Class(服务类别)参数的 7 个可能值的每一个值,这个域将包含 0 个或 1 个 AccessCondition 类型的值。与 Read(读)服务类型相关联的 AccessCondition 应出现在 accessControlListElement 的 readAccessCondition 域中。对于其他的服务类别参数值,依此类推。

9.3.2 DefineAccessControlList-Response

ConfirmedServiceResponse 类型的 defineAccessControlList-Response 选择的抽象语法是 DefineAccessControlList-Response。

9.4 GetAccessControlListAttributes(获取访问控制表属性)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 getAccessControlListAttributes 选择的抽象语法分别由 GetAccessControlListAttributes-Request 类型和 GetAccessControlListAttributes-Response 类型规定。下面是对这些类型的规定,其后各节给出它们的描述。对于本章中未给出显式导出的所有参数,它们的导出在 5.5 中描述。

```
  GetAccessControlListAttributes-Request ::= CHOICE {
    accessControlListName          [0] IMPLICIT Identifier,
    vMD                            [1] IMPLICIT NULL,
    namedObject                    [2] IMPLICIT SEQUENCE {
        objectClass                    [0] ObjectClass,
        objectName                     [1] ObjectName
        }
    }
  GetAccessControlListAttributes-Response ::= SEQUENCE {
    name                         [0] Identifier,
    accessControlListElements    [1] IMPLICIT SEQUENCE {
        readAccessCondition            [0] AccessCondition OPTIONAL,
        storeAccessCondition           [1] AccessCondition OPTIONAL,
        writeAccessCondition           [2] AccessCondition OPTIONAL,
        loadAccessCondition            [3] AccessCondition OPTIONAL,
        executeAccessCondition         [4] AccessCondition OPTIONAL,
        deleteAccessCondition          [5] AccessCondition OPTIONAL,
```

```
        editAccessCondition                [6] AccessCondition OPTIONAL
        },
    vMDuse                          [2] IMPLICIT BOOLEAN,
    references                      [3] IMPLICIT SEQUENCE OF SEQUENCE {
        objectClass                        [0] ObjectClass,
        objectCount                        [1] IMPLICIT INTEGER
        }
IF ( aco )
,   accessControlList           [4] IMPLICIT Identifier OPTIONAL
                            —shall be included if and only if
                            —aco has been negotiated
ENDIF
    }
```

9.4.1 GetAccessControlListAttributes-Request

ConfirmedServiceRequest 类型的 getAccessControlListAttributes-Request 选择的抽象语法是 GetAccessControlListAttributes-Request。

9.4.2 GetAccessControlListAttributes-Response

ConfirmedServiceResponse 类型的 getAccessControlListAttributes-Response 选择的抽象语法是 GetAccessControlListAttributes-Response。

9.4.2.1 accessControlListElements

accessControlListElements 域是 DefineAccessControlList. response 原语的访问控制元素表参数，它应作为 DefineAccessControlList. Confirm 的访问控制元素表参数出现。对于服务类别参数的 7 个可能值的每一个值，这个域将包含 0 个或 1 个 AccessCondition 类型的值。与 Read 服务类型相关联的 AccessCondition 将出现在 accessControlListElements 的 readAccessCondition 域中。对于其他的服务类别参数值，依此类推。

9.4.2.2 控制对象计数

CountsofControlledObjects(控制对象计数)参数的抽象语法是 GetAccessControlListAttributes-Response 类型的 references 域。

对于包含一个或多个对象(这些对象又引用这个访问控制表对象)的每个对象类别，应包含一个 objectClass 和 objectCount 序列，如果在引用这个访问控制表对象的类中没有对象(即计数为 0)，那么，就不包括这个序列。如果 aco 未被商定，那么，此域就不出现。

9.4.2.3 访问控制表

当且仅当 aco CBB 商定后，accessControlList 参数才出现。

9.5 ReportAccessControlledObjects(报告访问受控对象)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 reportAccessControlledObjects 选择的抽象语法分别由 ReportAccessControlledObjects-Request 类型和 ReportAccessControlledObjects-Response 类型规定。下面是对这些类型的规定，其后各节给出它们的描述。对于本章中未给出显式导出的所有参数，它们的导出在 5.5 中描述。

```
ReportAccessControlledObjects-Request ::= SEQUENCE {
    accessControlList               [0] IMPLICIT Identifier,
    objectClass                     [1] ObjectClass,
    continueAfter                   [2] ObjectName OPTIONAL
    }
```

```
ReportAccessControlledObjects-Response ::= SEQUENCE {
    listOfNames                 [0] IMPLICIT SEQUENCE OF ObjectName,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT FALSE
    }
```

9.5.1 ReportAccessControlledObjects-Request

ConfirmedServiceRequest 类型的 reportAccessControlledObjects-Request 选择的抽象语法是 ReportAccessControlledObjects-Request。

9.5.2 ReportAccessControlledObjects-Response

ConfirmedServiceResponse 类型的 reportAccessControlledObjects-Response 选择的抽象语法是 ReportAccessControlledObjects-Response。

9.6 DeleteAccessControlList(删除访问控制表)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 deleteAccessControlList 选择的抽象语法分别由 DeleteAccessControlList-Request 和 DeleteAccessControlList-Response 规定。下面是对这些类型的规定,其后各节给出它们的描述。对于本章中未给出显式导出的所有参数,它们的导出在 5.5 中给出。

```
DeleteAccessControlList-Request ::= Identifier
                    —Name of Access Control List Object
DeleteAccessControlList-Response ::= NULL
```

9.6.1 DeleteAccessControlList-Request

ConfirmedServiceRequest 类型的 deleteAccessControlList-Request 选择的抽象语法是 DeleteAccessControlList-Request。

9.6.2 DeleteAccessControlList-Response

ConfirmedServiceResponse 类型的 deleteAccessControlList-Response 选择的抽象语法是 DeleteAccessControlList-Response。

9.7 ChangeAccessControl(修改访问控制)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 changeAccessControl 选择的抽象语法分别由 ChangeAccessControl-Request 和 ChangeAccessControl-Response 类型规定。下面是对这些类型的规定,其后各节给出它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
ChangeAccessControl-Request ::= SEQUENCE {
    scopeOfChange               CHOICE {
      vMDOnly                       [0] IMPLICIT NULL,
      listOfObjects                 [1] IMPLICIT SEQUENCE {
          objectClass                   [0] ObjectClass,
          objectScope                   [1] CHOICE {
              specific                      [0] IMPLICIT SEQUENCE OF
ObjectName,
                                  —Names of the objects (of class objectClass)
                                  —whose access is to be changed
              aa-specific                       [1] IMPLICIT NULL,
              domain                            [2] IMPLICIT Identifier,
                                  —Name of the Domain whose elements
                                  —are to be changed
```

```
            vmd                                    [3] IMPLICIT NULL
            }
        }
    },
    accessControlListName          [2] IMPLICIT Identifier
                    —name of the AccessControlList Object that contains
                    —the conditions for access control
    }
ChangeAccessControl-Response ::= SEQUENCE {
    numberMatched                  [0] IMPLICIT Unsigned32,
    numberChanged                  [1] IMPLICIT Unsigned32
    }
ChangeAccessControl-Error ::= Unsigned32
```

9.7.1 ChangeAccessControl- Request

ConfirmedServiceRequest 类型的 changeAccessControl-Request 选择的抽象语法是 ChangeAccess Control-Request。

9.7.2 ChangeAccessControl- Response

ConfirmedServicetResponse 类型的 changeAccessControl-Response 选择的抽象语法是 ChangeAccess

Control-Response。

9.7.3 ChangeAccessControl-Error

ServiceError 类型的 changeAccessControl 选择的抽象语法是 ChangeAccess Control-Error。

10 VMD(虚拟制造设备)支持协议

10.1 引言

本章描述 VMD 支持服务的 PDUs,并为实现下列服务规定必需的协议:

Status(状态);

UnsolicitedStatus(非请求状态);

GetNameList(获取名字表);

Identify(标识);

Rename(更名);

GetCapabilityList(获取能力表);

VMDStop (VMD 停止);

VMDReset (VMD 复位)。

10.2 Status Response(状态应答)参数

状态应答参数的抽象语法是 statusResponse(状态应答)类型。

```
StatusResponse ::= SEQUENCE {
    vmdLogicalStatus               [0] IMPLICIT INTEGER {
        state-changes-allowed          (0),
        no-state-changes-allowed       (1),
        limited-services-permitted     (2),
        support-services-allowed       (3)
```

```
    } (0..3),
  vmdPhysicalStatus              [1] IMPLICIT INTEGER {
    operational                      (0),
    partially-operational            (1),
    inoperable                       (2),
    needs-commissioning              (3)
    } (0..3),
  localDetail                    [2] IMPLICIT BIT STRING (SIZE(0..128)) OPTIONAL
  }
```

10.2.1 CS-Status-Response(CS-状态应答)

```
  CS-Status-Response ::= CHOICE {
IF ( csr )
  fullResponse SEQUENCE {
    operationState               [0] IMPLICIT OperationState,
    extendedStatus               [1] IMPLICIT ExtendedStatus,
    extendedStatusMask           [2] IMPLICIT ExtendedStatus DEFAULT '1111'B,
    selectedProgramInvocation     CHOICE {
      programInvocation            [3] IMPLICIT Identifier,
      noneSelected                 [4] IMPLICIT NULL } }
ENDIF
IF ( csr cspi )
,
ENDIF
IF ( cspi )
    noExtraResponse          NULL
ENDIF
    }
```

10.2.2 OperationState(操作状况)

```
  OperationState ::= INTEGER {
    idle                         (0),
    loaded                       (1),
    ready                        (2),
    executing                    (3),
    motion-paused                (4),
    manualInterventionRequired   (5) } (0..5)
```

10.2.3 ExtendedStatus(扩充状态)

```
  ExtendedStatus ::= BIT STRING {
    safetyInterlocksViolated        (0),
    anyPhysicalResourcePowerOn      (1),
    allPhysicalResourcesCalibrated  (2),
    localControl                    (3) } (SIZE(4))
```

10.3 Status(状态)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 status 选择的抽象语法分别 Status-Re-

quest 类型和 Status-Response 类型规定。下面给出这些类型的规定，其后各节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

Status-Request ::= BOOLEAN—Extended Derivation

Status-Response ::= StatusResponse

10.3.1 Status- Request

ConfirmedServiceRequest 的 status 选择的抽象语法是 Status-Request。

10.3.2 Status-Response

ConfirmedServiceResponse 的 status 选择的抽象语法是 Status-Response。

10.4 UnsolicitedStatus(非请求状态)

UnconfirmedService 的 unsolicitedStatus 选择的抽象语法由 UnsolicitedStatus 类型规定。下面给出该类型的规定，下一节是对它的描述。

UnsolicitedStatus ::= StatusResponse

10.4.1 UnsolicitedStatus

UnconfirmedService 的 unsolicitedStatus 选择的抽象语法是 UnsolicitedStatus。

10.5 GetNameList(获取名字表)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 getNameList 选择的抽象语法分别由 GetNameList- Request 类型和 GetNameList- Response 类型规定。下面给出这些类型的规定，其后各节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
GetNameList-Request ::= SEQUENCE {
    objectClass             [0] ObjectClass,
    objectScope             [1] CHOICE {
        vmdSpecific             [0] IMPLICIT NULL,
        domainSpecific          [1] IMPLICIT Identifier,
        aaSpecific              [2] IMPLICIT NULL },
    continueAfter           [2] IMPLICIT Identifier OPTIONAL }
GetNameList-Response ::= SEQUENCE {
    listOfIdentifier        [0] IMPLICIT SEQUENCE OF Identifier,
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

10.5.1 GetNameList-Request

ConfirmedServiceRequest 的 getNameList 选择的抽象语法是 GetNameList- Request。

10.5.1.1 object scope(Object Scope)

如果服务请求原语的 Object Scope 参数的值是 VMD-specific，那么，就应选取 GetName List-Request中的 vmdSpecific 选择。

如果服务请求原语的 Object Scope 参数的值是 Domain-Specific，那么，就应选取 GetName List-Request中的 domainSpecific 选择。domainSpecific 类型的值是从服务请求原语的 Domain Name 参数的值导出。

如果服务请求原语的 Object Scope 参数的值是 AA-Specific，那么，就应选取 GetNameList-Request中的 aaSpecific 选择。

10.5.2 GetNameList-Response

ConfirmedServiceResponse 的 getNameList 选择的抽象语法是 GetNameList- Response。

10.6 Identify(标识)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 identify 选择的抽象语法分别由 Identify-Request 类型和 Identify-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。

对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
Identify-Request ::= NULL
Identify-Response ::= SEQUENCE {
    vendorName              [0] IMPLICIT MMSString,
    modelName               [1] IMPLICIT MMSString,
    revision                [2] IMPLICIT MMSString,
    listOfAbstractSyntaxes  [3] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL
    }
```

10.6.1 Identify-Request

ConfirmedServiceRequest 的 identify 选择的抽象语法是 Identify-Request。

10.6.2 Identify-Response

ConfirmedServiceResponse 的 identify 选择的抽象语法是 Identify- Response。

10.7 Rename(更名)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 rename 选择的抽象语法分别由 Rename-Request类型及 Rename-Response 类型规定。下面给出这两个类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
Rename-Request ::= SEQUENCE {
    objectClass             [0] ObjectClass,
    currentName             [1] ObjectName,
    newIdentifier           [2] IMPLICIT Identifier }
Rename-Response ::= NULL
```

10.7.1 Rename-Request

ConfirmedServiceRequest 的 rename 选择的抽象语法是 Rename-Request。

10.7.2 Rename-Response

ConfirmedServiceResponse 的 rename 选择的抽象语法是 Rename-Response。

10.8 GetCapabilityList(获取能力表)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 getCapabilityList 选择的抽象语法分别由 GetCapabilityList-Request 类型及 GetCapabilityList- Response 类型规定。下面给出这两个类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
GetCapabilityList-Request ::= SEQUENCE {
    continueAfter                MMSString OPTIONAL
    }
GetCapabilityList-Response ::= SEQUENCE {
    listOfCapabilities           [0] IMPLICIT SEQUENCE OF MMSString,
    moreFollows                  [1] IMPLICIT BOOLEAN DEFAULT TRUE
    }
```

10.8.1 GetCapabilityList-Request

ConfirmedServiceRequest 的 getCapabilityList 选择的抽象语法是 GetCapabilityList-Request。

10.8.2 GetCapabilityList- Response

ConfirmedServiceResponse 的 getCapabilityList 选择的抽象语法是 GetCapabilityList- Response。

10.9 VMDStop(VMD 停止)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 vMDStop 选择的抽象语法分别由

VMDStop- Request 类型及 VMDStop-Response 类型规定。下面给出这两种类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

VMDStop-Request ::= NULL

VMDStop-Response ::= NULL

10.9.1 VMDStop—Request

ConfirmedServiceRequest 的 vMDStop 选择的抽象语法是 VMDStop-Request。

10.9.2 VMDStop-Response

ConfirmedServiceResponse 的 vMDStop 选择的抽象语法是 VMDStop-Response。

10.10 VMDReset(VMD 复位)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 vMDReset 选择的抽象语法分别由 VMDReset - Request 类型及 VMDReset -Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

VMDReset-Request ::= BOOLEAN—Extended Derivation

VMDReset-Response ::= StatusResponse

10.10.1 VMDReset—Request

ConfirmedServiceRequest 的 vMDReset 选择的抽象语法是 VMDReset-Request。

10.10.2 VMDReset-Response

ConfirmedServiceResponse 的 vMDReset 选择的抽象语法是 VMDReset-Response。

11 域管理协议

11.1 引言

本章描述在“MMS 服务定义”的“域管理”一章中所定义的服务的服务-专用协议元素。这些元素包括：

InitiateDownloadSequence(启动下载队列)

DownloadSegment(下载段)

TerminateDownloadSequence(终止下载队列)

InitiateUploadSequence(启动上载队列)

UploadSegment(上载段)

TerminateUploadSequence(终止上载队列)

RequestDomainDownload(请求域下载)

RequestDomainUpload(请求域上载)

LoadDomainConfent(装入域内容)

StoreDomainContent(存储域内容)

DeleteDomain(删除域)

GetDomainAttributes(获取域属性)

11.2 InitiateDownloadSequence(启动下载序列)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 initiateDownloadSequence 选择的抽象语法分别由 InitiateDownloadSequence-Request 类型及 InitiateDownloadSequence -Response 类型规定。下面是这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
InitiateDownloadSequence-Request ::= SEQUENCE {
    domainName                  [0] IMPLICIT Identifier,
    listOfCapabilities          [1] IMPLICIT SEQUENCE OF MMSString,
```

```
    sharable                                [2] IMPLICIT BOOLEAN }
  InitiateDownloadSequence-Response ::= NULL
```

11.2.1 InitiateDownloadSequence-Request

ConfirmedServiceRequest 的 initiateDownloadSequence 选择的抽象语法是 Initiate DownloadSequence-Request。

11.2.2 InitiateDownloadSequence -Response

ConfirmedServiceResponse 的 initiateDownloadSequence 选择的抽象语法是 Initiate DownloadSequence-Response。

11.3 DownloadSegment(下载段)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 downloadSegment 选择的抽象语法分别由 DownloadSegment- Request 类型及 DownloadSegment -Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
  DownloadSegment-Request ::= Identifier—Domain Name
  DownloadSegment-Response ::= SEQUENCE {
    loadData                    LoadData,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT TRUE }
  LoadData ::= CHOICE {
    non-coded                   [0] IMPLICIT OCTET STRING,
    coded                       EXTERNAL,
    embedded                    EMBEDDED PDV }
```

11.3.1 DownloadSegment-Request

ConfirmedServiceRequest 的 downloadSequence 选择的抽象语法是 DownloadSegment-Request。

11.3.2 DownloadSegment-Response

ConfirmedServiceResponse 的 downloadSequence 选择的抽象语法是 DownloadSegment-Response。

11.3.2.1 Load Data(装载数据)

DownloadSegment 服务应答的 Load Data 参数的抽象语法是在 OCTET STRING 及 EXTERNAL (或 EMBEDDED PDV)之间选其一。OCTET STRING 指明该参数的值无须进一步描述,对它的解释是本地事务。EXTERNAL 或 EMBEDDED PDV 指明由这个 EXTERNAL 或 EMBEDDED PDV 引用的抽象语法包含用于解释此参数值的编码规则。

11.4 TerminateDownloadSequence(终止下载序列)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 terminateDownloadSequence 选择的抽象语法分别由 TerminateDownloadsequence-Request 类型及 TerminateDownload Sequence-Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
  TerminateDownloadSequence-Request ::= SEQUENCE {
    domainName                          [0] IMPLICIT Identifier,
    discard                             [1] IMPLICIT ServiceError OPTIONAL }
  TerminateDownloadSequence-Response ::= NULL
```

11.4.1 TerminateDownloadSequence - Request

ConfirmedServiceRequest 的 terminateDownloadSequence 选择的抽象语法是 Terminate DownloadSequence-Request。

11.4.1.1 Discard(废弃)

Discard 参数的抽象语法由 ServiceError 提供。如果 Discard 参数出现在 TerminateDownloadSequence 服务请求中,那么,就应该包含 ServiceError 类型,用于提供废弃的原因。如果 TerminateDownloadSequence 中未出现 Discard 参数,那么,就不包含 ServiceError 域。

11.4.2 TerminateDownloadSequence - Response

ConfirmedServiceResponse 的 terminateDownloadSequence 选择的抽象语法是 Terminate DownloadSequence-Response。

11.5 InitiateUploadSequence(启动上载序列)

ConfirmedServiceRequest 及 ConfirmedServiceResponse 的 initiateUploadSequence 选择的抽象语法分别由 InitiateUploadSequence - Request 类型及 InitiateUploadSequence-Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
InitiateUploadSequence-Request ::= Identifier—Domain Name
InitiateUploadSequence-Response ::= SEQUENCE {
    UlsmID                          [0] IMPLICIT Integer32,
    listOfCapabilities              [1] IMPLICIT SEQUENCE OF MMSString }
```

11.5.1 InitiateUploadSequence - Request

ConfirmedServiceRequest 的 initiateUploadSequence 选择的抽象语法是 InitiateUpload Sequence-Request。

11.5.2 InitiateUploadSequence -Response

ConfirmedServiceResponse 的 initiateUploadSequence 选择的抽象语法是 InitiateUpload Sequence-Response。

11.6 UploadSegment(上载段)

ConfirmedServiceRequest 及 ConfirmedServiceResponse 的 uploadSegment 选择的抽象语法分别由 UploadSegment- Request 类型和 UploadSegment -Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
UploadSegment-Request ::= Integer32—ULSM ID
UploadSegment-Response ::= SEQUENCE {
    loadData                    LoadData,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

11.6.1 UploadSegment- Request

ConfirmedServiceRequest 的 uploadSegment 选择的抽象语法是 UploadSegment-Request。

11.6.2 UploadSegment -Response

ConfirmedServiceResponse 的 uploadSegment 选择的抽象语法是 UploadSegment-Response。

11.6.2.1 Load Data

UploadSegment 服务应答的 Load Data 参数的抽象语法是在 OCTET STRING 及 EXTERNAL(或 EMBEDDED PDV)二者之间作一选择。OCTET STRING 指明该参数的值无须进一步描述,对它的解释是本地事务。EXTERNAL 或 EMBEDDED PDV 指明这个 EXTERNAL 或 EMBEDDED PDV 所引用的抽象语法包含用于解释此参数值的编码规则。

11.7 TerminateUploadSequence(终止上载序列)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 terminateUploadSequence 选择的抽象

语法分别由 TerminateUploadSequence-Request 类型和 TerminateUploadSequence-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
TerminateUploadSequence-Request ::= Integer32—ULSM ID
TerminateUploadSequence-Response ::= NULL
```

11.7.1 TerminateUploadSequence - Request

ConfirmedServiceRequest 的 terminateUploadSequence 选择的抽象语法是 Terminate UploadSequence-Request。

11.7.2 TerminateUploadSequence - Response

ConfirmedServiceResponse 的 terminateUploadSequence 选择的抽象语法是 Terminate UploadSequence-Response

11.8 RequestDomainDownload(请求域下载)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 requestDomainDownload 选择的抽象语法分别由 RequestDomainDownload-Request 类型和 RequestDomainDownload-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
RequestDomainDownload-Request ::= SEQUENCE {
    domainName                  [0] IMPLICIT Identifier,
    listOfCapabilities          [1] IMPLICIT SEQUENCE OF MMSString
OPTIONAL,
    sharable                    [2] IMPLICIT BOOLEAN,
    fileName                    [4] IMPLICIT FileName }
RequestDomainDownload-Response ::= NULL
```

11.8.1 RequestDomainDownload-Request

ConfirmedServiceRequest 的 requestDomainDownload 选择的抽象语法是 RequestDomain Download-Request。

如果在服务请求中给出了 List Of Capabilities 参数，并且该参数指定了一个空表，那么，传递带有零个元素的 SEQUENCE OF。如果没有给出此参数，则不传送 SEQUENCE OF 域。

11.8.2 RequesDomainDownload-Response

ConfirmedServiceResponse 的 requestDomainDownload 选择的抽象语法是 RequestDomain Download-Response。

11.9 RequestDomainUpload(请求域上载)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 requestDomainUpload 选择的抽象语法分别由 RequestDomainUpload-Request 类型和 RequestDomainUpload-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
RequestDomainUpload-Request ::= SEQUENCE {
    domainName                  [0] IMPLICIT Identifier,
    fileName                    [1] IMPLICIT FileName }
RequestDomainUpload-Response ::= NULL
```

11.9.1 RequestDomainUpload-Request

ConfirmedServiceRequest 的 requestDomainUpload 选择的抽象语法是 RequestDomain Upload-Request。

11.9.2 **RequestDomainUpload-Response**。

ConfirmedService Response 的 requestDomainUpload 选择的抽象语法是 RequestDomain Upload-Response。

11.10 **LoadDomainContent(装载域内容)**

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 loadDomainContent 选择的抽象语法分别由 LoadDomainContent-Request 类型和 LoadDomainContent-Response 类型规定。这些类型的规定如下,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
LoadDomainContent-Request ::= SEQUENCE {
    domainName                  [0] IMPLICIT Identifier,
    listOfCapabilities          [1] IMPLICIT SEQUENCE OF MMSString OPTIONAL,
    sharable                    [2] IMPLICIT BOOLEAN,
    fileName                    [4] IMPLICIT FileName
IF ( tpy )
,   thirdParty                  [5] IMPLICIT ApplicationReference OPTIONAL
ENDIF
    }
LoadDomainContent-Response ::= NULL
```

11.10.1 **LoadDomainContent-Request**

ConfirmedServiceRequest 的 loadDomainContent 选择的抽象语法是 LoadDomainContent-Request。

如果在服务请求中给出了 ListOfCapabilities 参数,并且该参数指定了一个空表,那么,传递带有零个元素的 SEQUENCE OF。如果没有给出此参数表,则不传送 SEQUENCE OF。

10.10.1.1 **ApplicationReference**

LoadDomainContent 服务的 Third Party(第三方)参数的抽象语法是 ApplicationReference。只有当支持 tpy 参数一致性构造块时,才给出此参数,否则,使用 Third Party 参数是任选的。

11.10.2 **Load Content-Response**

ConfirmedServiceResponse 的 loadDomainContent 选择的抽象语法是 LoadDomain Content-Response。

11.11 **StoreDomainContent(存贮域内容)**

ConfirmedServiceRequest 和 Confirme ServiceResponse 的 storeDomainContent 选择的抽象语法分别由 StoreDomainContent-Request 和 StoreDomainContent-Response 类型规定。这些类型的规定如下,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
StoreDomainContent-Request ::= SEQUENCE {
    DomainName                  [0] IMPLICIT Identifier,
    fileName                    [1] IMPLICIT FileName
IF ( tpy )
,   thirdParty                  [2] IMPLICIT ApplicationReference OPTIONAL
ENDIF
    }
StoreDomainContent-Response ::= NULL
```

11.11.1 **StoreDomainContent-Request**

ConfirmedServiceRequest 的 storeDomainContent 选择的抽象语法是 StoreDomainContent-Request。

11.11.1.1 ApplicationReference

StoreDomainContent 服务的 third Party 参数的抽象语法是 ApplicationReference。只有当支持 tpy 参数一致性构造块时，才给出此参数，而且，使用 third Party 参数是任选的。

11.11.2 StoreDomainContent-Response

ConfirmedServiceResponse 的 storeDomainContent 选择的抽象语法是 StoreDomain Content-Response。

11.12 DeleteDomain(删除域)

ConfirmedServiceRequest 和 Confirmed ServiceResponse 的 deleteDomain 选择的抽象语法分别由 DeleteDomain-Request 类型和 DeleteDomain-Response 类型规定。这些类型的规定如下，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
DeleteDomain-Request ::= Identifier—Domain Name
DeleteDomain-Response ::= NULL
```

11.12.1 DeleteDomain-Request

ConfirmedServiceRequest 的 deleteDomain 选择的抽象语法是 DeletDomain-Request。

11.12.2 DeleteDomain-Response

ConfirmedServicetResponse 的 deleteDomain 选择的抽象语法是 DeleteDomain-Response。

11.13 GetDomainAttributes(获取域属性)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 getDomainAttributes 选择的抽象语法分别由 GetDomainAttributes-Request 类型和 GetDomainAttributes-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
GetDomainAttributes-Request ::= Identifier—Domain Name
GetDomainAttributes-Response ::= SEQUENCE {
    listOfCapabilities          [0] IMPLICIT SEQUENCE OF MMSString,
    state                       [1] IMPLICIT DomainState,
    mmsDeletable                [2] IMPLICIT BOOLEAN,
    sharable                    [3] IMPLICIT BOOLEAN,
    listOfProgramInvocations    [4] IMPLICIT SEQUENCE OF Identifier,
                                —Program Invocation Names
    uploadInProgress            [5] IMPLICIT Integer8
IF ( aco )
,   accessControlList           [6] IMPLICIT Identifier OPTIONAL
                                —Shall not appear in minor version one or two
ENDIF
    }
```

11.13.1 GetDomainAttributes-Request

ConfirmedServiceRequest 的 getDomainAttributes 选择的抽象语法是 GetDomain Attributes-Request。

11.13.2 GetDomainAttributes-Response

ConfirmedServiceResponse 的 getDomainAttributes 选择的抽象语法是 GetDomain Attributes-Response。

11.13.2.1 state

DomainState 类型在 GB/T 16720.1 的第 11 章定义。

11.13.2.2 accessControlList

当且仅当 aco CBB 商定之后，才给出 accessControlList 域。

12 程序调用管理协议

12.1 引言

本章描述的协议是实现 GB/T 16720.1(MMS 服务定义)的“程序调用管理”一章中定义的服务所必须的。它们包括：

CreateProgramInvocation	(建立程序调用)
DeleteProgramInvocation	(删除程序调用)
Start	(启动)
Stop	(停止)
Resume	(恢复)
Reset	(复位)
Kill	(截杀)
GetProgramInvocationAttributes	(获取程序调用属性)
Select	(选取)
AlterProgramInvocationAttributes	(变更程序调用属性)
ReconfigureProgramInvocation	(重新配置程序调用)

12.2 CreateProgramInvocation(建立程序调用)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 createProgramInvocation 选择的抽象语法分别由 CreateProgramInvocation-Request 类型和 CreateProgramInvocation-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
  CreateProgramInvocation-Request ::= SEQUENCE {
    programInvocationName          [0] IMPLICIT Identifier,
    listOfDomainNames              [1] IMPLICIT SEQUENCE OF Identifier,
    reusable                       [2] IMPLICIT BOOLEAN DEFAULT TRUE
IF ( eventNotification )
IF ( getProgramInvocationAttributes )
,   monitorType                    [3] IMPLICIT BOOLEAN OPTIONAL
                    —TRUE indicates PERMANENT monitoring,
                    —FALSE indicates CURRENT monitoring
ENDIF
ENDIF
    }
  CreateProgramInvocation-Response ::= NULL
  CS-CreateProgramInvocation-Request ::= INTEGER {
    normal                         (0),
    controlling                    (1),
    controlled                     (2)
    } (0..2)
```

12.2.1 CreateProgramInvocation-Request

ConfirmedServiceRequest 的 createProgramInvocation 选择的抽象语法是 CreateProgram Invoca-

tion-Request。

12.2.1.1 Monitor(监控器)

ConfirmedServiceRequest 服务的 Monitor 参数的抽象语法是根据是否给出 monitorType 域而导出来的。如果给出 monitorType 域,它指明 Monitor 参数为"真"而 monitorType 域的值则指明服务请求的 monitorType 参数的值,同时指明导出的 Event Enrollment(事件登录)是永久的,还是当前的。如果未给出 monitorType 域,则 Monitor 参数为"假",并且,不需要监控。

12.2.2 CreateProgramInvocation-Response

ConfirmedServiceResponse 的 createProgramInvocation 选择的抽象语法是 CreateProgram Invocation-Response。

12.2.3 CS-CreateProgramInvocation-Request

Request-Detail 的 createProgramInvocation 选择的抽象语法是 CS-CreateProgram Invocation-Request。

12.3 DeleteProgramInvocation(删除程序调用)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 deleteProgramInvocation 选择的抽象语法分别由 DeleteProgramInvocation-Request 类型和 DeleteProgramInvocation-Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
DeleteProgramInvocation-Request ::= Identifier—Program Invocation Name
DeleteProgramInvocation-Response ::= NULL
```

12.3.1 DeleteProgramInvocation-Request

ConfirmedServiceRequest 的 deleteProgramInvocation 选择的抽象语法是 DeleteProgram Invocation-Request

12.3.2 DeleteProgramInvocation-Response

ConfirmedServiceResponse 的 deleteProgramInvocation 选择的抽象语法是 DeleteProgram Invocation-Response。

12.4 Start(开始)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 start 选择的抽象语法分别由 Start-Request类型和 Start-Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
Start-Request ::= SEQUENCE {
  programInvocationName      [0] IMPLICIT Identifier,
executionArgument            CHOICE {
      simpleString               [1] IMPLICIT MMSString,
      encodedString              EXTERNAL,
      embeddedString             EMBEDDED PDV } OPTIONAL }
Start-Response ::= NULL
Start-Error ::= ProgramInvocationState
CS-Start-Request ::= [0] CHOICE {
  normal          NULL,
  controlling     SEQUENCE {
      startLocation          [0] IMPLICIT VisibleString OPTIONAL,
      startCount             [1] StartCount DEFAULT cycleCount: 1
      } }
```

```
StartCount ::= CHOICE {
  noLimit        [0] IMPLICIT NULL,
  cycleCount     [1] IMPLICIT INTEGER,
  stepCount      [2] IMPLICIT INTEGER }
```

12.4.1 **Start-Request**

ConfirmedServiceRequest 的 start 选择的抽象语法是 Start-Request。

12.4.1.1 **Execution Argument(执行自变量)**

启动服务请求的 Execution Argument 参数是在 MMSString 及 EXTERNAL(或 EMBEDDED PDV)之间选其一,MMSString 指明该参数的值无需进一步描述,对它的解释是本地事务。EXTERNAL 或 EMBEDDED PDV 则指明由这个 EXTERNAL 或 EMBEDDED PDV 所引用的抽象语法包含用于解释此参数值的编码规则。

12.4.2 **Start-Response**

ConfirmedServiceResponse 的 start 选择的抽象语法是 Start-Response。

12.4.3 **StartError**

ConfirmedServiceError 类型的 serviceSpecificInformation 选择的 start 选择的抽象语法是 Start-Error,它是 Start. Response 原语的 Result(一)参数的 ProgramInvocationState(program Invocation-State)子参数。并且,它将作为 Start. Confirm 原语(如果发送的话)的 Result(一)参数的Program Invocation State 子参数出现。

12.4.3.1 **programInvocationState**

programInvocationState 域的抽象语法在 GB/T 16720.1 的第 12 章定义。

12.4.4 **CS-Start-Request**

Request-Detail 的 start 选择的抽象语法是 CS-Start-Request。

12.5 **Stop(停止)**

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 stop 选择的抽象语法分别由 Stopt-Request 类型和 Stop-Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
Stop-Request ::= SEQUENCE {
  programInvocationName        [0] IMPLICIT Identifier }
Stop-Response ::= NULL
Stop-Error ::= ProgramInvocationState
```

12.5.1 **Stop-Request**

ConfirmedServiceRequest 的 stop 选择的抽象语法是 Stop-Request。

12.5.2 **Stop-Response**

ConfirmedServiceResponse 的 stop 选择的抽象语法是 Stop-Response。

12.5.3 **Stop-Error**

ConfirmedServiceError 类型的 serviceSpecificoationInformation 选择的 stop 选择的抽象语法是 Stop-Error,它是 Stop. response 原语的 Result(一)参数的 ProgramInvocationState 子参数。并且,它将作为 Stop. Confirm 原语(如果发送的话)的 Result(一)参数的 ProgramInvocationState 子参数出现。

12.6 **Resume(恢复)**

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 resume 选择的抽象语法分别由 Resume-Request 类型和 Resume-Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
Resume-Request ::= SEQUENCE {
```

```
    programInvocationName       [0] IMPLICIT Identifier,
    executionArgument           CHOICE {
        simpleString                    [1] IMPLICIT MMSString,
        encodedString                   EXTERNAL,
        enmbeddedString                 EMBEDDED PDV } OPTIONAL
    }
  Resume-Response ::= NULL
  Resume-Error ::= ProgramInvocationState
  CS-Resume-Request ::= [0] CHOICE {
    Normal                      NULL,
    controlling                 SEQUENCE {
        modeType                        CHOICE {
        continueMode                        [0] IMPLICIT NULL,
        changeMode                          [1] StartCount
        } } }
```

12.6.1 **Resume-Request**

ConfirmedServiceRequest 的 resume 选择的抽象语法是 Resume-Request。

12.6.1.1 **Execution Argument**

恢复服务请求的 Execution Argument 参数是在 MMSString 及 EXTERNAL 或 EMBEDDED PDV 之间选其一，MMSString 指明该参数的值无需进一步描述，对它的解释是本地事务。而 EXTERNAL 或 EMBEDDED PDV 则指明由这个 EXTERNAL 或 EMBEDDED PDV 所引用的抽象语法包含用于解释此参数值的编码规则。

12.6.2 **Resume-Response**

ConfirmedServiceResponse 的 resume 选择的抽象语法是 Resume-Response。

12.6.3 **ResuM-Error**

ConfirmedServiceError 类型的 serviceSpecificInformation 选择的 resume 选择的抽象语法是 Resum-Error，它是 Resum. eresponse 原语的 Result(－)参数的 progranInvocationState(程序调用状况)子参数。并且，它将作为 Resume. Confirm 原语(如果发送的话)的 Result(－)参数的 programInvocationState 子参数出现。

12.6.4 **CS-Resum -Request**

Request-Detail 的 resume 选择的抽象语法是 CS-Resume-Request。

12.7 **Reset(复位)**

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 reset 选择的抽象语法分别由 Reset-Request 类型和 Reset-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
  Reset-Request ::= SEQUENCE {
    programInvocationName       [0] IMPLICIT Identifier }
  Reset-Response ::= NULL
  Reset-Error ::= ProgramInvocationState
```

12.7.1 **Reset-Request**

ConfirmedServiceRequest 的 reset 选择的抽象语法是 Reset-Request。

12.7.2 **Reset-Response**

ConfirmedServiceResponse 的 reset 选择的抽象语法是 Reset-Response。

12.7.3 Reset-Error

ConfirmedServiceError 类型的 ServiceSpecificoationInformation 选择的 reset 选择的抽象语法是 Reset-Error，它是 Reset. response 原语的 Result(—)参数的 programInvocationState 子参数。并且，它将作为 Reset. Confirm 原语(如果发送的话)的 Result(—)参数的 programInvocationState 子参数出现。

12.8 Kill(截杀)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 kill 选择的抽象语法分别由 Kill-Request类型和 Kill-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
Kill-Request ::= SEQUENCE {
    programInvocationName          [0] IMPLICIT Identifier }
Kill-Response ::= NULL
```

12.8.1 Kill-Request

ConfirmedServiceRequest 的 kill 选择的抽象语法是 Kill-Request。

12.8.2 Kill-Response

ConfirmedServiceResponse 的 kill 选择的抽象语法是 Kill-Response。

12.9 GetProgramInvocationAttributes(获取程序调用属性)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 getProgramInvocationAttributes 选择的抽象语法分别由 GetProgramInvocationAttributes-Request 类型和 GetProgramInvocationAttributes-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
GetProgramInvocationAttributes-Request ::= Identifier—Program InvocationName
GetProgramInvocationAttributes-Response ::= SEQUENCE {
    state                          [0] IMPLICIT ProgramInvocationState,
    listOfDomainNames              [1] IMPLICIT SEQUENCE OF Identifier,
    mmsDeletable                   [2] IMPLICIT BOOLEAN,
    reusable                       [3] IMPLICIT BOOLEAN,
    monitor                        [4] IMPLICIT BOOLEAN,
    executionArgument              CHOICE {
        simpleString               [5] IMPLICIT MMSString,
        encodedString                  EXTERNAL,
        embeddedString                 EMBEDDED PDV }
IF ( aco )
,   accessControlList              [6] IMPLICIT Identifier OPTIONAL
                —Shall not appear in minor version one or two
ENDIF
    }
CS-GetProgramInvocationAttributes-Response ::= SEQUENCE {
    errorCode                      [0] IMPLICIT INTEGER,
    control                            [1] CHOICE {
        controlling                        [0] IMPLICIT SEQUENCE {
            controlledPI                       [0] IMPLICIT SEQUENCE OF Identifier,
            programLocation                    [1] IMPLICIT VisibleString OPTIONAL,
            runningMode                        [2] CHOICE {
```

```
            freeRunning                    [0] IMPLICIT NULL,
            cycleLimited                   [1] IMPLICIT INTEGER,
            stepLimited                    [2] IMPLICIT INTEGER }
        },
    controlled                     [1] CHOICE {
        controllingPI                  [0] IMPLICIT Identifier,
        none                           [1] IMPLICIT NULL
    },
    normal                         [2] IMPLICIT NULL } }
```

12.9.1 GetProgramInvocationAttributes-Request

ConfirmedServiceRequest 的 getProgramInvocationAttributes 选择的抽象语法是 GetProgramInvocationAttributes-Request。

12.9.2 GetProgramInvocationAttributes-Response

ConfirmedServiceResponse 的 getProgramInvocationAttributes 选择的抽象语法 GetProgram InvocationAttributes-Response。

12.9.2.1 Execution Argument

获取程序调用属性服务应答的 Execution Argument 参数是在 MMSString 和 EXTERNAL 或 EMBEDDED PDV 之间选其一，MMSString 指明该参数的值无需进一步描述，对它的解释是本地事务。而 EXTERNAL 或 EMBEDDED PDV 则指明由这个 EXTERNAL 或 EMBEDDED PDV 所引用的抽象语法包含用于解释此参数值的编码规则。

12.9.2.2 Access Control List

当且仅当 aco CBB 商定后，AccessControlList 参数才出现。

12.9.3 CS-GetProgramInvocationAttributes-Response

Response-Detail 的 getProgramInvocationAttributes 选择的抽象语法是 CS-GetProgram InvocationAttributes-Response。

12.10 Select(选择)

AdditionalService-Request 和 AdditionalService-Response 的 select 选择的抽象语法分别由 Select-Request 和 Select-Response 规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
Select-Request ::= SEQUENCE {
    controlling             [0] IMPLICIT Identifier OPTIONAL,
    controlled              [1] IMPLICIT SEQUENCE OF Identifier OPTIONAL
        —this field shall appear if and only if the controlling field is included
    }
Select-Response ::= NULL
```

12.10.1 Select-Request

CorfirmedService-Request 的 select 选择的抽象语法是 Select-Request。

12.10.2 Select-Response

CorfirmedService-Response 的 select 选择的抽象语法是 Select-Response。

12.11 AlterProgramInvocationAttributes(变更程序调用属性)

AdditionalService-Request 和 AdditionalService-Response 的 alterProgramInvocoation Attributes 选择的抽象语法分别是由 AlterProgramInvocationAttributes-Request 类型和 Alter ProgramInvocationAttributes-Response 类型规定。下面给出这些类型的规定，其后两节是对它们的描述。对于本章

中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
AlterProgramInvocationAttributes-Request ::= SEQUENCE {
    programInvocation                    [0] IMPLICIT Identifier,
    startCount                           [1] StartCount DEFAULT cycleCount: 1 }
AlterProgramInvocationAttributes-Response ::= NULL
```

12.11.1 AlterProgramInvocationAttributes-Request

ConfirmedServiceRequest 的 alterProgramInvocationAttributes 选择的抽象语法是 AlterProgramInvocationAttributes-Request。

12.11.2 AlterProgramInvocationAttributes-Response

ConfirmedServicetResponse 的 alterProgramInvocationAttributes 选择的抽象语法是 AlterProgramInvocationAttributes-Response。

12.12 ReconfigureProgramInvocation(重新配置程序调用)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 reconfigureProgramInvocation 选择的抽象语法分别由 ReconfigureProgramInvocation-Request 类型和 ReconfigureProgramInvocation-Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
ReconfigureProgramInvocation-Request ::= SEQUENCE {
    oldProgramInvocationName             [0] IMPLICIT Identifier,
    newProgramInvocationName             [1] IMPLICIT Identifier OPTIONAL,
    domainsToAdd                         [2] IMPLICIT SEQUENCE OF Identifier,
    domainsToRemove                      [3] IMPLICIT SEQUENCE OF Identifier }
ReconfigureProgramInvocation-Response ::= NULL
```

12.12.1 ReconfigureProgramInvocation-Request

ConfirmedServiceRequest 的 reconfigureProgramInvocation 选择的抽象语法是 ReconfigureProgramInvocation-Request。

12.12.2 ReconfigureProgramInvocation-Response

ConfirmedServiceResponse 的 reconfigureProgramInvocation 选择的抽象语法是 ReconfigureProgramInvocation-Response。

13 单元控制协议

13.1 引言

本章描述的协议是实现 GB/T 16720.1 第 13 章定义的服务所必须的。它们包括:

InitiateUnitContorlLoad	(启动单元控制装载)
UnitControlLoadSegment	(单元控制装载段)
UnitControlUpload	(单元控制上载)
StartUnitContorl	(开始单元控制)
StopUnitContorl	(停止单元控制)
CreateUnitControl	(建立单元控制)
AddToUnitControl	(向单元控制添加)
RemoveFromUnitControl	(从单元控制移走)
GetUnitControlAttributes	(获取单元控制属性)
LoadUnitControlFromFile	(从文件装载单元控制)
StoreUnitControlToFile	(将单元控制存入文件)

```
DeleteUnitControl                    (删除单元控制)
```

13.2 Control Element(控制元素)

Control Element 是一个复合参数,在一些服务中用来描述单元控制对象的单个元素。

```
ControlElement ::= CHOICE {
  beginDomainDef          [0] SEQUENCE {
      domainName                  [1] IMPLICIT Identifier,
      capabilities                [2] IMPLICIT SEQUENCE OF MMSString,
      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
      } }
```

13.2.1 Monitor

Control Element 参数的 Monitor 参数是根据 MonitorType 字段是否给出而导出出来的。如果给出了该域,则如明 Monitor 参数为"真"。MonitorType 字段的值指明服务请求的 Monitor Type 参数的值,GB/T 16720.1 的 11.2.1.1.4 中的规定。

13.3 InitiateUnitControlLoad(启动单元控制装载)服务

AdditionalService-Request 和 AdditionalService-Response 的 initiateUCLoad 选择的抽象语法分别由 InitiateUnitControlLoad-Request 类型和 InitiateUnitControlLoad-Response 类型规定。AdditionalService-Error 的 initiateUCLoad 选择的抽象语法由 InitiateUnitControl-Error 规定。下面给出这些类型的规定,其后几节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
InitiateUnitControlLoad-Request ::= Identifier—Unit Control Name
InitiateUnitControlLoad-Response ::= NULL
InitiateUnitControl-Error ::= CHOICE {
  domain                    [0] IMPLICIT Identifier,
  programInvocation         [1] IMPLICIT Identifier
}
```

13.3.1 InitiateUnitControlLoad-Request

AdditionalService-Request 类型的 initiateUCLoad 选择的抽象语法是 InitiateUnitControlLoad-Request。

13.3.2 InitiateUnitControlLoad-Response

AdditionalService-Response 类型的 initiateUCLoad 选择的抽象语法是 InitiateUnitControlLoad-

Response。

13.3.3 InitiateUnitControlLoad-Error

AdditionalService-Error 类型的 initiateUCLoad 选择的抽象语法是 InitiateUnitControlLoad-Error。

13.4 UnitControlLoadSegment(单元控制装载段)服务

AdditionalService-Request 和 AdditionalService-Response 的 UCload 选择的抽象语法分别由 UnitControlLoadSegment-Request 类型和 UnitControlLoadSegment-Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

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

13.4.1 UnitControlLoadSegment-Request

AdditionalService-Request 类型的 ucLoad 选择的抽象语法是 UnitControlLoadSegment-Request。

13.4.2 UnitControlLoadSegment-Response

AdditionalService-Response 类型的 ucLoad 选择的抽象语法是 UnitControlLoadSegment-Response。

13.5 UnitControlUpload(单元控制上载)服务

AdditionalService-Request 和 AdditionalService-Response 的 uCUpload 选择的抽象语法分别由 UnitContorlUpload-Request 类型和 UnitContorlUpload-Response 类型规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
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
    }
```

13.5.1 UnitControlUpload-Request

AdditionalService-Request 类型的 uCUpload 选择的抽象语法是 UnitControlUpload-Request。

13.5.2 UnitControluCUpload-Response

AdditionalService-Response 类型的 uCUpload 选择的抽象语法是 UnitControlUpload-Response。

13.6 StartUnitControl(开始单元控制)服务

AdditionalService-Request 和 AdditionalService-Response 的 startUC 选择的抽象语法分别由 StartUnitControl-Request 和 StartUnitControl-Response 规定。下面给出这些类型的规定,其后两节是

对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

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

13.6.1 StartUnitControl-Request

AdditionalService-Request 类型的 startUC 选择的抽象语法是 StartUnitControl-Request。

13.6.2 StartUnitControl-Response

AdditionalService-Response 类型的 startUC 选择的抽象语法是 StartUnitControl-Response。

13.6.3 StartUnitControl-Error

AdditionalService-Error 类型的 startUC 选择的抽象语法是 StartUnitControl-Error,它是 StartUnitControl. response 原语的 Result(—)参数的 ProgramInvocationName(程序调用名)子参数和 programInvocationState 子参数。并且,它也将作为 StartUnitControl. Confirm 原语(如果发送的话)的 Result(—)参数的 programInvocationName 子参数和 programInvocationState 子参数出现。

13.7 StopUnitControl(停止单元控制)服务

AdditionalService-Request 和 AdditionalService-Response 的 stopUC 选择的抽象语法分别由 StopUnitControl-Request 和 StopUnitControl-Response 规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

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

13.7.1 StopUnitControl-Request

AdditionalService-Request 类型的 stopUC 选择的抽象语法是 StopUnitControl-Request。

13.7.2 StopUnitControl-Response

AdditionalService-Response 类型的 stopUC 选择的抽象语法是 StopUnitControl-Response。

13.7.3 StopUnitControl-Error

AdditionalService-Error 的 stopUC 选择的抽象语法是 StopUnitControl-Error,它是 StopUnitControl. response 原语的 Result(—)参数的 programInvocationName 子参数和 programInvocationState 子参数。并且,它将作为 StopUnitControl. Confirm 原语(如果发送的话)的 Result(—)参数的 programInvocationName 子参数和 program InvocationState 子参数出现。

13.8 CreateUnitControl(建立单元控制)服务

AdditionalService-Request 和 AdditionalService-Response 的 createUC 选择的抽象语法分别由 CreateUnitControl-Request 和 CreateUnitControl-Response 规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
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
```

13.8.1 CreateUnitControl-Request

AdditionalService-Request 类型的 createUC 选择的抽象语法是 CreateUnitControl-Request。

13.8.2 CreateUnitControl-Response

AdditionalService-Response 类型的 createUC 选择的抽象语法是 CreateUnitControl-Response。

13.9 AddToUnitControl(加入单元控制)服务

AdditionalService-Request 和 AdditionalService-Response 的 addToUC 选择的抽象语法分别由 AddToUnitControl-Request 和 AddToUnitControl-Response 规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
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
```

13.9.1 AddToUnitControl-Request

AdditionalService-Request 类型的 addToUC 选择的抽象语法是 AddToUnitControl-Request。

13.9.2 AddToUnitControl-Response

AdditionalService-Response 类型的 addToUC 选择的抽象语法是 AddToUnitControl-Response。

13.10 RemoveFromUnitControl(退出单元控制)服务

AdditionalService-Request 和 AdditionalService-Response 的 removeFromUC 选择的抽象语法分别由 RemoveFromUnitControl-Request 和 RemoveFromUnitControl-Response 规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
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
```

13.10.1 RemoveFromUnitControl-Request

AdditionalService-Request 类型的 removeFromUC 选择的抽象语法是 Remove FromUnit Control-Request。

13.10.2 RemoveFromUnitControl-Response

AdditionalService-Response 类型的 removeFromUC 选择的抽象语法是 RemoveFromUnit Control-Response。

13.11 GetUnitControlAttributes(获取单元控制属性)服务

AdditionalService-Request 和 AdditionalService-Response 的 getUnitControlAttributes 选择的抽象语法分别由 GetUnitControlAttributes-Request 和 GetUnitControlAttributes-Response 规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出

在 5.5 中描述。

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

13.11.1 GetUnitControlAttributes-Request

AdditionalService-Request 类型的 getUCAttributes 选择的抽象语法是 GetUnitControlAttributes-Request。

13.11.2 GetUnitControlAttributes-Response

AdditionalService-Response 类型的 getUCAttributes 选择的抽象语法是 GetUnit ControlAttributes-Response。

13.12 LoadUnitControlFromFile(从文件装载单元控制)服务

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 loadUCFromFile 选择的抽象语法分别由 LoadUnitControlFromFile-Request 和 LoadUnitControlFromFile-Response 规定。下面给出这些类型的规定,其后两几节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

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

13.12.1 LoadUnitControlFromFile-Request

AdditionalService-Request 类型的 loadUCFromFile 选择的抽象语法是 LoadUnitControlFromFile-Request。

13.12.2 LoadUnitControlFromFile-Response

AdditionalService-Response 类型的 loadUCFromFile 选择的抽象语法是 LoadUnitControlFrom-File-Response。

13.12.3 LoadUnitControlFromFile-Error

AdditionalService-Error 类型的 loadUCFromFile 选择的抽象语法是 LoadUnitControlFromFile-Error。

13.13 StoreUnitControlToFile(向文件存入单元控制)服务

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 storeUCToFile 选择的抽象语法分别由 StoreUnitControlToFile-Request 和 StoreUnitControlToFile-Response 规定。下面给出这些类型的规定,其后两节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
StoreUnitControlToFile-Request ::= SEQUENCE {
```

```
    unitControlName          [0] IMPLICIT Identifier,
    fileName                 [1] IMPLICIT FileName
IF ( tpy )
,   thirdParty               [2] IMPLICIT ApplicationReference OPTIONAL
ENDIF
    }
  StoreUnitControlToFile-Response ::= NULL
```

13.13.1 StoreUnitControlToFile-Request

AdditionalService-Request 类型的 storeUCToFile 选择的抽象语法是 StoreUnitControlToFile-Request。

13.13.2 StoreUnitControlToFile-Response

AdditionalService-Response 类型的 storeUCToFile 选择的抽象语法是 StoreUnitControlToFile-Response。

13.14 DeleteUnitControl(删除单元控制)服务

AdditionalService-Request 和 AdditionalService-Response 的 deleteUC 选择的抽象语法分别由 DeleteUnitControl-Request 和 DeleteUnitControl-Response 规定。下面给出这些类型的规定,其后几节是对它们的描述。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
  DeleteUnitControl-Request ::= Identifier—Unit Control Name
  DeleteUnitControl-Response ::= NULL
  DeleteUnitControl-Error ::= CHOICE {
    domain                   [0] IMPLICIT Identifier,
    programInvocation        [1] IMPLICIT Identifier }
```

13.14.1 DeleteUnitControl-Request

AdditionalService-Request 类型的 deleteUC 选择的抽象语法是 DeleteUnitControl-Request。

13.14.2 DeleteUnitControl-Response

AdditionalService-Response 类型的 deleteUC 选择的抽象语法是 DeleteUnitControl-Response。

13.14.3 DeleteUnitControl-Error

AdditionalService-Error 类型的 deleteUC 选择的抽象语法是 DeleteUnitControl-Error。

14 变量访问协议

本章描述由 GB/T 16720.1(MMS 服务定义)的变量访问软设施定义的那些服务的服务专用协议元素。它包括的协议是实现下列服务所必须的:

Read	(读)
Write	(写)
InformationReport	(信息报告)
GetVariableAccessAttributes	(获取变量访问属性)
DefineNamedVariable	(定义有名变量)
DeleteVariableAccess	(删除变量访问)
DefineNamedVariableList	(定义有名变量表)
GetNamedVariableListAttributes	(获取有名变量表)
DeletetNamedVariableList	(删除有名变量表)
DefineNamedType	(定义有名类型)
GetNamedTypeAttributes	(获取有名类型属性)

```
    DeletetNamedType                    (删除有名类型)
```

14.1 约定

除非另加说明,否则,本章中的所有协议元素均遵守 5.5 的约定。当这些约定完全不适用时,或者,当有可能存在二义性解释时,应提供进一步地说明。除了 5.5 的约定之外,在本章中一个服务定义的枚举参数与协议之间的关系如下:

a) 选择 CHOICE 类型(通过在一个互不相容的参数表中进行选取)的请求(应答)原语的枚举参数不出现在协议中。而所选参数应该作为 CHOICE 的选取出现,它具有一个导出的引用名,用以指明这个所选参数。这个所选参数应该像该枚举参数那样,出现在指示(确认)原语(如果发送的话)中,它与在请求(应答)原语中取相同的值和所选参数。

b) 当一个枚举参数表示整数类型值时,其参数的值与它在协议中的值之间的对应关系由 ASN.1 注释指明。

一个给定参数只能按以上两种方式之一来定义。

14.2 规定类型的协议

14.2.1 TypeSpecification(类型说明)

TypeSpecification 参数的抽象语法规定如下。对于本章中未提供显式导出的那些参数,它们的导出在 14.1 中描述。

```
  TypeSpecification ::= CHOICE {
    typeName              [0] ObjectName,
    typeDescription       TypeDescription }
```

14.3 规定替代访问的协议

14.3.1 AlternateAccess(替代访问)

AlternateAccess 参数的抽象语法规定如下。对于本章中未提供显式导出的那些参数,它们的导出在 14.1 中说明。

```
  AlternateAccess ::= SEQUENCE OF CHOICE {
    unnamed               AlternateAccessSelection
IF ( str2 )
,   named                 [5] IMPLICIT SEQUENCE {
    componentName             [0] IMPLICIT Identifier,
    access                    AlternateAccessSelection }
ENDIF
    }
  AlternateAccessSelection ::= CHOICE {
    selectAlternateAccess     [0] IMPLICIT SEQUENCE {
        accessSelection           CHOICE {
IF ( str2 )
        component                     [0] IMPLICIT Identifier,
ELSE
    component                         [0] IMPLICIT NULL,
ENDIF
IF ( str1 )
        index                         [1] IMPLICIT Unsigned32,
        indexRange                    [2] IMPLICIT SEQUENCE {
            lowIndex                      [0] IMPLICIT Unsigned32,
```

```
        numberOfElements                [1] IMPLICIT Unsigned32
        },
ELSE
    index                               [1] IMPLICIT NULL,
    indexRange                          [2] IMPLICIT NULL,
ENDIF
    allElements                         [3] IMPLICIT NULL
    },
  alternateAccess           AlternateAccess
  },
 selectAccess                   CHOICE {
IF ( str2 )
  component                         [1] IMPLICIT Identifier,
ELSE
  component                         [1] IMPLICIT NULL,
ENDIF
IF ( str1 )
  index                             [2] IMPLICIT Unsigned32,
  indexRange                        [3] IMPLICIT SEQUENCE {
    lowIndex                            [0] IMPLICIT Unsigned32,
    numberOfElements                    [1] IMPLICIT Unsigned32
    },
ELSE
  index                           [2] IMPLICIT NULL,
  indexRange                      [3] IMPLICIT NULL,
ENDIF
  allElements                     [4] IMPLICIT NULL
  } }
```

AlternateAccess 类型是 AlternateAccess 参数。该参数的 List Of Alternate Access Selection(替代访问选择表)参数的元素应包含在 Alternate Access Sequence-oftype(替代访问后续类型)的相应元素中。根据 list of Alternate Access Selection 参数元素的 ComponetName(分量名)参数是否出现,每个元素或者选择有名选取,或者选择无名选取。

如果 list of Alternate Access Selection 参数的元素规定了 ComponetName 参数,那么,为了表达选择该元素,应选择有名选取。在这种情况下,ComponetName 是参数,access 是 AlternateAccessSelection 类型,该类型按如下说明规定特定选择。

如果 list of Alternate Access Selection 参数的元素没有规定 ComponetName 参数,为了表达选择该元素,应选择无名选取。它是 AlternateAccessSelection 类型,并按如下说明规定特定选择。

AlternateAccessSelection 类型是从 list of Alternate Access Selection 参数的相应元素的参数(不包括 ComponentName)导出而来的,其导出如下:

a) 指定如果 Kind Of Selection 为 SELECT-ALTERNATE-ACCESS,则选 SelectAlternate Acces,该选择的参数导出如下:

   1) 按照 14.1 的约定,accessSelection 字段应从 Access Selecion(访问选择)参数、Component(分量)参数、Index(索引)参数以及 Index Range(索引范围)参数导出。如果 AccessSe-

lection 参数规定为 INDEX-RANGE，并且，LowIndex 和 Number Of Elements 均为 0，那么，作为发送者的一种任选方案，对于 accessSelection 可以选 AllElements 选择，替代选 indexRange。这两种选择之间不存在语义上的差别。

2) 通过递归引用这一过程，alternateAccess 字段从 alternateAccess 参数导出出来。

b) 如果指定 Kind of Selecion 为 SELECT-ACCESS，则选 selectAccess。按照 14.1 的约定，accessSelection 字段应从访问选择参数、分量参数、索引参数以及索引范围参数导出。如果 AccessSelection 参数规定为 INDEX-RANGE，并且，LowIndex 和 NumberOfElements 均为 0，那么，作为发送者的一种选择方案，accessSelection 可以选 AllElements 选择，替代 indexRange 选择。这两种选择之间不存在语义上的差别。

14.4 规定数据值的协议

14.4.1 AccessResult(访问结果)

AccessResult 参数的抽象语法规定如下。对于本章中未提供显式导出的那些参数，他们的导出在 14.1 中说明。

```
AccessResult ::= CHOICE {
    failure             [0] IMPLICIT DataAccessError,
    success             Data }
```

success 域应由请求(应答)原语中的 Access Result 参数的 Success 参数(值为“真”)指明。并且，它将作为指示(确认)原语(如果发送的话)的 Success 参数(取值为“真”)出现。

failure 域由请求(应答)原语中的 Access Result 参数的 success 参数(取值为“假”)指明。它是 AccessResult(访问结果)的 Data Access Error(数据访问错误)参数，并作为指示(确认)原语的 Data Access Error 参数及 success 参数(取值为“假”)出现。

14.4.2 Data(数据)

Data 参数的抽象语法规定如下。对于本章中未提供显式导出的那些参数，他们的导出在 14.1 中说明。

```
Data ::= CHOICE {
    —context tag 0 is reserved for AccessResult
IF ( str1 )
    array                       [1] IMPLICIT SEQUENCE OF Data,
ELSE
    array                       [1] IMPLICIT NULL,
ENDIF
IF ( str2 )
    structure                   [2] IMPLICIT SEQUENCE OF Data,
ELSE
    structure                   [2] IMPLICIT NULL,
ENDIF
    boolean                     [3] IMPLICIT BOOLEAN,
    bit-string                  [4] IMPLICIT BIT STRING,
    integer                     [5] IMPLICIT INTEGER,
    unsigned                    [6] IMPLICIT INTEGER,—shall not be negative
    floating-point              [7] IMPLICIT FloatingPoint,
    —[8] is reserved
    octet-string                [9] IMPLICIT OCTET STRING,
```

```
visible-string          [10] IMPLICIT VisibleString,
generalized-time        [11] IMPLICIT GeneralizedTime,
binary-time             [12] IMPLICIT TimeOfDay,
bcd                     [13] IMPLICIT INTEGER,—shall not be negative
booleanArray            [14] IMPLICIT BIT STRING,
objId                   [15] IMPLICIT OBJECT IDENTIFIER,
...,
mMSString               [16] IMPLICIT MMSString
}
```

14.4.2.1 导出

数据参数的导出如下：

a) 如果 Kind Of Data 等于 ARRAY，则选 array，同时，此字段的内容为 Array(数组)参数的 List Of Data(数据表)参数。数据表的元素将按照在数据表参数中的顺序出现在(array)域中。

当 Array 类型的数据元素是 boolean(布尔)类型时，可以用数据产生式(Data Production)中的 booleanArray 选择取代 array 选择。在这种情况下，Array 的 List of Data 参数的元素(从第 0 个元素至表的最后一个元素)被放置入 booleanArray 的对应编号的二进制位中。“真”值用“1”表示，而“假”值用“0”表示。

booleanArray 和包含 boolean 值的 array 在语义上并无差别。使用 booleanArray 是发送者的一种选择，选择它是为了提高传输效率。而所有接受者都应该为接受布尔数据的任一种形式做好准备。

b) 如果 Kind Of Data 等于 STRUCTURE，则选 structure，同时，此字段的内容为 structure(结构)参数的 List of Data(数据表)参数。元素将按照在 List of Data 参数中的顺序出现在结构(structure)字段中。

c) 如果 Kind Of Data 等于 SIMPLE，那么，应按照 14.1 中规定的那样，对 Class(类别)参数值应选 Data 选择。

14.4.2.2 FloatingPoint(浮点)类型

FloatingPoint ::=OCTET STRING

FloatingPoint 类型定义一种带有区分值的简单类型，它的值可以是正、负实数(包括 0)，并且能表示正无穷及负无穷及 NaN(Not a Number)。FloatingPoint 的可能取值由其表达式限制，描述如下。

FloatingPoint 值被描述为由符号 S、有效数 M、指数 E 及指数宽度 N 组成，其中，N 大于 0，有效数 M 的范围规定如下：

如果 E=0

则 $0.0 \leqslant M < 1.0$

否则

$1.0 \leqslant M < 2.0$

当表示一个 FloatingPoint 值时，这个 FloatingPoint 值应包含以下 4 部分：

a) 指数宽度部分——指定包含在指数部分中的二进制位的位数；

b) 符号部分——指定 FloatingPoint 的符号；

c) 指数部分——指定指数的值；

d) 小数部分——指定(在以 2 为基的表示中)位于二进制小数点右边的有效数域的值。

FloatingPoint 的 4 部分用包含两个或多个八位位组的 COTET STRING 来表示。这个 COTET STRING 的第一个八位位组包含指数宽度部分，它被表示为一个二进制整数。FloatingPoint 的剩余部分用 COTET STRING 的连续的八位位组表示，说明如下：

a) 将连续的八位位组从“0”到“K”编号，第一个八位位组的最高有效位为“0”，最后一个八位位组

的最低有效位为“K”。利用这个编号，按下述说明，将 FloatingPoint 值的各部分的二进制位赋予连续八位位组的二进制位中：

1) 符号部分赋予第“0”位，正号被表示为“0”，负号被表示为“1”。

2) 按照二进制有效位递减的顺序，将指数部分赋予第 1 位至第 n 位。

3) 按照二进制有效位递减的顺序，将小数部分赋予第“n+1”位至第“K”位。

注：对于单个 FloatingPoint 值，可能存在多个表示，因为指数宽度可能有不同的值，对单个 FloatingPoint 值的不同表示未赋予语义上的区别。

b) 值“NaN”由指数部分包含的二进制位都等于“1”，并且，小数部分包含的二进制位至少有 1 个等于“1”的所有值来表示。符号位的值无关紧要。

c) 无穷由指数部分包含的二进制位都等于“1”，并且，小数部分包含的二进制位全都等于“0”的所有值来表示。符号位的值决定无穷的符号。

d) 值“0”由指数部分和小数部分包含的二进制位全都等于“0”的所有值来表示。符号位的值无关紧要。

e) 非零的、有限的 FloatingPoint 数由指数部分至少包含一个等于“0”的二进制位的 FloatingPoint 值来表示。所表示的 FloatingPoint 值由下列公式决定：

$V=-1^{s}*F*2**(2-2^{(N-1)})$ 当 E=0 时

$V=-1^{s}*(F+1)*2**(E-2^{(N-1)}+1)$ 当 E≠0 时

其中：

1) “S”是符号位的值；

2) “E”是指数部分的值；

3) “N”是指数部分中二进制位的位数；

4) “F”是小数部分的各二进制位的加权值之和。

小数部分的最高有效位(按上述 3)的规定，编号为“n+1”的位)的加权值等于该二进制位的值乘以 2^{-1}。小数部分的最低有效位(按上述 3)的规定，编号为第“K”的位)的加权值等于该二进制位的值乘以 $2^{(N-K)}$。

f) 所有其他值都是无效的。

当后续八位位组的个数为 4，并且，头一个八位位组的值为“8”时，后续八位位组的表示与单精度 IEEE 754 浮点表示是相容的。当后续八位位组的个数为 8，并且头一个八位位组的值为“11”时，它与双精度 IEEE 754 浮点表示相容。

由于 GB/T 16720.1 和 GB/T 16720.2 允许在浮点值的小数部分和指数部分中有任意数量的二进制位(包括，但不局限于在 IEEE 754 中定义的那些对格式的规定)，并且，由于实际系统可能不支持这些条款的所有可能的值，因此，不可能保证一个特定值在接收系统中是可表示的。当接收到一个在执行时不可表示的浮点值时，将应用下列规定：

a) 如果指数值是可表示的，而小数值是不可表示的，此时，若小数的不可表示部分的最高有效位包含“1”，则将小数四舍五入为最接近的可表示值，否则，将小数作截断处理。

b) 如果指数值是不可表示的，同时，包含的二进制位不全部等于 1，那么：

1) 如果指数是负的(指数部分小于指数偏移)，则执行四舍五入为 0。否则，

2) 根据浮点值的符号，执行将其舍入为正无穷或负无穷。

c) 如果指数值是不可表示的，同时，其包含的二进制位全部等于 1，那么：

1) 如果小数部分包含的二进制位全部等于 0，那么，根据浮点值的符号，执行表示为正无穷或负无穷。否则，

2) 执行表示为 NaN。

14.4.2.3 BCD 类型

对于 BCD 类型的数据，它们的值被转换为等价的整数值。例如，BCD 值 82，’10000010’B 被转换

为整数值 82,或'1010010'B。

14.4.3 DataAccessError(数据访问错误)

Data Access Error 参数的抽象语法规定如下。对于本章中未提供显式导出的那些参数,它们的导出在 14.1 中描述。

```
DataAccessError ::= INTEGER {
    object-invalidated              (0),
    hardware-fault                  (1),
    temporarily-unavailable         (2),
    object-access-denied            (3),
    object-undefined                (4),
    invalid-address                 (5),
    type-unsupported                (6),
    type-inconsistent               (7),
    object-attribute-inconsistent   (8),
    object-access-unsupported       (9),
    object-non-existent             (10),
    object-value-invalid            (11)
    } (0..11)
```

14.5 规定变量访问的协议

14.5.1 VariableAccessSpecification(变量访问说明)

Variable Access Specification 参数的抽象语法规定如下。对于本章中未提供显式导出的那些参数,它们的导出在 14.1 中描述。

```
VariableAccessSpecification ::= CHOICE {
    listOfVariable              [0] IMPLICIT SEQUENCE OF SEQUENCE {
        variableSpecification       VariableSpecification,
IF ( valt )
        alternateAccess             [5] IMPLICIT AlternateAccess OPTIONAL
ENDIF
        }
IF ( vlis )
,   variableListName            [1] ObjectName
ENDIF
    }
```

14.5.2 VariableSpecification(变量说明)

Variable Specification 参数的抽象语法规定如下。对于本章中未提供显式导出的那些参数,它们的导出在 14.1 中描述。

```
VariableSpecification ::= CHOICE {
IF ( vnam )
    name                    [0] ObjectName,
ENDIF
IF ( vadr )
    address                 [1] Address,
    variableDescription     [2] IMPLICIT SEQUENCE {
```

```
        address                         Address,
        typeSpecification               TypeSpecification
        },
ENDIF
    —the following element is only present to support the services
    —defined in annex E
IF ( vsca )
    scatteredAccessDescription    [3] IMPLICIT ScatteredAccessDescription,
ELSE
    scatteredAccessDescription    [3] IMPLICIT NULL,
ENDIF
    invalidated                   [4] IMPLICIT NULL
    }
```

invalidated 选择由 Kind Of Variable(变量种类)参数值 INVALIDATED 时引起,因此,该选择应该在 Kind of variable 参数取值为 INVALIDATED 时出现。

14.6 Read(读)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 类型的 read 选择的抽象语法规定如下,其描述在后两节中给出来。对于本章中未提供显式导出的那些参数,它们的导出在 14.1 中描述。

```
Read-Request ::= SEQUENCE {
    specificationWithResult          [0] IMPLICIT BOOLEAN DEFAULT FALSE,
    variableAccessSpecification      [1] VariableAccessSpecification }
  Read-Response ::= SEQUENCE {
    variableAccessSpecification       [0] VariableAccessSpecification OPTIONAL,
    listOfAccessResult                [1] IMPLICIT SEQUENCE OF AccessResult }
```

14.6.1 Read-Request

ConfirmedServiceRequest 的 read 选择的抽象语法是 Read-Request 类型。

14.6.2 Read-Response

ConfirmedService Response 的 read 选择的抽象语法是 Read-Response 类型。

14.7 Write(写)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 类型的 write 选择的抽象语法规定如下,其描述在后两节中给出来。对于本章中未提供显式导出的那些参数,它们的导出在 14.1 中描述。

```
  Write-Request ::= SEQUENCE {
    variableAccessSpecification          VariableAccessSpecification,
    listOfData                             [0] IMPLICIT SEQUENCE OF Data }
  Write-Response ::= SEQUENCE OF CHOICE {
    failure                                [0] IMPLICIT DataAccessError,
    success                                [1] IMPLICIT NULL }
```

14.7.1 Write - Request

ConfirmedServiceRequest 的 write 选择的抽象语法是 Write-Request 类型。

14.7.2 Write - Response

ConfirmedService Response 的 write 选择的抽象语法是 Write-Response 类型。

Success 域由 Write. response 原语的 Success 参数取值为“真”时指定,因此,该域应该作为 Write. confirm 原语的 Success 参数(取值为“真”)出现。

failure 域由 Write.Response 原语的 Success 参数取值为“假”时指定，它是 Write. response 原语的 Data Access Error(数值访问错误)参数，并且，作为 Write.confirm 原语的 Success 参数(取值为“假”)及 Data Access Error 参数出现。

14.8 InformationReport(信息报告)

UnconfirmedService 类型的 informationReport 选择的抽象语法规定如下，下一节给出其描述。对于本章中未提供显式导出的那些参数，它们的导出在 14.1 中描述。

```
  InformationReport ::= SEQUENCE {
    variableAccessSpecification          VariableAccessSpecification,
    listOfAccessResult                   [0] IMPLICIT SEQUENCE OF AccessResult }
```

14.8.1 InformationReport

UnconfirmedService 类型的 informationReport 选择的抽象语法是 InformationReport 类型。

注：InformationReport 服务是无确认的。

14.9 GetVariableAccessAttributes(获取变量访问属性)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 getVariableAccessAttributes 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在14.1中描述。

```
  GetVariableAccessAttributes-Request ::= CHOICE {
IF ( vnam )
    name                          [0] ObjectName
IF ( vadr )
,
ENDIF
IF ( vadr)
    address                       [1] Address
ENDIF
}
  GetVariableAccessAttributes-Response ::= SEQUENCE {
    mmsDeletable                  [0] IMPLICIT BOOLEAN,
IF ( vadr )
    address                       [1] Address OPTIONAL,
ENDIF
    typeDescription               [2] TypeDescription
IF ( aco )
,   accessControlList             [3] IMPLICIT Identifier OPTIONAL
            —Shall not appear in minor version one or two
ENDIF
IF ( sem )
,   meaning                       [4] IMPLICIT VisibleString OPTIONAL
ENDIF
    }
```

14.9.1 GetVariableAccessAttributes-Request

ConfirmedServiceRequest 类型的 getVariableAccessAttributes 选择的抽象语法是 GetVariableAccessAttributes-Request 类型。

14.9.2 GetVariableAccessAttributes-Response

ConfirmedServicetResponse 类型的 getVariableAccessAttributes 选择的抽象语法是 GetVariableAccessAttributes-Response 类型。

14.9.2.1 accessControlList

当且仅当 aco CBB 商定后，accessCntrolList 参数才出现。

14.10 DefineNamedVariable(定义有名变量)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 defineNamedVariable 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 14.1 中描述。

```
DefineNamedVariable-Request ::= SEQUENCE {
    variableName                [0] ObjectName,
    address                     [1] Address,
    typeSpecification           [2] TypeSpecification OPTIONAL }
DefineNamedVariable-Response ::= NULL
```

14.10.1 DefineNamedVariable - Request

ConfirmedServiceRequest 类型的 defineNamedVariable 选择的抽象语法是 DefineNamedVariable-Request 类型。

14.10.2 DefineNamedVariable-Response

ConfirmedServiceResponse 类型的 defineNamedVariable 选择的抽象语法是 DefineNamedVariable-Response 类型，其值为 NULL。

14.11 DeleteVariableAccess(删除变量访问)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 deleteVariableAcces 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 14.1 中描述。

```
DeleteVariableAccess-Request ::= SEQUENCE {
    scopeOfDelete               [0] IMPLICIT INTEGER {
        specific                    (0),
        aa-specific                 (1),
        domain                      (2),
        vmd                         (3)
        } (0..3) DEFAULT specific,
    listOfName                  [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    domainName                  [2] IMPLICIT Identifier OPTIONAL }
DeleteVariableAccess-Response ::= SEQUENCE {
    numberMatched               [0] IMPLICIT Unsigned32,
    numberDeleted               [1] IMPLICIT Unsigned32 }
DeleteVariableAccess-Error ::= Unsigned32—numberDeleted
```

14.11.1 DeleteVariableAccess-Request

ConfirmedServiceRequest 类型的 deleteVariableAcces 选择的抽象语法是 DeleteVariableAcces-Request 类型。

14.11.2 DeleteVariableAccess-Response

ConfirmedServiceResponse 类型的 deleteVariableAcces 选择的抽象语法是 DeleteVariableAcces-Response 类型。

14.12 DefineNamedVariableList(定义有名变量表)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 defineNamedVariableList 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 14.1 中描述。

```
DefineNamedVariableList-Request ::= SEQUENCE {
    variableListName                ObjectName,
    listOfVariable                  [0] IMPLICIT SEQUENCE OF SEQUENCE {
        variableSpecification           VariableSpecification
IF (valt)
,       alternateAccess             [5] IMPLICIT AlternateAccess OPTIONAL
ENDIF
        } }
DefineNamedVariableList-Response ::= NULL
```

14.12.1 DefineNamedVariableList-Request

ConfirmedServiceRequest 类型的 defineNamedVariableList 选择的抽象语法是 DefineNamedVariableList-Request 类型。

14.12.2 DefineNamedVariableList-Response

ConfirmedServiceResponse 类型的 defineNamedVariableList 选择的抽象语法是 DefineNamedVariableList-Response 类型,它的值为 NULL。

14.13 GetNamedVariableListAttributes(获取有名变量表属性)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 getNamedVariableListAttributes 选择的规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 14.1 中描述。

```
GetNamedVariableListAttributes-Request ::= ObjectName—VariableListName
GetNamedVariableListAttributes-Response ::= SEQUENCE {
    mmsDeletable                [0] IMPLICIT BOOLEAN,
    listOfVariable              [1] IMPLICIT SEQUENCE OF SEQUENCE {
        variableSpecification       VariableSpecification
IF ( valt )
,       alternateAccess             [5] IMPLICIT AlternateAccess OPTIONAL }
ENDIF
IF ( aco )
,   accessControlList           [2] IMPLICIT Identifier OPTIONAL
            —Shall not appear in minor version one or two
ENDIF
    }
```

14.13.1 GetNamedVariableListAttributes-Request

ConfirmedServiceRequest 类型的 getNamedVariableListAttributes 选择的抽象语法是 GetNamedVariableListAttributes-Request 类型。

14.13.2 GetNamedVariableListAttributes-Response

ConfirmedServicetResponse 类型的 getNamedVariableListAttributes 选择的抽象语法是 GetNamedVariableListAttributes-Response 类型。

14.13.2.1 accessControlList

当且仅当 aco CBB 商定后，accessControlList 参数才出现。

14.14 DeleteNamedVariableList(删除有名变量表)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 deleteNamedVariableList 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 14.1 中描述。

```
DeleteNamedVariableList-Request ::= SEQUENCE {
    scopeOfDelete               [0] IMPLICIT INTEGER {
        specific                    (0),
        aa-specific                 (1),
        domain                      (2),
        vmd                         (3)
        } (0..3) DEFAULT specific,
    listOfVariableListName      [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    domainName                  [2] IMPLICIT Identifier OPTIONAL }
DeleteNamedVariableList-Response ::= SEQUENCE {
    numberMatched               [0] IMPLICIT Unsigned32,
    numberDeleted               [1] IMPLICIT Unsigned32 }
DeleteNamedVariableList-Error ::= Unsigned32—numberDeleted
```

14.14.1 DeleteNamedVariableList-Request

ConfirmedServiceRequest 类型的 deleteNamedVariableList 选择的抽象语法是 DeleteNamedVariableList-Request 类型。

14.14.2 DeleteNamedVariableList-Response

ConfirmedServiceResponse 类型的 deleteNamedVariableList 选择的抽象语法是 DeleteNamedVariableList-Response 类型。

14.15 DefineNamedType(定义有名类型)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 defineNamedType 选择的规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 14.1 中描述。

```
DefineNamedType-Request ::= SEQUENCE {
    typeName                    ObjectName,
    typeSpecification           TypeSpecification }
DefineNamedType-Response ::= NULL
```

14.15.1 DefineNamedType - Request

ConfirmedServiceRequest 类型的 defineNamedType 选择的抽象语法是 DefineNamedType-Request 类型。

14.15.2 DefineNamedType-Response

ConfirmedServiceResponse 类型的 defineNamedType 选择的抽象语法是 DefineNamedType-Response 类型，其值为 NULL。

14.16 GetNamedTypeAttributes(获取有名类型属性)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 getNamedTypeAttributes 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 14.1 中描述。

```
GetNamedTypeAttributes-Request ::= ObjectName —TypeName
```

```
GetNamedTypeAttributes-Response ::= SEQUENCE {
    mmsDeletable                [0] IMPLICIT BOOLEAN,
    typeSpecification           TypeSpecification
IF ( aco )
,   accessControlList           [1] IMPLICIT Identifier OPTIONAL
        —Shall not appear in minor version one or two
ENDIF
IF ( sem )
,   meaning                     [4] IMPLICIT VisibleString OPTIONAL
ENDIF
    }
```

14.16.1 GetNamedTypeAttributes-Request

ConfirmedServiceRequest 类型的 getNamedTypeAttributes 选择的抽象语法是 GetNamedTypeAttributes-Request 类型。

14.16.2 GetNamedTypeAttributes-Response

ConfirmedServicetResponse 类型的 getNamedTypeAttributes 选择的抽象语法是 Get NamedTypeAttributes-Response 类型。

14.16.2.1 accessControlList

当且仅当 aco CBB 商定后，accessControlList 参数才出现。

14.17 DeleteNamedType(删除有名类型)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 deleteNamedType 选择的规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 14.1 中描述。

```
DeleteNamedType-Request ::= SEQUENCE {
    scopeOfDelete               [0] IMPLICIT INTEGER {
        specific                    (0),
        aa-specific                 (1),
        domain                      (2),
        vmd                         (3)
        } (0..3) DEFAULT specific,
    listOfTypeName              [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    domainName                  [2] IMPLICIT Identifier OPTIONAL }
    DeleteNamedType-Response ::= SEQUENCE {
    numberMatched               [0] IMPLICIT Unsigned32,
    numberDeleted               [1] IMPLICIT Unsigned32
    }
DeleteNamedType-Error ::= Unsigned32—numberDeleted
```

14.17.1 DeleteNamedType - Request

ConfirmedServiceRequest 类型的 defineNamedType 选择的抽象语法是 DeleteNamedTypeRequest 类型。

14.17.2 DeleteNamedType - Response

ConfirmedServiceResponse 类型的 deleteNamedType 选择的抽象语法是 DeleteNamedType-Response 类型。

15 数据交换协议

15.1 引言

本章描述的协议是实现 GB/T 16720.1 第 15 章定义的服务所必须的。它们包括：

Get DataExchangeAttributes(获取数据交换属性)

ExchangeData(交换数据)

15.2 ExchangeData(交换数据)

ConfirmedServiceRequest 和 ConfirmedServiceResponse 的 exchangeData 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
ExchangeData-Request ::= SEQUENCE {
    dataExchangeName            [0] ObjectName,
    listOfRequestData           [1] IMPLICIT SEQUENCE OF Data }
ExchangeData-Response ::= SEQUENCE {
    listOfResponseData          [0] IMPLICIT SEQUENCE OF Data }
```

15.2.1 ExchangeData-Request

ConfirmedServiceRequest 类型的 exchangeData 选择的抽象语法是 ExchangeData-Request 类型。

15.2.2 Exchange Data-Response

ConfirmedServiceResponse 类型的 exchangeData 选择的抽象语法是 ExchangeData-Response 类型。

15.3 GetDataExchangeAttributes(获取数据交换属性)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 getDataExchangeAttributes 选择的抽象语法规定如下，其描述在后几节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
GetDataExchangeAttributes-Request ::= ObjectName
GetDataExchangeAttributes-Response ::= SEQUENCE {
    inUse                                   [0] IMPLICIT BOOLEAN,
    listOfRequestTypeDescriptions           [1] IMPLICIT SEQUENCE OF TypeDescription,
    listOfResponseTypeDescriptions          [2] IMPLICIT SEQUENCE OF TypeDescription,
    programInvocation                       [3] IMPLICIT Identifier OPTIONAL
IF (aco)
,   accessControlList                       [4] IMPLICIT Identifier OPTIONAL
ENDIF
                -—Shall not appear in minor version one or two
    }
```

15.3.1 GetDataExchangeAttributes-Request

ConfirmedServiceRequest 类型的 getDataExchangeAttributes 选择的抽象语法是 Get DataExchangeAttributes-Request。

15.3.2 GetDataExchangeAttributes-Response

ConfirmedServiceResponse 类型的 getDataExchangeAttributes 选择的抽象语法是 Get DataExchangeAttributes-Response。

15.3.2.1 程序调用

GetDataexchangeAttributes-Response 的 ProgramInvocation(程序调用)元素的存在表明数据交换对象的连接属性值为“真”。如果给出，该参数值将传送由数据交换对象的程序调用引用 ProgramInvo-

cationReference(程序调用引用)属性所引用的程序调用名。

15.3.2.2 **accessControlList**

当且仅当 aco CBB 商定后,accessControlList 参数才出现。

16 信标管理协议

16.1 引言

本章描述信标管理服务的服务专用协议元素:

TakeControl (取得控制)

RelinquishControl (放弃控制)

DefineSemaphore (定义信标)

DeleteSemaphore (删除信标)

ReportSemaphoreStatus (报告信标状态)

ReportPoolSemaphoreStatus (报告预存信标状态)

ReportSemaphoreEntryStatus (报告信标项状态)

除上述服务以外,本章还描述了附加于信标的修饰符专用协议元素。

除非另加说明,否则,本章的所有协议均遵循 5.5 的约定。当这些协议不适用,或者,当有可能存在二义性解释时,将提供说明。

16.2 TakeControl(取得控制)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 takeControl 选择的抽象语法规定如下,其描述在后几节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
TakeControl-Request ::= SEQUENCE {
    semaphoreName              [0] ObjectName,
    namedToken                 [1] IMPLICIT Identifier OPTIONAL,
    priority                   [2] IMPLICIT Priority DEFAULT normalPriority,
    acceptableDelay            [3] IMPLICIT Unsigned32 OPTIONAL,
    controlTimeOut             [4] IMPLICIT Unsigned32 OPTIONAL,
    abortOnTimeOut             [5] IMPLICIT BOOLEAN OPTIONAL,
    relinquishIfConnectionLost [6] IMPLICIT BOOLEAN DEFAULT TRUE
IF ( tpy )
,   applicationToPreempt       [7] IMPLICIT ApplicationReference OPTIONAL
ENDIF
    }
TakeControl-Response ::= CHOICE {
    noResult                   [0] IMPLICIT NULL,
    namedToken                 [1] IMPLICIT Identifier }
```

16.2.1 TakeControl-Request

ConfirmedServiceRequest 类型的 takeControl 选择的抽象语法是 TakeControl-Request。

namedToken 域是 TakeControl. request 原语的 Named Token(有名令牌)参数,它将作为 TakeControl. confirme 原语的 Named Token 参数出现。

如果确认原语中的 Acceptable Delay(可接受延迟)参数的值是一个整数,那么,该值将作为 Acceptable Delay(可接受延迟)域的值出现。如果 Acceptable Delay 参数之值为 FOREVER,那么,Acceptable Delay 域将不出现。

如果确认原语中的 Control Time out(控制超时)参数的值是一个整数,那么这个值将作为 Control Time out 字段的值出现。如果 Control Time out 参数的值是 FOREVER,那么,Control Time out 字段将不出现。

当且仅当确认原语中给出了 Abort On Time Out(超时故障)字段时,abortOnTimeOut 域才出现。

16.2.2 TakeControl-Response

ConfirmedServiceResponse 类型的 takeControl 选择的抽象语法是 TakeControl-Response。

16.3 RelinquishControl(放弃控制)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 relinquishControl 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
RelinquishControl-Request ::= SEQUENCE {
    semaphoreName               [0] ObjectName,
    namedToken                  [1] IMPLICIT Identifier OPTIONAL }
RelinquishControl-Response ::= NULL
```

16.3.1 RelinquishControl-Request

ConfirmedServiceRequest 类型的 relinquishControl 选择的抽象语法是 RelinquishControl-Request。

16.3.2 RelinquishControl-Response

ConfirmedServiceResponse 类型的 relinquishControl 选择的抽象语法是 RelinquishControl-Response。

16.4 DefineSemaphore(定义信标)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 define Semaphore 选择的抽象语法规定如下,其描述在后几两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
DefineSemaphore-Request ::= SEQUENCE {
    semaphoreName               [0] ObjectName,
    numberOfTokens              [1] IMPLICIT Unsigned16 }
DefineSemaphore-Response ::= NULL
```

16.4.1 DefineSemaphore-Request

ConfirmedServiceRequest 类型的 defineSemaphore 选择的抽象语法是 DefineSemaphore-Request。

16.4.2 DefineSemaphore-Response

ConfirmedServiceResponse 类型的 defineSemaphore 选择的抽象语法是 DefineSemaphore-Response。

16.5 DeleteSemaphore(删除信标)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 deleteSemaphore 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
DeleteSemaphore-Request ::= ObjectName—Semaphore Name
DeleteSemaphore-Response ::= NULL
```

16.5.1 DeleteSemaphore - Request

ConfirmedServiceRequest 类型的 deleteSemaphore 选择的抽象语法是 DeleteSemaphore-Request。

16.5.2 DeleteSemaphore - Response

ConfirmedServiceResponse 类型的 deleteSemaphore 选择的抽象语法是 DeleteSemaphore-Re-

sponse。

16.6 ReportSemaphoreStatus(报告信标状态)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 reportSemaphoreStatus 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
ReportSemaphoreStatus-Request ::= ObjectName—Semaphore Name
ReportSemaphoreStatus-Response ::= SEQUENCE {
    mmsDeletable                [0] IMPLICIT BOOLEAN,
    class                       [1] IMPLICIT INTEGER {
        token                       (0),
        pool                        (1) } (0..1),
    numberOfTokens              [2] IMPLICIT Unsigned16,
    numberOfOwnedTokens         [3] IMPLICIT Unsigned16,
    numberOfHungTokens          [4] IMPLICIT Unsigned16
IF (aco)
,   accessControlList           [5] IMPLICIT Identifier   OPTIONAL
        —Shall not appear in minor version one or two
ENDIF
    }
```

16.6.1 ReportSemaphoreStatus-Request

ConfirmedServiceRequest 类型的 reportSemaphoreStatus 选择的抽象语法是 ReportSemaphoreStatus-Request。

16.6.2 ReportSemaphoreStatus-Response

ConfirmedServiceResponse 类型的 reportSemaphoreStatus 选择的抽象语法是 ReportSemaphoreStatus-Response。

16.6.2.1 AccessControlList

当且仅当 aco CBB 商定后,accessControlList 参数才出现。

16.7 ReportPoolSemaphoreStatus(报告预存信标状态)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 reportPoolSemaphoreStatus 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
ReportPoolSemaphoreStatus-Request ::= SEQUENCE {
    semaphoreName               [0] ObjectName,
    nameToStartAfter            [1] IMPLICIT Identifier OPTIONAL }
ReportPoolSemaphoreStatus-Response ::= SEQUENCE {
    listOfNamedTokens           [0] IMPLICIT SEQUENCE OF CHOICE {
        freeNamedToken              [0] IMPLICIT Identifier,
        ownedNamedToken             [1] IMPLICIT Identifier,
        hungNamedToken              [2] IMPLICIT Identifier },
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT TRUE
    }
```

16.7.1 ReportPoolSemaphoreStatus-Request

ConfirmedServiceRequest 类型的 reportPoolSemaphoreStatus 选择的抽象语法是 ReportPoolSemaphoreStatus-Request。

16.7.2 ReportPoolSemaphoreStatus-Response

ConfirmedServiceResponse 类型的 reportPoolSemaphoreStatus 选择的抽象语法是 ReportPoolSemaphoreStatus-Response。

16.8 ReportSemaphoreEntryStatus(报告信标项状态)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 reportSemaphoreEntryStatus 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
ReportSemaphoreEntryStatus-Request ::= SEQUENCE {
    semaphoreName                [0] ObjectName,
    state                        [1] IMPLICIT INTEGER {
        queued                       (0),
        owner                        (1),
        hung                         (2) } (0..2),
    entryIDToStartAfter          [2] IMPLICIT OCTET STRING OPTIONAL }
ReportSemaphoreEntryStatus-Response ::= SEQUENCE {
    listOfSemaphoreEntry         [0] IMPLICIT SEQUENCE OF SemaphoreEntry,
    moreFollows                  [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

16.8.1 ReportSemaphoreEntryStatus-Request

ConfirmedServiceRequest 类型的 reportPoolSemaphoreStatus 选择的抽象语法是 ReportSemaphoreEntryStatus-Request。

16.8.2 ReportSemaphoreEntryStatus-Response

ConfirmedServiceResponse 类型的 reportSemaphoreEntryStatus 选择的抽象语法是 ReportSemaphoreEntryStatus-Response。

16.8.3 SemaphoreEntry(信标项)

```
SemaphoreEntry ::= SEQUENCE {
    entryID                      [0] IMPLICIT OCTET STRING,
    entryClass                   [1] IMPLICIT INTEGER {
        simple                       (0),
        modifier                     (1) } (0..1),
    applicationReference             [2] ApplicationReference,
    namedToken                       [3] IMPLICIT Identifier OPTIONAL,
    priority                         [4] IMPLICIT Priority DEFAULT normalPriority,
    remainingTimeOut                 [5] IMPLICIT Unsigned32 OPTIONAL,
    abortOnTimeOut                   [6] IMPLICIT BOOLEAN OPTIONAL,
    relinquishIfConnectionLost       [7] IMPLICIT BOOLEAN DEFAULT TRUE }
```

16.9 AttachToSemaphoreModifier(附加信标修饰符)

Modifier 类型的 attachToSemaphore 选择的抽象语法语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
AttachToSemaphore ::= SEQUENCE {
    semaphoreName                [0] ObjectName,
```

```
    namedToken                  [1] IMPLICIT Identifier OPTIONAL,
    priority                    [2] IMPLICIT Priority DEFAULT normalPriority,
    acceptableDelay             [3] IMPLICIT Unsigned32 OPTIONAL,
    controlTimeOut              [4] IMPLICIT Unsigned32 OPTIONAL,
    abortOnTimeOut              [5] IMPLICIT BOOLEAN OPTIONAL,
    relinquishIfConnectionLost  [6] IMPLICIT BOOLEAN DEFAULT TRUE }
```

17 操作员通信协议

17.1 引言

本章描述用于操作员通信服务的 PDUs。更具体地说,本章规定了实现下列服务所必须的协议:

Input(输出)

Output(输出)

17.2 Input(输入)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 input 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
  Input-Request ::= SEQUENCE {
    operatorStationName     [0] IMPLICIT Identifier,
    echo                    [1] IMPLICIT BOOLEAN DEFAULT TRUE,
IF (output)
    listOfPromptData        [2] IMPLICIT SEQUENCE OF MMSString OPTIONAL,
ENDIF
    inputTimeOut            [3] IMPLICIT Unsigned32 OPTIONAL }
  Input-Response ::= MMSString—Input String
```

17.2.1 Input-Request

ConfirmedServiceRequest 类型的 input 选择的抽象语法是 Input-Request。

17.2.2 Input-Response

ConfirmedServiceResponse 类型的 input 选择的抽象语法是 Input-Response。

17.3 Output(输出)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 output 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
  Output-Request ::= SEQUENCE {
    operatorStationName     [0] IMPLICIT Identifier,
    listOfOutputData        [1] IMPLICIT SEQUENCE OF MMSString }
  Output-Response ::= NULL
```

17.3.1 Output-Request

ConfirmedServiceRequest 类型的 output 选择的抽象语法是 Output-Request。

17.3.2 Output-Response

ConfirmedServiceResponse 类型的 output 选择的抽象语法是 Output-Response。

18 事件管理协议

18.1 引言

本章描述 MMS 事件管理功能单元所定义的服务及服务修饰符的服务专用协议元素,它们包括:

TriggerEvent (触发事件)

EventNotification （事件通告）

AcknowledgeEventNotification （确认收到事件通告）

GetAlarmSummary （获取报警总汇）

GetAlarmEnrollmentSummary(获取报警登录总汇)

18.2 TriggerEvent(触发事件)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 triggerEvent 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
TriggerEvent-Request ::= SEQUENCE {
    eventConditionName          [0] ObjectName,
    priority                    [1] IMPLICIT Priority OPTIONAL }
TriggerEvent-Response ::= NULL
```

18.2.1 TriggerEvent-Request

ConfirmedServiceRequest 类型的 triggerEvent 选择的抽象语法是 TriggerEvent-Request。

18.2.2 TriggerEvent-Response

ConfirmedServiceResponse 类型的 triggerEvent 选择的抽象语法是 TriggerEvent-Response。

18.3 EventNotification(事件通告)

UnconfirmedService 类型的 eventNotification 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

注:事件通告是一项非确认服务,因而不定义应答或错误类型。

```
EventNotification ::= SEQUENCE {
    eventEnrollmentName         [0] ObjectName,
    eventConditionName          [1] ObjectName,
    severity                    [2] IMPLICIT Severity,
    currentState                [3] IMPLICIT EC-State OPTIONAL,
    transitionTime              [4] EventTime,
    notificationLost            [6] IMPLICIT BOOLEAN DEFAULT FALSE,
    alarmAcknowledgmentRule     [7] IMPLICIT AlarmAckRule OPTIONAL,
    actionResult                [8] IMPLICIT SEQUENCE {
        eventActionName             ObjectName,
        successOrFailure            CHOICE {
            success                     [0] IMPLICIT SEQUENCE {
                confirmedServiceResponse        ConfirmedServiceResponse
IF ( csr cspi )
,               cs-Response-Detail              [79] Response-Detail OPTIONAL
                        —shall not be transmitted if value is the
                        —value of a tagged type derived from NULL
ENDIF
                },
            failure                     [1] IMPLICIT SEQUENCE {
                modifierPosition            [0] IMPLICIT Unsigned32 OPTIONAL,
                serviceError                [1] IMPLICIT ServiceError }
            }
```

```
        } OPTIONAL
    }
  CS-EventNotification ::= [0] CHOICE {
IF ( des )
    string                          [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
    index                           [1] IMPLICIT INTEGER,
ENDIF
    noEnhancement                   NULL }
```

18.3.1 EventNotification

UnconfirmedService 的 eventNotification 选择的抽象语法是 EventNotification。以下给出该类型字段的导出。

18.3.1.1 actionResult(活动结果)

如果给出 actionResult 字段,那么,它的导出遵照 5.5 的规定。如果在 EventNotification. request 原语中给出 Action Result 参数,那么,它的 successOrFailure 字段按下述原则决定。

a) 如果 EventNotification. request 原语的 action Result 参数的 Success 或 Failure 子参数等于"真",那么,SuccessOrFailure 字段将选 Success,同时, EventNotification. indication 原语(如果发送的话),action Result 参数的 Success 参数或 Failure 参数为"真"。否则,eventAction-Result 字段将选 failure,同时, EventNotification. indication 原语(如果发送的话),action Result 参数的 Success 参数或 Failure 参数为"假"。

b) 如果选 Success,那么,将利用 Success 选择的 ConfirmedServiceResponse 类型传递由 Event Action(事件活动)对象的 &ConfirmedServiceRequest 字段所请求的服务的 Result(+)参数,同时,利用 5.5 的规定。

c) 如果选 Failure,并且,失败是在执行 Event Action 对象的 &Modifier 域中所规定的一个修饰符时发生的,那么,将传递 failure 选择的 ModifierPosition(修饰符位置)选择,指明发生失败的修饰符。

d) 如果选 failure,并且,失败是在执行请求确认服务时发生的,那么,将利用 failure 选择的 serviceError 选择传递由 Event Action 对象的 &ConfimedServiceRequest 字段所请求的服务的 Result(-)参数,同时,利用 5.5 的规定。

18.3.1.1.1 ConfimedServiceResponse

Event Notification 服务的 Confimed Service Response 参数的抽象语法是 ConfimedServiceResponse 类型,再加上与 ConfimedServiceResponse 选择相对应的 CS-Response-Detail 类型选择。

18.3.1.2 Display Enhancement(显示增强)

Unconfimed-Detail(无确认-细目)的 EventNotification 选择的抽象语法是 CS-EventNotification,同时,此字段将传递 Display Enhancement 参数。

18.4 AcknowledgeEventNotification(确认收到事件通告)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 acknowledgeEventNotification 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
  AcknowledgeEventNotification-Request ::= SEQUENCE {
    eventEnrollmentName                 [0] ObjectName,
    acknowledgedState                   [2] IMPLICIT EC-State,
```

```
    timeOfAcknowledgedTransition                [3] EventTime }
  AcknowledgeEventNotification-Response ::= NULL
```

18.4.1 AcknowledgeEventNotification-Request

ConfirmedServiceRequest 类型的 acknowledgeEventNotification 选择的抽象语法是 AcknowledgeEventNotification-Request。

18.4.2 AcknowledgeEventNotification-Response

ConfirmedServiceResponse 类型的 acknowledgeEventNotification 选择的抽象语法是 AcknowledgeEventNotification-Response。

18.5 GetAlarmSummary(获取报警总汇)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 getAlarmSummary 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
  GetAlarmSummary-Request ::= SEQUENCE {
    enrollmentsOnly             [0] IMPLICIT BOOLEAN DEFAULT TRUE,
    activeAlarmsOnly            [1] IMPLICIT BOOLEAN DEFAULT TRUE,
    acknowledgementFilter       [2] IMPLICIT INTEGER {
        not-acked                       (0),
        acked                           (1),
        all                             (2)
        } (0..2) DEFAULT not-acked,
    severityFilter              [3] IMPLICIT SEQUENCE {
        mostSevere                      [0] IMPLICIT Unsigned8,
        leastSevere                     [1] IMPLICIT Unsigned8 }
                                    DEFAULT { mostSevere 0, leastSevere 127 },
    continueAfter               [5] ObjectName OPTIONAL
    }
  GetAlarmSummary-Response ::= SEQUENCE {
    listOfAlarmSummary          [0] IMPLICIT SEQUENCE OF AlarmSummary,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT FALSE }
  AlarmSummary ::= SEQUENCE {
    eventConditionName          [0] ObjectName,
    severity                    [1] IMPLICIT Unsigned8,
    currentState                [2] IMPLICIT EC-State,
    unacknowledgedState         [3] IMPLICIT INTEGER {
        none                        (0),
        active                      (1),
        idle                        (2),
        both                        (3)
        } (0..3),
IF ( csr cspi )
    displayEnhancement          [4] EN-Additional-Detail OPTIONAL,
                    —shall not be transmitted if the value is NULL
ELSE
```

```
    displayEnhancement              [4] NULL,
ENDIF
    timeOfLastTransitionToActive    [5] EventTime OPTIONAL,
    timeOfLastTransitionToIdle      [6] EventTime OPTIONAL }
  EN-Additional-Detail ::=          [0] CHOICE {
IF ( des )
    string                          [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
    index                           [1] IMPLICIT INTEGER,
ENDIF
    noEnhancement                       NULL }
```

18.5.1 GetAlarmSummary-Request

ConfirmedServiceRequest 类型的 getAlarmSummary 选择的抽象语法是 GetAlarmSummary-Request。

18.5.2 GetAlarmSummary-Response

ConfirmedServiceResponse 类型的 getAlarmSummary 选择的抽象语法是 GetalarmSummary-Response。

18.5.2.1 listOfAlarmSummary

listOfAlarmSummary 字段是 GetAlarmSummary. response 原语的 listOfAlarmSummary(报警总汇)参数,并且作为 GetAlarmSummary. Confirm 原语的 listOfAlarmSummary 参数出现。该字段包含 0 个或多个 AlarmSummary 类型的出现,而每个出现的类型按照报警总汇中的顺序,含有按报警总汇参数规定的各个总表单项值。

18.5.2.1.1 DisplayEnhancement

一个给定 AlarmSummary 的 DisplayEnhancement 域是取自 GetAlarmSummary. response 原语的报警总汇参数的相应报警总汇的显示增强参数,并且,作为 GetAlarmSummary. confirm 原语的报警总汇参数的相应报警总汇的显示增强参数出现。该域的抽象语法是 EN-Additional-Detail(EN-补充细目)类型,并且,利用 5.5 的规定。

18.6 GetAlarmEnrollmentSummary(获取报警登录总汇)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型 GetAlarmEnrollment Summary 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
  GetAlarmEnrollmentSummary-Request ::= SEQUENCE {
    enrollmentsOnly                 [0] IMPLICIT BOOLEAN DEFAULT TRUE,
    activeAlarmsOnly                [1] IMPLICIT BOOLEAN DEFAULT TRUE,
    acknowledgementFilter           [2] IMPLICIT INTEGER {
        not-acked                           (0),
        acked                               (1),
        all                                 (2)
        } (0..2) DEFAULT not-acked,
    severityFilter                  [3] IMPLICIT SEQUENCE {
        mostSevere                      [0] IMPLICIT Unsigned8,
        leastSevere                     [1] IMPLICIT Unsigned8 }
```

```
                                                DEFAULT { mostSevere 0, leastSevere 127 },
    continueAfter                       [5] ObjectName OPTIONAL
    }
  GetAlarmEnrollmentSummary-Response ::= SEQUENCE {
    listOfAlarmEnrollmentSummary        [0] IMPLICIT SEQUENCE OF
    AlarmEnrollmentSummary,
    moreFollows                         [1] IMPLICIT BOOLEAN DEFAULT FALSE }
  AlarmEnrollmentSummary ::= SEQUENCE {
    eventEnrollmentName                 [0] ObjectName,
IF ( tpy )
    clientApplication                   [2] ApplicationReference OPTIONAL,
ELSE
    clientApplication                   [2] NULL,
ENDIF
    severity                            [3] IMPLICIT Unsigned8,
    currentState                        [4] IMPLICIT EC-State,
IF ( cspi )
    displayEnhancement                  [5] EN-Additional-Detail OPTIONAL,
                -shall not be transmitted if the value is NULL
ELSE
    displayEnhancement                  [5] NULL,
ENDIF
    notificationLost                    [6] IMPLICIT BOOLEAN DEFAULT FALSE,
    alarmAcknowledgmentRule             [7] IMPLICIT AlarmAckRule,
    enrollmentState                     [8] IMPLICIT EE-State OPTIONAL,
    timeOfLastTransitionToActive        [9] EventTime OPTIONAL,
    timeActiveAcknowledged              [10] EventTime OPTIONAL,
    timeOfLastTransitionToIdle          [11] EventTime OPTIONAL,
    timeIdleAcknowledged                [12] EventTime OPTIONAL }
```

18.6.1 GetAlarmEnrollmentSummary-Request

ConfirmedServiceRequest 类型的 getAlarmEnrollmentSummary 选择的抽象语法是 GetAlarmEnrollmentSummary-Request。

18.6.2 GetAlarmEnrollmentSummary-Response

ConfirmedServiceResponse 类型的 getAlarmEnrollmentSummary 选择的抽象语法是 GetAlarmEnrollmentSummary-Response。

18.6.2.1.1 listOfAlarmEnrollmentSummary

listOfAlarmEnrollmentSummary 字段是 GetAlarmEnrollmentSummary. response 原语的报警登录总汇参数，并且，它将作为 GetAlarmEnrollmentSummary. confirm 原语的报警登录总汇参数出现。该字段含有 0 个或多个 AlarmEnrollmentSummary 类型，而每一出现的类型按照报警登录总汇提供的顺序，含有按报警登录总汇规定的多个总表单项值。

18.6.2.1.1 displayEnhancement

一个给定 AlarmEnhancementSummary 的 displayEnhancement 域取自 GetAlarmSummary. response 原语的报警登录总汇参数的相应报警登录摘要的显示增强参数，并且，它将作为 GetAlarmEn-

rollmentSummary.confirm 原语中的报警登录总汇参数中的相应报警登录总汇参数的显示增强参数出现。该字段的抽象语法是 EN-Additional-Detail 类型，并且，利用 5.5 的规定。

18.7 AttachToEventCondition(附加事件条件)

Modifier 类型的 attachToEventCondition 选择的抽象语法由 AttachToEventCondition 类型规定，该类型规定如下。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
AttachToEventCondition ::= SEQUENCE {
    eventEnrollmentName         [0] ObjectName,
    eventConditionName          [1] ObjectName,
    causingTransitions          [2] IMPLICIT Transitions,
    acceptableDelay             [3] IMPLICIT Unsigned32 OPTIONAL }
```

19 事件条件协议

19.1 引言

本章描述由 MMS 事件管理功能单元定义的服务和服务修饰符的一些服务专用协议元素。它们包括：

DefineEventCondition (定义事件条件)
DeleteEventCondition (删除事件条件)
GetEventConditionAttributes(获取事件条件属性)
ReportEventConditionStatus(报告事件条件状态)
AlterEventConditionMonitoring(变更事件条件监控)

19.2 DefineEventCondition(定义事件条件)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 defineEventCondition 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
DefineEventCondition-Request ::= SEQUENCE {
    eventConditionName              [0] ObjectName,
    class                           [1] IMPLICIT EC-Class,
    priority                        [2] IMPLICIT Priority DEFAULT normalPriority,
    severity                        [3] IMPLICIT Unsigned8 DEFAULT
  normalSeverity,
    alarmSummaryReports             [4] IMPLICIT BOOLEAN OPTIONAL,
    monitoredVariable               [6] VariableSpecification OPTIONAL,
    evaluationInterval              [7] IMPLICIT Unsigned32 OPTIONAL }
  DefineEventCondition-Response ::= NULL
  CS-DefineEventCondition-Request ::= [0] CHOICE {
IF ( des )
    string                          [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
    index                           [1] IMPLICIT INTEGER,
ENDIF
    noEnhancement                   NULL }
```

19.2.1 DefineEventCondition-Request

ConfirmedServiceRequest 类型的 defineEventCondition 选择的抽象语法是 DefineEventCondition-Request。

19.2.2 DefineEventCondition-Response

ConfirmedServiceResponse 类型的 defineEventCondition 选择的抽象语法是 DefineEventCondition-Response。

19.2.3 CS-DefineEventCondition-Request

Request-Detain 类型的 defineEventCondition 选择的抽象语法是 CS-DefineEventCondition-Request，并且，当发送时，该字段将传递显示增强参数的值。

19.3 DeleteEventCondition(删除事件条件)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 deleteEventCondition 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
DeleteEventCondition-Request ::= CHOICE {
    specific                    [0] IMPLICIT SEQUENCE OF ObjectName,
    aa-specific                 [1] IMPLICIT NULL,
    domain                      [2] IMPLICIT Identifier,
    vmd                         [3] IMPLICIT NULL }
DeleteEventCondition-Response ::= Unsigned32 —Candidates Not Deleted
```

19.3.1 DeleteEventCondition-Request

ConfirmedServiceRequest 类型的 deleteEventCondition 选择的抽象语法是 DeleteEventCondition-Request。该选择的值按下述说明确定。

如果 DeleteEventCondition. request 服务原语的 Scope Of Delete(删除范围)参数之值等于 SPECIFIC，那么，DeleteEventCondition-Request 应包含 specific 选择，这个选择所包含的 Event Condition Name(事件条件名)参数的值应该取自 DeleteEventCondition. request 服务原语。

如果 DeleteEventCondition. request 服务原语的 Scope Of Delete 参数之值等于 AA-SPECIFIC，那么，DeleteEventCondition-Request 应包含 aa-specific 选择。

如果 DeleteEventCondition. Request 服务原语的删除范围参数之值等于 DOMAIN，那么，DeleteEventCondition-Request 应包含 domain 选择，该选择所包含的 Domain Name(域名)参数的值应该取自 DeleteEventCondition. request 服务原语。

如果 DeleteEventCondition. Request 服务原语的 Scope Of Delete 参数之值等于 VMD，那么，DeleteEventCondition-Request 应包含 vmd 选择。

19.3.2 DeleteEventCondition-Response

ConfirmedServiceResponse 类型的 deleteEventCondition 选择的抽象语法是 DeleteEventCondition-Response。它是取自指明为 Result(+)的 DeleteEventCondition. response 原语的 Candidates Not Deleted(非删除候选者)参数，并且，作为指表明 Result(+)的 DeleteEventCondition. confirm 原语的非删除候选者参数出现。

19.4 GetEventConditionAttributes(获取事件条件属性)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 getEventConditionAttributes 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
GetEventConditionAttributes-Request ::= ObjectName —Event Condition Name
GetEventConditionAttributes-Response ::= SEQUENCE {
```

```
    mmsDeletable                    [0] IMPLICIT BOOLEAN DEFAULT FALSE,
    class                           [1] IMPLICIT EC-Class,
    priority                        [2] IMPLICIT Priority DEFAULT normalPriority,
    severity                        [3] IMPLICIT Unsigned8 DEFAULT normalSeverity,
    alarmSummaryReports             [4] IMPLICIT BOOLEAN DEFAULT FALSE,
    monitoredVariable               [6] CHOICE {
        variableReference                   [0] VariableSpecification,
        undefined                           [1] IMPLICIT NULL } OPTIONAL,
    evaluationInterval              [7] IMPLICIT Unsigned32 OPTIONAL
IF (aco),
    accessControlList               [8] IMPLICIT Identifier OPTIONAL
ENDIF
                    —Shall not appear in minor version one or two
    }
  CS-GetEventConditionAttributes-Response ::= SEQUENCE {
    groupPriorityOverride           [0] CHOICE {
        priority                        [0] IMPLICIT Priority,
        undefined                       [1] IMPLICIT NULL } OPTIONAL,
    listOfReferencingECL            [1] IMPLICIT SEQUENCE OF ObjectName
OPTIONAL,
    displayEnhancement              [2] CHOICE {
IF ( des )
    string                          [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
    index                           [1] IMPLICIT INTEGER,
ENDIF
    noEnhancement                   [2] IMPLICIT NULL }
    }
```

19.4.1 GetEventConditionAttributes-Request

ConfirmedServiceRequest 类型的 getEventConditionAttributes 选择的抽象语法是 GetEventConditionAttributes-Request，它是取自 GetEventConditionAttributes. request 原语的 Event Condition Name(事件条件名)参数，并且，它将作为参数 GetEventConditionAttributes. indication 原语(如果发送的话)的 EventConditionName 出现。

19.4.2 GetEventConditionAttributes-Response

ConfirmedServiceRequest 类型的 getEventConditionAttributes 选择的抽象语法是 GetEventConditionAttributes-Response。

19.4.2.1 alarmSummaryReport

如果 GetEventConditionAttributes. response 原语的 Class 参数之值不是 MONITORED，那么，提供的值为“假”。否则，该值等于 GetEventConditionAttributes. response 服务原语的 Alarm Summary Reports 参数的值。

19.4.2.2 monitoredVariable(监控变量)

如果 GetEventConditionAttributes. response 服务原语的 monitored variable 参数之值为 UNDE-

FINED,那么,monitoredVariable 域选 undefined。否则,选 VariableReference。

19.4.2.3 accessControlList

当且仅当 acoCBB 商定后,accessControlList 参数才出现。

19.4.3 CS—GetEventConditionAttributes-Response

Response-Detail 类型的 getEventConditionAttributes 选择的抽象语法是 CS—GetEventCondition-Attributes 类型。

19.5 ReportEventConditionStatus(报告事件条件状态)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 ReportEvent ConditionStatus 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
ReportEventConditionStatus-Request ::= ObjectName —Event Condition Name
ReportEventConditionStatus-Response ::= SEQUENCE {
    currentState                    [0] IMPLICIT EC-State,
    numberOfEventEnrollments        [1] IMPLICIT Unsigned32,
    enabled                         [2] IMPLICIT BOOLEAN OPTIONAL,
    timeOfLastTransitionToActive    [3] EventTime OPTIONAL,
    timeOfLastTransitionToIdle      [4] EventTime OPTIONAL }
```

19.5.1 ReportEventConditionStatus-Request

ConfirmedServiceRequest 类型的 reportEventConditionStatus 选择的抽象语法是 Report Event-ConditionStatus-Request。

19.5.2 ReportEventConditionStatus-Response

ConfirmedServiceResponse 类型的 reportEventConditionStatus 选择的抽象语法是 Report Event-ConditionStatus-Response。

19.6 AlterEventConditionMonitoring(变更事件条件监控)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 alterEvent ConditionMonitoring 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
AlterEventConditionMonitoring-Request ::= SEQUENCE {
    eventConditionName              [0] ObjectName,
    enabled                         [1] IMPLICIT BOOLEAN OPTIONAL,
    priority                        [2] IMPLICIT Priority OPTIONAL,
    alarmSummaryReports             [3] IMPLICIT BOOLEAN OPTIONAL
IF ( cei )
,   evaluationInterval              [4] IMPLICIT Unsigned32 OPTIONAL
ENDIF
        —At least one of enabled, priority, alarmSummaryReports, or
        —evaluationInterval shall be present.
    }
AlterEventConditionMonitoring-Response ::= NULL
CS-AlterEventConditionMonitoring-Request ::= SEQUENCE {
    changeDisplay                   CHOICE {
IF ( des )
        string                              [0] IMPLICIT VisibleString,
```

```
ENDIF
IF ( dei )
        index                                   [1] IMPLICIT INTEGER,
ENDIF
        noEnhancement                           [2] NULL } OPTIONAL
    }
```

19.6.1 AlterEventConditionMonitoring-Request

ConfirmedServiceRequest 类型的 alterEventConditionMonitoring 选择的抽象语法是 AlterEventConditionMonitoring-Request。

19.6.2 AlterEventConditionMonitoring-Response

ConfirmedServiceResponse 类型的 alterEventConditionMonitoring 选择的抽象语法是 AlterEventConditionMonitoring-Response。

19.6.3 CS-AlterEventConditionMonitoring-Request

Request-Detail 的 alterEventConditionMonitoring 选择的抽象语法是 CS-AlterEventConditionMonitoring-Response，该字段传递显示增强参数。并且，当且仅当 AlterEventConditionMonitoring. indication 服务原语中给出了显示增强参数时，才包含此字段。

20 事件活动协议

20.1 引言

本章描述由 MMS 事件管理功能单元定义的服务和服务修饰符的一些服务专用协议元素。它们包括：

DefineEventAction （定义事件活动）

DeleteEventAction （删除事件活动）

GetEventActionAttributes （获取事件活动属性）

ReportEventActionStatus （报告事件活动状态）

20.2 DefineEventAction(定义事件活动)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 defineEventAction 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
DefineEventAction-Request ::= SEQUENCE {
    eventActionName             [0] ObjectName,
    listOfModifier              [1] IMPLICIT SEQUENCE OF Modifier OPTIONAL,
    confirmedServiceRequest     [2] ConfirmedServiceRequest
IF ( csr cspi )
,   cs-extension                [79] Request-Detail OPTIONAL
        --shall not be transmitted if value is the value
        --of a tagged type derived from NULL
ENDIF
    }
DefineEventAction-Response ::= NULL
```

20.2.1 DefineEventAction-Request

ConfirmedServiceRequest 类型的 defineEventAction 选择的抽象语法是 DefineEventAction-Request。

20.2.1.1 ConfirmedServiceRequest

定义事件活动服务的 Confirmed Service Request 参数的抽象语法是 ConfirmedServiceRequest 类型，并后随有与 ConfirmedServiceRequest 类型选择相对应的 CS-Request-Detail 类型。

20.2.2 DefineEventAction-Response

ConfirmedServiceResponse 类型的 defineEventAction 选择的抽象语法是 DefineEventAction-Response。

20.3 DeleteEventAction(删除事件活动)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 deleteEventAction 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
DeleteEventAction-Request ::= CHOICE {
    specific                [0] IMPLICIT SEQUENCE OF ObjectName,
    aa-specific             [1] IMPLICIT NULL,
    domain                  [3] IMPLICIT Identifier,
    vmd                     [4] IMPLICIT NULL }
DeleteEventAction-Response ::= Unsigned32 —Candidates Not Deleted
```

20.3.1 DeleteEventAction-Request

ConfirmedServiceRequest 类型的 DeleteEventAction 选择的抽象语法是 DefineEventAction-Request。该选择的值按以下描述确定。

如果 DeleteEventAction. request 服务原语的 Scope Of Delete 参数之值等于 SPECIFIC，那么，DeleteEventAction-Request 应包含 specific 选择，这个选择包含 DeleteEventAction. request 服务原语的 Event Action Names 参数的值。

如果 DeleteEventAction. request 服务原语的 Scope Of Delete 参数之值等于 AA-SPECIFIC，那么，DeleteEventAction-Request 将包含 aa-specific 选择。

如果 DeleteEventAction. request 服务原语的 Scope Of Delete 参数之值等于 DOMAIN，那么，DeleteEventAction-Request 包含 domain 选择，这个选择包含 DeleteEventAction. request 服务原语的(Domain Name)参数之值。

如果 DeleteEventAction. request 服务原语的 Scope Of Delete 参数之值等于 VMD，那么，DeleteEventAction-Request 将包含 vmd 选择。

20.3.2 DeleteEventAction-Response

ConfirmedServiceResponse 类型的 deleteEventAction 选择的抽象语法是 DefineEventAction-Response。它取自指明为 Result(+)的 DeleteEventAction. response 原语的 CandidatesNotDeleted 参数，并且，作为在指明 Result(+)的 DeleteEventAction. confirm 原语的 CandidatesNotDeleted 参数出现。

20.4 GetEventActionAttributes(获取事件活动属性)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 getEventAction Attributes 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
GetEventActionAttributes-Request ::= ObjectName —EventActionName
GetEventActionAttributes-Response ::= SEQUENCE {
    mmsDeletable                [0] IMPLICIT BOOLEAN DEFAULT FALSE,
    listOfModifier              [1] IMPLICIT SEQUENCE OF Modifier,
    confirmedServiceRequest     [2] ConfirmedServiceRequest
IF ( csr cspi )
```

, cs-extension [79] Request-Detail OPTIONAL

—shall not be transmitted if value is the value

—of a tagged type derived from NULL

ENDIF

IF (aco)

, accessControlList [3] IMPLICIT Identifier OPTIONAL

ENDIF

—Shall not appear in minor version one or two

}

20.4.1 GetEventActionAttributes-Request

ConfirmedServiceRequest 类型的 GetEventActionAttributes 选择的抽象语法是 GetEventAction-Attributes-Request。

20.4.2 GetEventActionAttributes-Response

ConfirmedServiceResponse 类型的 GetEventActionAttributes 选择的抽象语法是 GetEventActionAttributes-Response。

20.4.2.1 ConfirmedServiceRequest

GetEventActionAttributes(获取事件活动属性)服务的 ConfirmedServiceRequest 确认服务请求参数的抽象语法是 ConfirmedServiceRequest 类型,并后随有与 ConfirmedServiceRequest 类型选择相对应的 Request-Detail 类型。

20.4.2.2 accessControlList

当且仅当 aco CBB 商定后,accessControlList 参数才出现。

20.5 ReportEventActionStatus(报告事件活动状态)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 reportEventAction Status 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

ReportEventActionStatus-Request ::= ObjectName—Event Action Name

ReportEventActionStatus-Response ::= Unsigned32—Number of Event Enrollments

20.5.1 ReportEventActionStatus-Request

ConfirmedServiceRequest 类型的 reportEventActionStatus 选择的抽象语法是 ReportEvent ActionStatus-Request。

20.5.2 ReportEventActionStatus-Response

ConfirmedServiceResponse 类型的 reportEventActionStatus 选择的抽象语法是 Report EventActionStatus-Response。它由一个 Result(+)指明,这个 Result(+)包含 ReportEventActionStatus. response 服务原语中的事件登录个数参数,同时,它还作为 ReportEventActionStatus. confirm 服务原语中的 Result(+)参数出现,而这个 Result(+)包含事件登录个数参数。

21 事件登录协议

21.1 引言

本章描述由 MMS 事件管理功能单元定义的服务及服务修饰符的一些服务专用协议元素。它们包括:

DefineEventEnrollment (定义事件登录)

DeleteEventEnrollment (删除事件登录)

GetEventEnrollmentAttributes (获取事件登录属性)

ReportEventEnrollmentStatus (报告事件登录状态)

AlterEventEnrollment （变更事件登录）

21.2 DefineEventEnrollment(定义事件登录)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 defineEventEnrollment 选择的抽象语法规定如下，其描述在后几节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
  DefineEventEnrollment-Request ::= SEQUENCE {
    eventEnrollmentName             [0] ObjectName,
    eventConditionName              [1] ObjectName,
    eventConditionTransitions       [2] IMPLICIT Transitions,
    alarmAcknowledgmentRule         [3] IMPLICIT AlarmAckRule,
    eventActionName                 [4] ObjectName OPTIONAL
IF ( tpy )
,   clientApplication               [5] ApplicationReference OPTIONAL
ENDIF
    }
  DefineEventEnrollment-Response ::= NULL
  DefineEventEnrollment-Error ::= ObjectName
  CS-DefineEventEnrollment-Request ::= [0] CHOICE {
IF ( des )
    string                          [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
    index                           [1] IMPLICIT INTEGER,
ENDIF
    noEnhancement                   NULL }
```

21.2.1 DefineEventEnrollment-Request

ConfirmedServiceRequest 类型的 defineEventEnrollment 选择的抽象语法是 Define EventEnrollment-Request。

21.2.2 DefineEventEnrollment-Response

ConfirmedServiceResponse 类型的 defineEventEnrollment 选择的抽象语法是 Define EventEnrollment-Response。

21.2.3 DefineEventEnrollment-Error

ConfirmedServiceErrort 类型的 serviceSpeificInformation 选择的 defineEventEnrollment 选择的抽象语法是 DefineEventEnrollment-Error，它是 DefineEventEnrollment. response 原语的 Result(－)参数的 Object Not Defined(未定义对象)参数。它将作为 DefineEventEnrollment. confirm 原语(如果发送的话)的 Result(－)的 Object Not Define 参数出现。

21.2.4 CS-DefineEventEnrollment-Request

Request-Detail 的 defineEventEnrollment 选择的抽象语法是 CS-DefineEventEnrollment-Request，同时，它还传递(Display Enhancement)参数的值。

21.3 DeleteEventEnrollment(删除事件登录)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 deleteEvent Enrollment 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
DeleteEventEnrollment-Request ::= CHOICE {
    specific                    [0] IMPLICIT SEQUENCE OF ObjectName,
    ec                          [1] ObjectName,
    ea                          [2] ObjectName }
DeleteEventEnrollment-Response ::= Unsigned32 —Candidates Not Deleted
```

21.3.1 DeleteEventEnrollment-Request

ConfirmedServiceRequest 类型的 deleteEventEnrollment 选择的抽象语法是 Delete EventEnrollment-Request。该选择的值按以下说明确定。

如果 Scope Of Delete 参数指明为 List Of Event Enrollment Names(事件登录名称表)参数,那么,DeleteEventEnrollment-Request 选 specific,该选择包含有来自 DeleteEventEnrollment. request 服务原语的 List Of Event Enrollment Names 参数的值。

如果删除范围参数指明为 Event Condition Name(事件条件名)参数,那么,DeleteEventEnrollment-Request 选 ec,该选择包含有来自 DeleteEventEnrollment. request 服务原语的 Event Condition Name 参数的值。

如果删除范围参数指明为 Event Action Name(事件活动名)参数,那么,DeleteEventEnrollment-Request 选 ea 选择,该选择包含有来自 DeleteEventEnrollment. request 服务原语的 Event Action Name 参数的值。

21.3.2 DeleteEventEnrollment-Response

ConfirmedServiceResponse 类型的 deleteEventEnrollment 选择的抽象语法是 Delete EventEnrollment-Response。它是取自指明有 Result(+)DeleteEventEnrollment. response 服务原语的 Candidates Not Deleted 参数。同时,它将作为在指明有 Result(+)的 DeleteEventEnrollment. confirm 服务原语的 Candidates Not Deleted 参数出现。

21.4 GetEventEnrollmentAttributes(获取事件登录属性)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 GetEventEnrollmentAttributes 选择的抽象语法规定如下,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
GetEventEnrollmentAttributes-Request ::= SEQUENCE {
    scopeOfRequest              [0] IMPLICIT INTEGER {
        specific                    (0),
        client                      (1),
        ec                          (2),
        ea                          (3) } (0..3) DEFAULT client,
    eventEnrollmentNames        [1] IMPLICIT SEQUENCE OF ObjectName
OPTIONAL,
IF ( tpy )
    clientApplication           [2] ApplicationReference OPTIONAL,
ELSE
    clientApplication           [2] NULL,
ENDIF
    eventConditionName          [3] ObjectName OPTIONAL,
    eventActionName             [4] ObjectName OPTIONAL,
    continueAfter               [5] ObjectName OPTIONAL }
GetEventEnrollmentAttributes-Response ::= SEQUENCE {
```

```
    listOfEEAttributes              [0] IMPLICIT SEQUENCE OF EEAttributes,
    moreFollows                     [1] IMPLICIT BOOLEAN DEFAULT FALSE }
  EEAttributes ::= SEQUENCE {
    eventEnrollmentName             [0] ObjectName,
    eventConditionName              [1] CHOICE {
        eventCondition                      [0] ObjectName,
        undefined                           [1] IMPLICIT NULL },
    eventActionName                 [2] CHOICE {
        eventAction                         [0] ObjectName,
        undefined                           [1] IMPLICIT NULL } OPTIONAL,
IF ( tpy )
    clientApplication               [3] ApplicationReference OPTIONAL,
ELSE
    clientApplication               [3] NULL,
ENDIF
    mmsDeletable                    [4] IMPLICIT BOOLEAN DEFAULT FALSE,
    enrollmentClass                 [5] IMPLICIT EE-Class,
    duration                        [6] IMPLICIT EE-Duration DEFAULT current,
    invokeID                        [7] IMPLICIT Unsigned32 OPTIONAL,
    remainingAcceptableDelay        [8] IMPLICIT Unsigned32 OPTIONAL
IF ( csr cspi )
,   displayEnhancement              [9] CHOICE {
IF ( des )
        string                              [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )

        index                               [1] IMPLICIT INTEGER,
ENDIF
        noEnhancement                   NULL }
                —shall not be transmitted if the value is NULL
ELSE
    displayEnhancement              [9] NULL
IF ( aco )
,   accessControlList               [11] IMPLICIT Identifier
                —shall not appear in minor version one or two
ENDIF
    }
```

21.4.1 GetEventEnrollmentAttributes-Request

ConfirmedServiceRequest 类型的 getEventEnrollmentAttributes 选择的抽象语法是 Get EventEnrollment Attributes-Request。

continueAfter(后续)字段是 GetEventEnrollmentAttributes. request 原语的 Continue After 参数的 Enrollment ID,并且,它将作为 GetEventEnrollmentAttributes. indication(如果发送的话)的 contin-

ue After 参数的 Enrollment ID 出现。

如果在请求原语中不存在 Continue After 参数，那么，Continue After 字段也不在 ConfirmedServiceRequest 中，同时，在指示原语（如果发送的话）中也不存在 Continue After 参数。

21.4.1.1 ScopeOfRequest（请求范围）

ScopeOfRequest 字段指明对请求原语中的 Scope Of Request 参数所作的选择。如果在请求原语中选 List of Event Enrollment Names，那么，ScopeOfRequest 字段的 specific。如果在请求原语中选 Client Application（客户应用），那么，ScopeOfRequest 字段应选 client。如果，在请求原语中选 Event Condition Names，那么，ScopeOfRequest 字段应选 ec。如果，在请求原语中选 EventAction Names，那么，ScopeOfRequest 字段应选 ea。

21.4.1.2 eventEnrollmentNames（事件登录名）

如果 Scope Of Request 参数选定为 List Of Event Enrollment Name，那么 eventEnrollmentNames 字段包含这个参数的值，否则，这个字段就不出现。

21.4.1.3 clientApplication

如果 Scope Of Request 参数选定为 Clint Application，同时，这个参数未指定这个服务请求的 MMS 客户，那么，该域包含这个 Scope Of Request 参数的值，否则，它就不出现。

如果 Scope Of Request 参数选定为 Event Condition Name 或 Event Action Name，并且，指定 Clint Application 作为该参数的任选项，那么，该字段将包含这个 Scope Of Request 参数的值。否则，这个域就不出现。

21.4.1.4 eventConditionName（事件条件名）

如果 Scope of Request 参数选定为 Event Condition Name，那么，该字段包含这个参数的值。否则，该字段将不出现。

21.4.1.5 eventActionName（事件活动名）

如果 Scope of Request 参数选定为 Event Condition Name，那么，该字段包含这个参数的值。否则，该字段不出现。

21.4.2 GetEventEnrollmentAttributes-Response

ConfirmedServiceResponse 类型的 getEventEnrollmentAttributes 选择的抽象语法是 GetEventEnrollmentAttributes-Response。

21.4.2.1 listOfEEAttributes（EE 属性表）

listOfEEAttributes 字段是 GetEventEnrollmentAttributes. response 原语的 EE 属性表参数，同时，作为 GetEventEnrollmentAttributes. confirm 原语中的 EE 属性表参数出现。该字段包含 0 个或多个 EEAttribute 类型的实例，而每个实例按表中的顺序，包含 EE 属性表的一个 EEAttributes 参数的值。为了导出 List Of EEAttribute 参数的对应元素，可将 5.5 的规定应用于 List Of EEAttribute 参数的每个 EEAttribute 参数的实例。

21.4.2.1.1 eventConditionName（事件条件名）

如果 GetEventEnrollmentAttributes. response 服务原语的 eventConditionName 参数的值为 UNDEFINED，那么，eventConditionName 域选 undefined。否则，选 eventCondition。

21.4.2.1.2 eventActionName（事件活动名）

如果 GetEventEnrollmentAttributes. response 服务原语的 EventActionName 参数之值为 UNDEFINED，那么，eventActionName 域选 undefined。否则，选 eventAction。

21.4.2.1.3 accessControlList

当且仅当 aco CBB 商定后，accessControlList 参数才出现。

21.5 ReportEventEnrollmentStatus（报告事件登录状态）

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 reportEvent EnrollmentStatus

选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在5.5中描述。

```
ReportEventEnrollmentStatus-Request ::= ObjectName —Event Enrollment Name
ReportEventEnrollmentStatus-Response ::= SEQUENCE {
    eventConditionTransitions          [0] IMPLICIT Transitions,
    notificationLost                   [1] IMPLICIT BOOLEAN DEFAULT FALSE,
    duration                           [2] IMPLICIT EE-Duration,
    alarmAcknowledgmentRule            [3] IMPLICIT AlarmAckRule OPTIONAL,
    currentState                       [4] IMPLICIT EE-State }
```

21.5.1 ReportEventEnrollmentStatus-Request

ConfirmedServiceRequest类型的reportEventEnrollmentStatus选择的抽象语法是ReportEventEnrollmentStatus-Request。

21.5.2 ReportEventEnrollmentStatus-Response

ConfirmedServiceResponse类型的reportEventEnrollmentStatus选择的抽象语法是ReportEventEnrollmentStatus-Response。

21.6 AlterEventEnrollment(变更事件登录)

ConfirmedServiceRequest类型和ConfirmedServiceResponse类型的alterEventEnrollment选择的抽象语法规定如下，其描述在后几节中给出。对于本章中未提供显式导出的那些参数，它们的导出在5.5中描述。

```
AlterEventEnrollment-Request ::= SEQUENCE {
    eventEnrollmentName            [0] ObjectName,
    eventConditionTransitions      [1] IMPLICIT Transitions OPTIONAL,
    alarmAcknowledgmentRule        [2] IMPLICIT AlarmAckRule OPTIONAL }
AlterEventEnrollment-Response ::= SEQUENCE {
    currentState                    [0] CHOICE {
        state                           [0] IMPLICIT EE-State,
        undefined                       [1] IMPLICIT NULL },
    transitionTime                  [1] EventTime }
CS-AlterEventEnrollment-Request ::= SEQUENCE {
    changeDisplay                CHOICE {
IF ( des )
        string                      [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
        index                       [1] IMPLICIT INTEGER,
ENDIF
        noEnhancement               [2] NULL } OPTIONAL }
```

21.6.1 AlterEventEnrollment-Request

ConfirmedServiceRequest类型的alterEventEnrollment选择的抽象语法是AlterEvent Enrollment-Request。

21.6.2 AlterEventEnrollment-Response

ConfirmedServiceResponse类型的alterEventEnrollment选择的抽象语法是AlterEvent Enrollment-Response。

21.6.2.1 CurrentStatus(当前状态)

如果 AlterEventEnrollment. confirm 服务原语的 Current Status 参数的值等于 UNDEFINED,那么,currentStatus 字段应选 undefined。否则,选 state。

21.6.3 CS-AlterEventEnrollment-Request

Request-Detail 的 alterEventEnrollment 选择的抽象语法是 CS-AlterEventEnrollment-Request,同时,传递显示增强参数的值。如果显示增强参数出现在请求原语中,那么,changeDisplay(修改显示)字段将出现在具有相应类型选择的 CS-AlterEventEnrollment-Request 字段中。如果显示增强参数未出现在请求原语中,那么,changeDisplay 域就不出现在 CS-AlterEventEnrollment-Request 中同时,该字段将由一个空的 SEQUENCE 组成。

21.7 Supporting Productions(支持产生式)

下面给出支持事件管理协议的定义的抽象语法。

21.7.1 EE-state

EE-state 类型用于传递事件条件对象和事件登录对象的组合信息的参数。

```
EE-State ::= INTEGER {
    disabled            (0),
    idle                (1),
    active              (2),
    activeNoAckA        (3),
    idleNoAckI          (4),
    idleNoAckA          (5),
    idleAcked           (6),
    activeAcked         (7),
    undefined           (8)
    }
```

22 事件条件表协议

22.1 引言

本章描述由 MMS 事件条件表功能单元定义的服务及服务修饰符的一些服务专用协议元素。它们包括:

DefineEventConditionList (定义事件条件表)
DeleteEventConditionList(删除事件条件表)
AddEventConditionListReference(增加事件条件表引用)
RemoveEventConditionListReference(取消事件条件表引用)
GetEventConditionListAttributes (获取事件条件表属性)
ReportEventConditionListStatus (报告事件条件表状态)
AlterEventConditionListMonitoring (变更事件条件表监控)

22.2 DefineEventConditionList (定义事件条件表)协议

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 defineECL 选择的抽象语法规定如下,其描述在后几节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
DefineEventConditionList-Request ::= SEQUENCE {
    eventConditionListName              [0] ObjectName,
    listOfEventConditionName            [1] IMPLICIT SEQUENCE OF
```

ObjectName
IF (recl)
, listOfEventConditionListName [2] IMPLICIT SEQUENCE OF
ObjectName OPTIONAL
—shall appear if an only if recl has been negotiated.
ENDIF
}

DefineEventConditionList-Response ::= NULL
DefineEventConditionList-Error ::= ObjectName

22.2.1 DefineEventConditionList-Request

ConfirmedServiceRequest 类型的 defineECL 选择的抽象语法是 DefineEventConditionList-Request。

22.2.2 DefineEventConditionList-Response

ConfirmedServiceResponse 类型的 defineECL 选择的抽象语法是 DefineEventCondition List-Response。

22.2.3 DefineEventConditionList-Error

AdditionalService-Error 的 defineECL 选择的抽象语法是 DefineEventConditionList-Error，它是 DefineEventConditionList. response 原语的 Result(－)参数的 Error 参数中的 Object，并作为 DefineEventConditionList. confirm 原语(如果发送的话)的 Error 参数中的 Object 出现。

22.3 DeleteEventConditionList(删除事件条件表)协议

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 deleteECL 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

DeleteEventConditionList-Request ::= ObjectName—EventConditionListName
DeleteEventConditionList-Response ::= NULL

22.3.1 DeleteEventConditionList-Request

ConfirmedServiceRequest 类型的 deleteECL 选择的抽象语法是 DeleteEventConditionList-Request。

22.3.2 DeleteEventConditionList-Response

ConfirmedServiceResponse 类型的 deleteECL 选择的抽象语法是 DeleteEventConditionList-Response。

22.4 AddEventConditionListReference(添加事件条件表引用)协议

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 AddECLReference 选择的抽象语法规定如下，其描述在后几节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

AddEventConditionListReference-Request ::= SEQUENCE {
eventConditionListName [0] ObjectName,
listOfEventConditionName [1] IMPLICIT SEQUENCE OF ObjectName
IF (recl)
, listOfEventConditionListName [2] IMPLICIT SEQUENCE OF ObjectName OPTIONAL
—shall appear if an only if recl has been negotiated.
ENDIF
}

```
AddEventConditionListReference-Response ::= NULL
AddEventConditionListReference-Error ::= ObjectName
```

22.4.1 AddEventConditionListReference-Request

ConfirmedServiceRequest 类型的 addECLReference 选择的抽象语法是 AddEventConditionListReference-Request。

22.4.2 AddEventConditionListReference-Response

ConfirmedServiceResponse 类型的 addEventConditionListReference 选择的抽象语法是 AddEventConditionListReference-Response。

22.4.3 AddEventConditionListReference-Error

AdditionalService-Error 类型的 addECLReference 选择的抽象语法是 AddEventConditionListReference-Error，它是 AddEventConditionListReference. response 原语的 Result(－)参数的 Error 参数中的 Object，并且，作为 AddEventConditionListReference. confirm 原语(如果发送的话)的 Error 参数中的 Object 出现。

22.5 RemoveEventConditionListReference(取消事件条件表引用)协议

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 removeECLReference 选择的抽象语法规定如下，其描述在后几节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
RemoveEventConditionListReference-Request ::= SEQUENCE {
    eventConditionListName              [0] ObjectName,
    listOfEventConditionName            [1] IMPLICIT SEQUENCE OF ObjectName
IF ( recl )
,   listOfEventConditionListName        [2] IMPLICIT SEQUENCE OF ObjectName
                    —shall appear if an only if recl has been negotiated.
ENDIF
    }
RemoveEventConditionListReference-Response ::= NULL
RemoveEventConditionListReference-Error ::= CHOICE {
    eventCondition                      [0] ObjectName,
    eventConditionList                  [1] ObjectName }
```

22.5.1 RemoveEventConditionListReference-Request

ConfirmedServiceRequest 类型的 removeECLReference 选择的抽象语法是 RemoveEventConditionListReference-Request。

22.5.2 RemoveEventConditionListReference-Response

Confirmed ServiceResponse 类型的 removeECLReference 选择的抽象语法是 RemoveEventConditionListReference-Response。

22.5.3 RemoveEventConditionListReference-Error

AdditionalService-Error 类型的 removeECLReference 选择的抽象语法是 RemoveEventConditionListReference-Error，它是 RemoveEventConditionListReference. response 原语的 Result(－)参数的 Error 参数中的 Object，并且，作为 RemoveEventConditionListReference. confirm 原语(如果发送的话)的 Error 参数中的 Object 出现。

22.6 GetEventConditionListAttributes(获取事件条件表属性)协议

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 getECLAttributes 选择的抽

象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在5.5中描述。

```
GetEventConditionListAttributes-Request ::= ObjectName—eventConditionListName
GetEventConditionListAttributes-Response ::= SEQUENCE {
    listOfEventConditionName          [1] IMPLICIT SEQUENCE OF ObjectName
IF ( recl )
,   listOfEventConditionListName      [2] IMPLICIT SEQUENCE OF ObjectName
OPTIONAL
                —shall appear if an only if recl has been negotiated.
ENDIF
    }
```

22.6.1 GetEventConditionListAttributes-Request

ConfirmedServiceRequest 类型的 getECLAttributes 选择的抽象语法是 GetEventConditionListAttributes-Request。

22.6.2 GetEventConditionListAttributes-Response

ConfirmedServiceResponse 类型的 getECLAttributes 选择的抽象语法是 GetEventConditionListAttributes-Response。

22.7 ReportEventConditionListStatus(报告事件条件表状态)协议

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 reportECLStatus 选择的抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在5.5中描述。

```
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 }
```

22.7.1 ReportEventConditionListStatus-Request

ConfirmedServiceRequest 类型的 reportECLStatus 选择的抽象语法是 ReportEventConditionListStatus-Request。

22.7.2 ReportEventConditionListStatus-Response

ConfirmedServiceResponse 类型的 reportECLStatus 选择的抽象语法是 ReportEventConditionListStatus-Response。

22.8 AlterEventConditionListMonitoring(变更事件条件表监控)协议

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 alterECLMonitoring 选择的

抽象语法规定如下，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在5.5中描述。

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

22.8.1 AlterEventConditionListMonitoring-Request

ConfirmedServiceRequest类型的alterECLMonitoring选择的抽象语法是AlterEventCondition-ListMonitoring-Request。

22.8.2 AlterEventConditionListMonitoring-Response

ConfirmedServiceResponse类型的alterECLMonitoring选择的抽象语法是AlterEventCondition-ListMonitoring-Response。

23 日志管理协议

23.1 引言

本章描述MMS服务定义的日志管理一章中所定义的服务和服务专用协议元素。它们包括下列服务：

ReadJournal （读日志）
WriteJournal （写日志）
InitializeJournal （初始化日志）
ReportJournalStatus （报告日志状态）
CreateJournal （建立日志）
DeleteJournal （删除日志）

23.2 ReadJournal(读日志)

ConfirmedServiceRequest类型和ConfirmedServiceResponse类型的readJournal选择的抽象语法分别由ReadJournal-Request类型及ReadJournal-Response类型规定。以下是这两种类型的规定，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在5.5中描述。

```
ReadJournal-Request ::= SEQUENCE {
    journalName             [0] ObjectName,
    rangeStartSpecification  [1] CHOICE {
        startingTime            [0] IMPLICIT TimeOfDay,
        startingEntry           [1] IMPLICIT OCTET STRING } OPTIONAL,
    rangeStopSpecification   [2] CHOICE {
        endingTime              [0] IMPLICIT TimeOfDay,
        numberOfEntries         [1] IMPLICIT Integer32 } OPTIONAL,
    listOfVariables         [4] IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
    entryToStartAfter       [5] IMPLICIT SEQUENCE {
```

```
        timeSpecification           [0] IMPLICIT TimeOfDay,
        entrySpecification          [1] IMPLICIT OCTET STRING } OPTIONAL
    }
  ReadJournal-Response ::= SEQUENCE {
    listOfJournalEntry      [0] IMPLICIT SEQUENCE OF JournalEntry,
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT FALSE }
  JournalEntry ::= SEQUENCE {
    entryIdentifier         [0] IMPLICIT OCTET STRING,
    originatingApplication  [1] ApplicationReference,
    entryContent            [2] IMPLICIT EntryContent }
```

23.2.1 ReadJournal-Request

ConfirmedServiceRequest 类型的 readJournal 选择的抽象语法是 ReadJournal-Request。

23.2.2 ReadJournal-Response

ConfirmedServiceResponse 类型的 readJournal 选择的抽象语法是 ReadJournal-Response。

23.3 WriteJournal(写日志)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 writeJournal 选择的抽象语法分别由 WriteJournal-Request 类型及 WriteJournal-Response 类型规定，以下是这两种类型的规定，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
  WriteJournal-Request ::= SEQUENCE {
    journalName             [0] ObjectName,
    listOfJournalEntry      [1] IMPLICIT SEQUENCE OF EntryContent }
  WriteJournal-Response ::= NULL
```

23.3.1 WriteJournal-Request

ConfirmedServiceRequest 类型的 writeJournal 选择的抽象语法是 WriteJournal-Request。

23.3.2 WriteJournal-Response

ConfirmedServiceResponse 类型的 writeJournal 选择的抽象语法是 WriteJournal-Response。

23.4 InitializeJournal(初始化日志)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 initializeJournal 选择的抽象语法分别由 InitializeJournal-Request 类型及 InitializeJournal-Response 类型规定。以下是这两种类型的规定，其描述在后两节中给出。对于本章中未提供显式导出的那些参数，它们的导出在 5.5 中描述。

```
  InitializeJournal-Request ::= SEQUENCE {
    journalName             [0] ObjectName,
    limitSpecification      [1] IMPLICIT SEQUENCE {
        limitingTime                [0] IMPLICIT TimeOfDay,
        limitingEntry               [1] IMPLICIT OCTET STRING OPTIONAL }
OPTIONAL
    }
InitializeJournal-Response ::= Unsigned32—Entries Deleted
```

23.4.1 InitializeJournal-Request

ConfirmedServiceRequest 类型的 initializeJournal 选择的抽象语法是 InitializeJournal-Request。

23.4.2 InitializeJournal-Response

ConfirmedServiceResponse 类型的 initializeJournal 选择的抽象语法是 InitializeJournal-Response。

该字段是 InitializeJournal. response 原语的已 Entries Deleted(删除登录项)参数,并且,作为 InitializeJournal. confirm 原语(如果发送的话)的 Entries Deleted 参数出现。

23.5 ReportJournalStatus(报告日志状态)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 reportJournal 选择的抽象语法分别由 ReportJournal-Request 类型及 ReportJournal-Response 类型规定。以下是这两种类型的规定,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
ReportJournalStatus-Request ::= ObjectName —Journal Name
ReportJournalStatus-Response ::= SEQUENCE {
    currentEntries              [0] IMPLICIT Unsigned32,
    mmsDeletable                [1] IMPLICIT BOOLEAN
IF ( aco )
,   accessControlList           [2] IMPLICIT Identifier OPTIONAL
            —Shall not appear in minor version one or two
ENDIF
    }
```

23.5.1 ReportJournalStatus-Request

ConfirmedServiceRequest 类型的 reportJournalStatus 选择的抽象语法是 ReportJournal Status-Request。

该字段是 ReportJournalStatus. response 原语的 Journal Name(日志名)参数,并作为 ReportJournalStatus. indication 原语(如果发送的话)的 Journal Name 参数出现。

23.5.2 ReportJournal Status-Response

ConfirmedServiceResponse 类型的 reportJournalStatus 选择的抽象语法是 ReportJournal Status-Response。

23.5.2.1 accessControlList

当且仅当 acoCBB 商定后,accessControlList 参数才出现。

23.6 CreateJournal(建立日志)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 createJournal 选择的抽象语法分别由 CreateJournal-Request 类型及 CreateJournal-Response 类型规定。以下是这两种类型的规定,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
CreateJournal-Request ::= SEQUENCE {
    journalName             [0] ObjectName }
CreateJournal-Response ::= NULL
```

23.6.1 CreateJournal-Request

ConfirmedServiceRequest 类型的 createJournal 选择的抽象语法是 CreateJournal-Request。

23.6.2 CreateJournal-Response

ConfirmedServiceResponse 类型的 createJournal 选择的抽象语法是 CreateJournal-Response。

23.7 DeleteJournal(删除日志)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 deleteJournal 选择的抽象语

法分别由 DeleteJournal-Request 类型及 DeleteJournal-Response 类型规定。以下是这两种类型的规定,其描述在后两节中给出。对于本章中未提供显式导出的那些参数,它们的导出在 5.5 中描述。

```
DeleteJournal-Request ::= SEQUENCE {
  journalName              [0] ObjectName }
DeleteJournal-Response ::= NULL
```

23.7.1 DeleteJournal-Request

ConfirmedServiceRequest 类型的 deleteJournal 选择的抽象语法是 DeleteJournal-Request。

23.7.2 DeleteJournal-Response

ConfirmedServiceResponse 类型的 deleteJournal 选择的抽象语法是 DeleteJournal-Response。

23.8 支持产生式

23.8.1 EntryContent(登录项内容)

```
EntryContent ::= SEQUENCE {
  occurrenceTime          [0] IMPLICIT TimeOfDay,
  entryForm               CHOICE {
    data                      [2] IMPLICIT SEQUENCE {
      event                       [0] IMPLICIT SEQUENCE {
        eventConditionName            [0] ObjectName,
        currentState                  [1] IMPLICIT EC-State } OPTIONAL,
      listOfVariables               [1] IMPLICIT SEQUENCE OF Journal-Variable
OPTIONAL
      },
    annotation                [3] MMSString }
  }
```

23.8.1.1 EntryForm(登录项格式)

如果 WriteJournal. request 服务原语的 Entry Form 参数的值是 DATA,那么,在 WriteJournal-Request 内选 data。如果 WriteJournal. request 服务原语的 Entry Form 参数的值为 ANNOTATION,那么,在 WriteJournal-Request 内选 annotation。

如果 ReadJournal. response 服务原语的 Entry Form 参数的值是 DATA,那么,在 ReadJournal-Response 内选 data。如果 ReadJournal. response 服务原语的 Entry Form 参数的值为 ANNOTATION,那么,在 ReadJournal-Response 内选 annotation 选择。

下列 END 语句用于结束第 7 章中已打开的模块。

```
END
```

24 向底层通信服务的映射

本章定义一种方法,借助这一方法,底层通信系统所提供的服务可以被制造报文协议机(MMPM)使用。任何不按本章说明使用这些服务,都将产生协议错误。

MMS 协议处于开放系统互连环境中应用层内。作为一个应用服务元素(ASE),这个国际标准利用并映射到底层通信系统的服务及服务原语。MMS 用户可以是应用过程中的或其他 ASE 中的元素。

24.1 PDUs 映射

所有 MMS PDUs 应该作为底层服务原语上的用户数据来传输。以下给出 PDUs 到服务的映射(所有 PDUs 在请求或应答服务原语上发送,在指示或确认服务原语上接收)。

MMS PDU	底层通信服务原语
Confirmed-RequestPDU	M-DATA 请求、指示
Confirmed-ResponsePDU	M-DATA 请求、指示
Confirmed-ErrorPDU	M-DATA 请求、指示
Unconfirmed-RequestPDU	M-DATA 请求、指示
RejectPDU	M-DATA 请求、指示
Cancel-RequestPDU	M-DATA 请求、指示
Cancel-ResponsePDU	M-DATA 请求、指示
Cancel-ErrorPDU	M-DATA 请求、指示
Initiate-RequestPDU	M-ASSOCIATE 请求、指示
Initiate-ResponsePDU	M-ASSOCIATE 应答、确认(带有接收的结果参数)
Initiate-ErrorPDU	M-ASSOCIATE 应答、确认(带有被拒绝的结果参数)
Conclude-RequestPDU	M-RELEASE 请求、指示
Conclude-ResponsePDU	M-RELEASE 应答、确认(带有接收的结果参数)
Conclude-ErrorPDU	M-RELEASE 应答、确认(带有被拒绝的结果参数)

任何其他的 MMS PDUs 到这些服务的映射都会产生协议错误。

24.2 M-ASSOCIATE(M-关联)数据

来自 M-ASSOCIATE 请求或应答服务的数据将被用于应用关联对象的属性初始化,这个关联对象是为此关联而建立的。Application Reference 参数用来指明关联中的同层 MMS 用户,它的值指明:

(1) 通信结点;

(2) 在该结点中的用户过程;

该值被用来建立应用关联对象的 &client 字段(参见 8.2)。对于 MMS 呼叫户,Responding Application Refeience 参数被用来作为 &client 字段的值。如果在请求服务原语,或者在应答服务原语中给出了 Authenticatiom Value(鉴别值)参数,那么,该参数值将被用于应用关联对象的 &anthenticationValue 字段的初始化。

24.3 应用关联的终止

一旦接收到一个有效的 Conclude-RequestPDU,MMPM 就发送一条 M-RELEASE. request 服务原语,并将这个 Conclude-RequestPDU,作为用户数据带上。

一旦接收到一个带有结果参数(指明该应用关联成功释放)的有效 Conclude-Request PDU,MMPM 就向 MMS 用户提交 Conclude. confirm 服务原语,指明 Result(+)。如果 Result 参数指明释放尝试不成功,那么,MMPM 将发送一条 M-U-ABORT. request 服务原语,并向 MMS 用户提交 Conclude. confirm 服务原语,指明 Result(+)。

24.4 直接映射的异常中止服务

MMS 异常中止服务直接映射到 M-U-ABORT 服务,因此,本国际标准不定义异常中止 PDU。

一旦接收到支持通信系统发出的、要求异常中止的指示服务原语(或者是 M-U-ABORT,或者是 M-P-ABORT),MMPM 就向 MMS 用户发送一条异常中止指示服务原语。如果 M-ABORT 请求是由 MMS 用户所在地的系统发出的(例如 M-P-ABORT),那么,MMS 异常中止指示原语中的 Locally Generated 参数的值应设定为"真"。否则,该参数的值应设定为"假"。

一旦接收到发自 MMS 用户的 MMS abort. request 服务原语,MMPM 就发出一条 M-U-ABORT.

request 服务原语。

任何时候 MMPM 都可以向 MMS 用户发送 abort. indication 服务原语和 M-P-ABORT. request 服务原语作为本地事务(由于本地测试条件的结果)。

24.5 MMS PDUs 的结构

一旦接收到对某个 MMS 服务的请求或应答服务原语(ABORT. request 服务原语、conclude. request 服务原语或 conclude. response 服务原语除外),MMPM 将会:

a) 按照该服务的协议要求(参见第 7 至第 23 章,以及附录 C 和附录 D),为原语中规定的服务构建第 7 章所要求的 PDU;

b) 按照 GB/T 16720.1 和 GB/T 16720.2 的要求,发送构成的 PDU,作为上面所规定的 M-DATA 服务原语的用户数据。

24.6 向 MMS 用户交付服务原语

一旦接收到来自底层通信系统的指示或确认服务原语(M-ABORT. indication、M-RELEASE. indication 或 M-RELEASE. confirm 除外),MMPM 将确定所接收的原语是否包含作为用户数据的有效 MMS PDU。一个有效的 MMS PDU 是指它满足定义 PDUs 的 MMS 抽象语法要求,映射到正确的服务原语(按照上述定义),并且达到与 GB/T 16720.1 和 GB/T 16720.2 定义的所有定序规则的一致性。

如果接收到的服务原语包含有效的 MMS PDU,那么,MMPM 将发送相应的指示或确认服务原语,以及在此原语中按照 GB/T 16720.1 和 GB/T 16720.2 的要求所导出的值。

如果接收到的服务原语不包含有效的 MMS PDU,那么,MMPM 将完成以下活动:

a) 如果 Initiate-RequestPDU 和 Initiate-ResponsePDU 的交换已经通过先前的应用关联间的通信成功完成,那么,MMPM 将向 MMS 用户发送一个 reject. indication,同时,构建一个 Reject-PDU(带有根据所检测出的错误而确定的参数),并在 M-DATA. request 服务原语上发送这个 PDU。

b) 否则,MMPM 将向 MMS 用户发送一个 abort. indication,并且,在应用关联存在时,发送一条 M-ABORT 服务原语。

24.7 对发送数据的正当要求

本部分要求底层通信系统必须支持全双工通信。

24.8 可靠的底层服务

本部分对丢失的报文、传送错误、错误序报文或重复报文不作处理。它假定在两个应用实体之间存在可靠的进行数据通信的底层服务。

24.9 流控制

MMS 中没有同层流控制。接收 MMPM 可以利用底层通信系统中的流控制机制去影响应用关联间的流控制,从而,也限制了同层发送数据的能力。决定何时或怎样地利用这种流控制是本地事务。

24.10 表示上下文的使用

OSI 定义一种表示上下文,作为应用与它的同层间进行通信的抽象语法,以及为传输该信息提供编码机制的传输语法。GB/T 16720 只定义在同层 MMS 用户间交换的报文的抽象语法。选择传输语法不属于本部分范围。

附录 A 仅对 OSI 环境下传输语法的使用提供指导。而在其他环境下,既可使用 OSI 传输语法,或者,也可以使用其他的、适合于这些环境的编码系统。对这种传输语法的唯一要求是:它应能从编码单值地重新形成用于特定 MMS 的抽象语法。

有关在给定通信实例上使用传输语法的协定可以是成员间主要的协议内容,或者,它也可以是底层通信系统服务内的协商内容。

24.11 抽象语法定义

本部分的本部分指定 ASN.1 对象标识符值{ iso standard 9506 part(2) mms-abstract-syntax-version1(1)}作为表示数据值集合的抽象语法，该集合中的每一个值是本部分第 7～23 章，以及附录 C、附录 D、附录 E 以及 GB/T 16720.1 第 7～23 章中定义的 ASN.1 模块的值。相应的 ASN.1 对象描述符的值是："mms-abstract-syntax-major-version1"。

GB/T 16720.1(ISO 9506-1)及 GB/T 16720.2(ISO 9506-2)的主版本号是 1，次版本号是 4。

本部分指定 ASN.1 对象标识符值{ iso standard 9506 part(2) mms-file-record-version1(2)}作为表示数据值集合的抽象语法，而该集合中的每一个值是本部分附录 B 中定义的 ASN.1 模块的值。对应的 ASN.1 对象描述符的值是："mms-file-record-version1"。

25 配置和初始化说明

25.1 引言

本章描述用于执行的配置和初始化语句，它为执行者提供各种表格，用于报告执行中各种字段的值。另外，它还在系统启动时为一些字段的初始化提供说明。每一个执行者应填写全部的 CIS。在本章中，执行者既涉及实现 VMD 所必需的硬、软件的卖主，又涉及将产品配置成一个专用设备的安装者或拥有者。

25.2 CIS 第 1 部分：初始化 VMD

表 1 中所有项的信息都应提供。对于需要附加信息或解释的项，执行者应在表中放入一个附注，以便查找由执行者提供的附加页、章或段。

执行应支持对象类 VMD 的一个对象。对于 VMD 模型的每个字段，执行者应赋予初始值。对于 VMD 中的每个预定义辅助对象，执行者应提供 &name 字段的值：或者为 1、或者为 2。值 1 表明引用该对象的完整定义(例如，应用 GB/T 16720.1 MMS 定义的对象)，值 2 表明，对于该对象的所有字段，以值的形式引用其完整定义。

表 1 CIS 执行信息

执行顺序号： 发送日期：

VMD 模型的字段	CBB	值
&executiveFunction(& 执行功能)		
&vendorName(& 卖主名)		
&modelName(& 模型名)		
&revision(& 修订版本)		
&AbstractSyntaxes(& 抽象语法)		
&accessControl(& 访问控制)		
&Capabilities(& 能力)		提供表 2
&local-detail(& 本地-细目)		
&AccessControlLists(& 访问控制表)		提供表 3
&Domains(& 域)		提供表 4
&ProgramInvocations(& 程序调用)		提供表 5
&UnitControls(& 单元控制)		提供表 6
&Unnamedvariables(& 无名变量)	vadr	提供表 7
&Namedvariables(& 有名变量)	vname	提供表 8

表 1（续）

VMD 模型的字段	CBB	值
&NamedVariablesLists(& 有名变量表)	vlis	提供表 9
&NamedTypes(& 有名类型)	vname	提供表 10
&DataExchange(& 数据交换)		提供表 11
&Semaphores(& 信标)		提供表 12
&OperatorStations(& 操作员站)		提供表 13
&EventConditions(& 事件条件)		提供表 14
&EventActions(& 事件活动)		提供表 15
&EventEnrollments(& 事件登录)		提供表 16
&EventConditionList(& 事件条件表)	cspi	提供表 17
&Journals(& 日志)		提供表 18
&Selected-program-Invocation(& 选定的程序调用)	csr	

25.2.1 &executiveFunction

执行者应提供 Application Reference(应用引用值)，用于在其网络环境内标识该执行。这个 Application Reference 的特性取决于该执行所依附的网络。对于 OSI 网络，这个字段是一个对象标识符。

25.2.2 &vendorName

执行者应提供一个字符串值，用于标识系统的卖主。

25.2.3 &modelName

执行者应提供一个字符串值，用于指明执行的模型。

25.2.4 &revision

执行者应提供一个字符串值，用于指明执行的版本。

25.2.5 &AbstractSyntaxes

执行者应提供一个抽象语法集，该抽象语法集可以在 Domain upload/download 操作中，或在 Start 及 Resume 操作中为这个执行所识别。该集合可以是空的。

25.2.6 &EATransactions

执行者应给此域赋予一个空集。

25.2.7 &accessControl

执行者应提供一个预定义访问控制表对象，该对象描述这个 VMD 的访问控制特性，它包含在 25.2.12中。

25.2.8 &logicalStatus(& 逻辑状态)

在系统启动时，该域必须初始化，以反映 VMD 底层硬件的条件。它的值应该是值 state-changes-allowed(允许的状况改变)、no-state- changes-allowed(不允许的状况改变)、limited-service-permitted(允许的有限服务)和 support-services-allowed(允许的支持服务)值中的一个。

25.2.9 &Capabilities

执行者应填写描述 VMD 每项能力的表 2。表中的行数可扩充，以适于描述所有的能力。该表可以是空的。

表 2 能 力 描 述

能力(字符串)	含 义	语法分析规则

25.2.10 **&physicalStatus(& 物理状态)**

在系统启动时,该域必须初始化,以反映 VMD 底层硬件的条件。它的值应是值 operational、partially-operational、inoperable 和 needs-commissioning 之一。

25.2.11 **&local_detail**

执行者应提供 &local-detail BIT STRING 的初始值,以及每一位的含义。如果没有定义 local-detail 位,那么,该值为"B"。

25.2.12 **&AccessControl lists**

执行者应提供一个预定义的访问控制表对象集。对于该集中的每一个对象,执行者要填写表 3,以便为该对象的所有字段元素提供有关的定义,或值。

访问控制表对象集至少应包含一个被表 1 中的 &accessControl 字段所提及的对象。

表 3 预定义访问控制对象

访问控制表对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&readAccessCondition(& 读访问条件)		
&storeAccessCondition(& 存储访问条件)		
&writeAccessCondition(& 写访问条件)		
&loadAccessCondition(& 装载访问条件)		
&executeAccessCondition(& 执行访问条件)		
&deleteAccessCondition(& 删除访问条件)		
&editAccessCondition(& 编辑访问条件)		
& AccessControlLists(& 访问控制表)		
&Domains(& 域)		
&ProgramInvocations(& 程序调用)		
&UnitControls(& 单元控制)		
&UnnamedVariables(& 无名变量)	vadr	
&NamedVariables(& 有名变量)	vnam	
&NamedVariablesLists(& 有名变量表)	vlis	
&NamedTypes(& 有名类型)	vnam	
&DataExchanges(& 数据交换)		
&Semaphores(& 信标)		

表 3（续）

访问控制表对象	CBB	值或引用
&OperatorStations(& 操作员站)		
&EventConditions(& 事件条件)		
&EventActions(& 事件活动)		
&EventEnrollments(& 事件登录)		
&Journals(& 日志)		
&EventConditionLists(& 事件条件表)	cspi	

25.2.13 & Domains

执行者应提供一个预定义的字段对象集。对于这样的每一个对象，执行者应填写表 4，以便为该对象的所有字段元素提供有关的定义或值。

对于 &Capabilities 字段，可以建立一个或多个表 2 的拷贝，用以说明分配给这个预定义域的能力。根据 &programInvocations 字段是否为空，将 &states 字段赋值为 ready 或 in-use。&aAssociation 字段应为空。

对于附属于该域的每个预定义对象，应建立与这个对象类型相对应的一个表，并在表 4 的 &content 域中引用它。

这个集合可以是空的。

表 4 预定义域对象

域 对 象	CBB	值或引用
&name(& 名字)		
引用定义		
&Capabilities(& 能力)		
&state(& 状况)		
&aAssociation(&a 关联)		空
&accessContorl(& 访问控制)		
&sharable(& 共享的)		
&ProgramInvocation(& 程序调用)		
&uploadsInProgress(& 处理中的上载)		设置为 0
&NamedVariables(& 有名变量)	vnam	提供表 8
& NamedVariablesLists(& 有名变量表)	vlis	提供表 9
&NamedType(& 有名类型)	vnam	提供表 10
&EventConditions(& 事件条件)		提供表 14
&EventActions(& 事件活动)		提供表 15
&EventEnrollments(& 事件登录)		提供表 16
&EventConditionLists(& 事件条件表)	cspi	提供表 17

25.2.14 &ProgramInvocation

执行者应提供一个预定义的程序调用对象集。对于该集合中的每个对象，执行者应填写表 5，以便为这个对象的所有域元素提供有关的定义或值。

这个预定义的程序调用集可以是空的。如果此集不为空，那么，表 4 也不应为空。

表 5　预定义程序调用对象

程序调用对象	CBB	值或引用
&name(& 名字)		
引用定义		
&programInvocationState(&program Invocation State)		
&Domains(& 域)		
&accessControl(& 访问控制)		
&reusable(& 可重复使用的)		
&monitior(& 监控器)		
&eventCondition(& 事件条件)		
&eventAction(& 事件活动)		
&eventEnrollment(& 事件登录)		
&executionArgument(& 执行自变量)		
&control(& 控制)	csr	
&controlling-pragram-Invocation(& 控制程序调用)	csr	
&controlled-pragram-Invocations(& 被控程序调用)	csr	

25.2.15　&UnitControls

执行者应提供一个预定义的单元控制对象集。对于集合中的每个对象，执行者应填写表 6，以便为该对象的所有域元素提供有关的定义或值。

这个预定义的单元控制对象集可以是空的。如果它不为空，那么，表 4 也不应是空的。

表 6　预定义单元控制对象

单元控制对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&Domains(& 域)		
&ProgramInvocation(& 程序调用)		

25.2.16　&Unnamedvariables

如果执行能够支持 vadr 一致性构造块，那么，执行者应填写表 7，以便提供构成无名变量的相关信息，包括格式和地址范围(它是否是数字的、符号的或是无结构的)，以及用于将类型说明选择与地址联系起来的算法。

表 7　无名变量对象

无名变量	说　明
Address(地址)	
Type Description(类型说明)	

25.2.17　&NamedVariables

如果执行能够支持 vnam 一致性构造块，那么，执行者应提供一个预定义的有名变量对象集。对于

集中的每个对象，执行者要填写表 8，以便为该对象的所有字段元素提供有关的定义或值。

有名变量对象集可以是空的。

表 8 预定义有名变量对象

有名变量对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&typeDescription(& 类型说明)		
&accessmethod(& 访问方法)		
&address(& 地址)	vadr	
&meaning(& 含义)	sem	

25.2.18 &NamedVariableList

如果执行能够同时支持 vnam 一致性构造块和 vlis 一致性构造块，那么，执行者应提供一个预定义的有名变量表对象集。对于集合中的每个对象，执行者要填写表 9，从而，为对象的所有字段元素提供有关的定义或值。

对于 &listOfVariables 字段中的每一项，执行者应指明引用的是无名变量，还是有名变量，同时，如果采用的话，还要提供替代访问说明。

预定义的有名变量对象集可以是空的。

表 9 预定义有名变量表对象

有名变量表对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&listOfVariables(& 变量表)		
unnamedItem(无名项)	vadr	
namedItem(有名项)	vnam	
AlternateAccess(替代访问)	valt	

25.2.19 &NamedType

如果执行能够支持 vnam 一致性构造块，那么，执行者应提供一个预定义的有名类型对象集。对于集中的每个对象，执行者要填写表 10，以便为该对象的所有字段元素提供有关的定义或值。

预定义的有名类型对象集可以是空的。

表 10 预定义有名类型对象

有名类型对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&typeDescription(& 类型说明)		
&meaning(& 含义)	sem	

25.2.20 &DataExchanges

执行者应提供一个预定义数据交换对象集。对于其中的每个对象，执行者要填写表 11，以便为该对象的所有字段元素提供有关的定义或值。

预定义数据交换对象集可以是空的。

表 11 预定义数据交换对象

数据交换对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&request(& 请求)		
&response(& 应答)		
&linked(& 被连接的)		
&ProgramInvocation(& 程序调用)		

25.2.21 &Semaphores

执行者应提供一个预定义的信标对象集。对于集中的每个对象，执行者要填写表 12，以便为该对象的所有字段元素提供有关的定义或值。填写 &numberOfTokens，还是填写 &Named-Tokens，取决于 &Class 的值。

如果给出 &numberOfOwnedTokens 字段，它的初始值为 0。字段 &Owners 字段和 &Requesters 字段被初始设置为空集。

预定义信标对象集可以是空的。如果该集不空，那么，&EventCondition 也不为空。

表 12 预定义信标对象

信标对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&class(& 类)		
&numberOfTokens(& 令牌号)		
&Named-Tokens(& 有名令牌)		
&eventCondition(& 事件条件)		

25.2.22 &OperatorStations

执行者应提供一个预定义的操作员站对象集。对于集中的每个对象，执行者要填写表 13，以便为该对象的所有字段元素提供有关的定义或值。

如果给出 &inputBuffer 字段，那么，它的初始设置为空串。&outputBuffer 字段初始设置为空串。&state 字段初始设置为 idle。

预定义操作员站对象集可以是空的。

表 13 预定义操作员站对象

操作员站对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&stationType(& 站类型)		

25.2.23 &EventCondition

执行者应提供一个预定义的事件条件对象集。对于集中的每个对象,执行者要填写表 14,以便为该对象的所有字段元素有关的定义或值。

&timeToActive 字段和 &timeToIdle 字段初始设置为 undefined.

预定义的事件条件对象集可以是空的。

表 14 预定义事件条件对象

事件条件对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&ecClass(&ec 类别)		
&ecState(&ec 状况)		
&priority(& 优先级)		
&severity(& 艰难程度)		
&EventEnrollments(& 事件登录)		
&enabled(& 能够做的)		
&alarmSummaryReports(& 报警汇总报告)		
&monitoredVariable(& 监控变量)		
&evaluationInterval(& 估计时限)		
&displyEnhancement(& 显示增强)	cspi	
&group-priority-Override(& 组优先超越)	cspi	
&ReferencingEventConditionLists(& 引用事件条件表)	cspi	

25.2.24 &EventAction

执行者应提供一个预定义的事件活动对象集。对于集中的每个对象,执行者要填写表 15,以便为该对象的所有字段元素提供引用定义,或提供引用值。

&timeToActive 字段和 &timeToIdle 字段应初始设置为 undefined。

预定义的事件活动对象集可以是空的。

表 15 预定义事件活动对象

事件活动对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&confirmedServiceRequest(& 确认服务请求)		
&Modifiers(& 修饰符)		
&EventEnrollments(& 事件登录)		

25.2.25 &EventEnrollments

执行者应提供一个预定义的事件登录对象集。对于集中的每个对象,执行者要填写表 16,以便为该对象的所有字段元素提供有关的定义或值。

&aAssociation 字段应初始设置为空。如果给出 &invokeID 字段,它应初始设置为 0。如果给出 &notificationLost 字段,那么,应将它初始设置为“假”。如果给出 &timeActiveAck 字段和 &timeIdleACK 字段,那么,应将它们初始设置为 undefined。如果给出 &ackState 字段,那么,将它初始设置为 acked。

预定义的事件登录对象集可以是空的。如果它非空,那么,&EventCondition 集也不为空。

表 16 预定义事件登录对象

事件登录对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&ecClass(&ec 类别)		
&eventCondition(& 事件条件)		
&ecTransition(&ec 事务)		
&remainingDelay (& 剩余延迟)		
&eventAction(& 事件活动)		
&duration(& 持续时间)		
&clientApplication(& 客户应用)		
&aaRule(&aa 规则)		
&displayEnhancement(& 显示增强)	cspi	

25.2.26 &EventConditionList

执行者应提供一个预定义的事件条件表对象集。对于集中的每个对象,执行者要填写表 17,以便为该对象的所有字段元素提供有关的定义或值。

预定义的事件条件表对象集可以是空的。如果它不为空,那么,&EventConditions 集也不为空。

表 17 预定义事件条件表对象

事件条件表对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&EventConditions(& 事件条件)		
&EventConditionLists(& 事件条件表)	recl	
&ReferencingEventConditionLists(& 引用事件条件表)	recl	

25.2.27 **&Journals**

执行者应提供一个预定义的日志对象集。对于集中的每个对象，执行者要填写表 18，以便为该对象的所有字段元素提供有关的定义或值。

预定义的日志对象集可以是空的。

表 18 预定义日志对象

日志对象	CBB	值或引用
&name(& 名字)		
引用定义		
&accessControl(& 访问控制)		
&Entries(& 登录项)		

对于日志对象的 &Entries 字段中的每一项，执行者应填写表 19，以便为该对象的所有字段元素提供有关的定义或值。

在给定的日志内，&entry 字段的指定值应为唯一的。

本集可以是空的。

表 19 预定义日志项对象

日志项对象	CBB	值或引用
&journal(& 日志)		
&entry(& 项)		
&clientApplication(& 客户应用)		
&timeStamp(& 时间标签)		
&orderOfReceipt(& 接收顺序)		
&informationType(& 信息类型)		
&textComment(& 正文注释)		
&eventTransitionRecord(& 事件事务记录)		
&journalVariables(& 日志变量)		

25.2.28 **&operation-State**

如果执行能够支持 csr 一致性构造块，那么，执行者应将该字段初始化，使其反映 VMD 底层硬件的条件。它的可能的值是：idle、loaded、ready、execution、motion-paused、manualInterventionRequired。

25.2.29 **&Safety-interlocks-violated(& 安全-互锁遭违背)**

如果执行能够支持 csr 一致性构造块，那么，执行者应将此字段初始化，使其反映 VMD 底层硬件的条件。

25.2.30 **&any_Resource_power_on(& 资源接通)**

如果执行能够支持 csr 一致性构造块,那么,执行者应将此字段初始化,使其反映 VMD 底层硬件的条件。

25.2.31 **&local-control(& 局部控制)**

如果执行能够支持 csr 一致性构造块,那么,执行者应将此字段初始化,使其反映 VMD 底层硬件的条件。

25.2.32 **&selected-program-Invocation**

如果执行能够支持 csr 一致性构造块,那么,执行者应提供所选程序调用的初始值。这个初始值可为空。

25.3 **CIS 第 2 部分:服务和参数 CBBs**

执行者应该提供下表中列出的、由执行支持的有关服务和参数 CBBs 的信息,这些信息指明,当按照本部分所定义的抽象语法进行操作时,执行是否满足服务需求、或客户需求,或者,两者同时满足。术语:客户、服务、请求和应答在 GB/T 16720.1 第 5 章定义。客户和服务对每个 CBB 的需求在 GB/T 16720.1 第 25 章描述。

25.3.1 **环境和通用管理服务**

对于本节,应在表 20 中指明执行是否能够支持请求者角色,或应答者角色,或二者都支持。

表 20 环境和通用管理服务

环境和通用管理	请求者	应答者
Initiate(启动)		
Conclude(结束)		
Concel(取消)		

在表 21 中,执行者应指明:

1) 是否支持 char、csr、csnc、csplc 和 cspi 参数 CBBs。
2) 本地细目呼叫/受叫(local Detail Calling/Called)。如果用于启动服务中的本地细目呼叫参数和本地细目受叫参数作为执行的一部分来提供,那么,执行者应说明这些参数的细节,提供所使用的每一个值的语义。
3) 对时间的支持程度:执行者应说明在现有系统时钟情况下,是否支持一天的日期和时间。执行者还应说明是否支持时间顺序标识符。
4) 以毫秒为单位的时间粒度:执行者应说明能够用于时间分辨的、以毫秒为单位的最小时间单位。该值仅在日期和时间都得以支持的情况下才有意义。

表 21 环境和通用管理参数

通用管理参数	值
char	
csr	
csnc	
csple	
cspi	
Local Detail(本地细目)	
Support for time(时间支持)	
Granularity of time (ms)(时间粒度)	

25.3.2 访问控制服务

执行者应在表22中指明，对于所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表22 访问控制服务

访问控制	服务器	客户
Define Access Control List(定义访问控制表)		
Get Access Control List Attributes(获取访问控制表属性)		
Report Access Controlled Objects(报告访问控制对象)		
Delete Access Control List(删除访问控制表)		
Change Access Control(修改访问控制)		

执行者应在表23中指明执行是否支持aco CBB。

表23 访问控制参数

访问控制参数	
aco	

25.3.3 VMD支持服务

执行者应在表24中指明，对于所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表24 VMD支持服务

VMD支持	服务器	客户
Status(状态)		
Unsolicited Status(非请求状态)		
Get Name List(获取名字表)		
Identify(识别)		
Rename(更名)		
Get Capability List(获取能力表)		
VMD Stop(VMD停止)		
VMD Reset(VMD复位)		

若执行指明应答者支持启动服务，那么。该执行需支持服务器对于识别服务的作用。

在表25中，执行者应指明：

1) local Detail(在状态和非请求状态服务中的参数)：执行者应说明本地细目是什么，以及如何在位串中对该参数进行语法分析(同时提供字符串的语法和语义)。
2) 用于扩充导出状态信息的方法：执行者应说明用于扩充导出(如10.3所述的)状态信息(如果有的话)的方法。

表25 VMD支持参数

VMD支持参数	值
Local Detail	
Extended Derivation(扩充导出)	

25.3.4 域管理服务

执行者应在表26中指明，对于表中所列服务，执行是否支持服务器角色，或支持客户角色，或二者

都支持。

表 26 域管理服务

域管理支持	服务器	客户
Initiate Download Sequence(启动域下载队列)		
Download Segment(下载段)		
Terminate Download Sequence(终止下载队列)		
Initiate Upload Sequence(启动上载队列)		
Upload Segment(上载段)		
Terminate Upload Sequence(终止上载队列)		
Request Domain Download(请求域下载)		
Request Domain Upload(请求域上载)		
Load Domain Content(装载域内容)		
Store Domain Content(存储域内容)		
Delete Domain(删除域)		
Get Domain Attributes(获取域属性)		

在表 27 中,执行者应指明:

1) 执行是否支持 tpy 参数 CBB。
2) 装载数据的格式。执行者应说明在 DownloadSegment 和 UploadSegment 服务中的 Load Data 参数的八位位组串的语义和进一步的语法定义。
3) 如果支持这些参数的 EXTERNAL 或 EMBEDDED PDV 选择,那么,执行者应说明所支持的抽象语法名。
4) 上载状态机(Upload State Mchines)的最大个数。执行者应提供可在一个域中同时调用的上载状态机的最大个数。

表 27 域管理参数

域参数	值
tpy	
Load Data-八位位组串	
Load Data-抽象语法	
MAX Upload state Machines	

25.3.5 程序调用管理服务

执行者应在表 28 中指明,对于表中所列服务,执行是否支持服务器角色,或支持客户角色,或二者都支持。

表 28 程序调用管理服务

程序调用管理支持	服务器	客户
Create Program Invocation(建立程序调用)		
Delete Program Invocation(删除程序调用)		
Start(开始)		
Stop(停止)		

表 28（续）

程序调用管理支持	服务器	客　户
Resume(恢复)		
Reset(复位)		
Kill(截杀)		
Get Program Invocation Attributes(获取程序调用属性)		
Select(选取)		
Alter Pregram Invocation Attributes(变更程序调用属性)		
Reconfigure Program Invocation(重新配置程序调用)		

在表 29 中，执行者应指明：

1) 执行自变量(开始和复位服务中的参数)：执行者应说明所支持的执行自变量参数中字符串的最大字符个数。
2) 执行自变量字符串的语法分析规则。
3) 如果支持该参数的 EXTERNAL 或 EMBEDDED PDV 选择，那么，执行者应说明所支持的抽象语法名。
4) &programLocation 字段(如果给出的话)的记法格式。
5) 是否支持步进操作方式。

表 29　程序调用管理参数

程 序 调 用 参 数	值
Execution Argument max size(执行自变量最大尺寸)	
Execution Argument parse rules(执行自变量语法分析规则)	
Execution Argument abstract syntax(执行自变量抽象语法)	
& programLocation(& 程序位置)	
Step Mode(步进方式)	

25.3.6　单元控制服务

执行者应在表 30 中指明，对于表中所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表 30　单元控制服务

单元控制管理支持	服务器	客　户
Initiate Unit Control Load(启动单元控制装载)		
Unit Control Load Segment(单元控制装载段)		
Unit Control Upload(单元控制上载)		
Start Unit Control(开始单元控制)		
Stop Unit Control(停止单元控制)		
Create Unit Control(创建单元控制)		
Add To Unit Control(加入单元控制添加)		
Remove From Unit Control(退出单元控制取消)		

表 30（续）

单元控制管理支持	服务器	客　户
Get Unit Control Attributes(获取单元控制属性)		
Load Unit Control From File(从文件装载单元控制)		
Store Unit Control To File(向文件存入单元控制)		
Delete Unit Control(删除单元控制)		

25.3.7 变量访问服务

执行者应在表 31 中指明，对于表中所列服务，执行是否支持服务器角色，或者支持客户角色，或者，二者都支持。

表 31 变量访问服务

变量访问支持	服务器	客　户
Read(读)		
Write(写)		
Information Report(信息报告)		
Get Variable Access Attributes(获取变量访问属性)		
Define Named Variable(定义有名变量)		
Delete Variable Access(删除变量访问)		
Define Named Variable List(定义有名变量表)		
Get Named Variable List Attributes(获取有名变量表属性)		
Delete Named Variable List(删除有名变量表)		
Define Named Type(定义有名类型)		
Get Named Type Attributes(获取有名类型属性)		
Delete Name Type(删除有名类型)		

在表 32 中，执行者应指明：

1) str1、str2、vnam、vadr、valt 和 vlis 参数 CBBs 的值。
2) 所支持的 nest 参数的最大值。
3) 如果执行支持 GB/T 16720.1 附录 F 中描述的工具，那么，应报告 real 参数 CBBs。
4) 对变量的可中断访问：执行者应说明在什么条件下可确保对变量的访问是不可中断的。同时也应说明使用 MMS 服务如何可能达到这一水平。
5) 如果支持 vadr，那么，执行者应说明是否支持 SINGLE 变量说明方式，或者 UNNAMED 变量说明方式，或者，二者都支持(参见 GB/T 16720.1 的 12.5.2.1)

表 32 变量访问参数

变量访问参数	值
str1	
str2	
vnam	
vadr	

表 32（续）

变量访问参数	值
valt	
vlis	
nest	
Uninterruptible access	
SINGLE	
UNNAMED	

在表 33 中，执行者应指明作为服务器所支持的 Data 类型的 Size 字段的可能值。

表 33 数据参数

Data-Size 参数	值
bit-string	
interger	
unsigned	
floating-point	
octet-string	
visible- string	
binary-time	
bcd	
mMSString	

25.3.8 数据交换服务

执行者应在表 34 中指明，对于表中所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表 34 数据交换服务

数据交换支持	服务器	客户
Exchange Data(交换数据)		
Get Data Exchange Attributes(获取数据交换属性)		

25.3.9 信标管理服务

执行者应在表 35 中指明，对于表中所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表 35 信标管理服务

信标支持	服务器	客户
Take Control(取得控制)		
Relinquish Control(放弃控制)		
Define Semaphore(定义信标)		
Delete Semaphore(删除信标)		
Report Semaphore Status(报告信标状态)		

表 35（续）

信标支持	服务器	客　户
Report Pool Semaphore Status(报告预存信标状态)		
Report Semaphore Entry(报告信标项状态)		
Attach To Semaphore Modifier(附加于信标的修饰符)		

在表 36 中，执行者应指明对信标优先级的处理；执行者应说明用于处理信标（如果支持的话）优先级的算法。事件粒度参数由表 25 给出。

表 36　信标管理参数

优先级处理	说　明
算法	

25.3.10　操作员通信服务

执行者应在表 37 中指明，对于表中所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表 37　操作员通信服务

操作员站支持	服务器	客　户
Input(输入)		
Output(输出)		

在表 38 中，执行者应指明 Input Time Out 参数的最大值（以秒为单位）。

表 38　操作员通信参数

操作员站参数	值
Input time out	

25.3.11　事件管理服务

执行者应在表 39 中指明，对于表中所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表 39　事件管理服务

事件管理支持	服务器	客　户
Trigger Event(触发事件)		
Event Notification(事件通告)		
Acknowledge Event Notification(确认收到事件通告)		
Get Alarm Summary(获取报警总汇)		
Get Alarm Enrollment Summary(获取报警登录总汇)		
Attach To Event Condition Modifier(附加于事件条件的修饰符)		

25.3.12　事件条件服务

执行者应在表 40 中指明，对于表中所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表 40 事件条件服务

事件条件管理支持	服务器	客户
Define Event Condition(定义事件条件)		
Delete Event Condition(删除事件条件)		
Get Event Condition Attributes(获取事件条件属性)		
Report Event Condition Status(报告事件条件状态)		
Alter Event Condition Monitoring(变更事件条件监控)		

在表 41 中,执行者应指明执行是否能支持 cei、des 和 dei 参数 CBBs。

表 41 事件条件参数

事件条件参数	值
cei	
des	
dei	

注: des 和 deiCBBs 也用于事件登录服务。

25.3.13 事件活动服务

执行者应在表 42 中指明,对于表中所列服务,执行是否支持服务器角色,或支持客户角色,或二者都支持。

表 42 事件活动服务

事件活动管理支持	服务器	客 户
Define Event Action(定义事件活动)		
Delete Event Action(删除事件活动)		
Get Event Action Attributes(获取事件活动属性)		
Report Event Action Status(报告事件活动状态)		
Alter Event Action Status(变更事件活动状态)		

25.3.14 事件登录服务

执行者应在表 43 中指明,对于表中所列服务,执行是否支持服务器角色,或支持客户角色,或二者都支持。

表 43 事件登录服务

事件登录管理支持	服务器	客 户
Define Event Enrollment(定义事件登录)		
Delete Event Enrollment(删除事件登录)		
Get Event Enrollment Attributes(获取事件登录属性)		
Report Enrollment Status(报告事件登录状态)		
Alter Event Enrollment(变更事件登录)		

25.3.15 事件条件表服务

执行者应在表 44 中指明,对于表中所列服务,执行是否支持服务器角色,或支持客户角色,或二者都支持。

表 44 事件条件表服务

事件条件表支持	服务器	客 户
Define Event Condition List(定义事件条件表)		
Delete Event Condition List(删除事件条件表)		
Add Event Condition List Reference(添加事件条件表引用)		
Remove Event Condition List Reference(取消事件条件表引用)		
Get Event Condition List Attributes(获取事件条件表属性)		
Report Event Condition List Status(报告事件条件表状态)		
Alter Event Condition List Monitoring(变更事件条件表监控)		

在表 45 中,执行者应指明该执行是否能支持 recl 参数 CBB。

表 45 事件条件表参数

事件条件表参数	值
recl	

25.3.16 日志管理服务

执行者应在表 46 中指明,对于表中所列服务,执行是否支持服务器角色,或支持客户角色,或二者都支持。

表 46 日志管理服务

日志管理支持	服务器	客 户
Read Journal(读日志)		
Write Journal(写日志)		
Initialize Journal(初始化日志)		
Report Journal Status(报告日志状态)		
Create Journal(建立日志)		
Delete Journal(删除日志)		

25.3.17 错误

在表 47 中,执行者应该指明:

1) Additional Code(Error 类型中的参数):执行者应说明在 Additional Code 参数中使用的代码。
2) Additional Detail(Error 类型中的参数):执行者应说明在 Additional Detail 参数中使用的细目(Detail),提供用于字符串的八位位组的最大和最小数目,以及字符串的语法和语义。

表 47 错误参数

错误参数	值	含 义
Additional Code(附加代码)		
Additional Detail(附加细目)		

25.3.18 文件访问服务

执行者应在表 48 中指明,对于表中所列服务,执行是否支持服务器角色,或支持客户角色,或二者都支持。

表 48 文件访问服务

文件访问支持	服务器	客户
Obtain File(获得文件)		

25.3.19 文件管理服务

执行者应在表 49 中指明，对于表中所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表 49 文件管理服务

文件管理支持	服务器	客户
File Open(打开文件)		
File Read(读文件)		
File Close(关闭文件)		
File Rename(文件改名)		
File delete(删除文件)		
File Directory(文件目录)		

在表 50 中，执行者应指明：

File Name Syntax(文件名语法)：执行者应说明在本地文件库中使用的文件名的语法，说明用于层次文件系统的定界字符。同时，还应说明对于文件名的可接受字符集和不可接受字符集，以及文件名的长度限制。

表 50 文件管理参数

文件管理参数	语法	字符	长度
File Name(文件名)			

25.3.20 分散访问服务

执行者应在表 51 中指明，对于表中所列服务，执行是否支持服务器角色，或支持客户角色，或二者都支持。

表 51 分散访问服务

分散访问支持	服务器	客户
Define Scattered Access(定义分散访问)		
Get Scattered Access Attributes(获取分散访问属性)		

如果执行支持附录 E 中描述的任一工具，那么就应支持 vsca 参数 CBBs。

在表 52 中，执行者应指明，该执行是否支持 vsca CBB。

表 52 分散访问参数

分散访问参数	值
vsca	

附 录 A
（规范性附录）
M-服务与 ACSE 和表示服务的关系

本章定义一种方法，借助此方法，可以用关联控制服务元素（ACSE）和表示层服务来实现制造报文协议机（MMPM）所需要的 M-服务。任何不按本章说明使用 ACSE 服务或表示服务都会导致协议错误。

MMS 协议处于开放系统互连环境中的应用层内。作为一个应用服务元素（ASE），本国际标准使用并映射到 ACSE 和表示层的服务和服务原语。MMS 用户可以是应用过程或其他 ASE 内的元素。

A.1 映射 M-服务

所有 MMS PDUs 将作为 ACSE 或表示服务原语的用户数据来传送。M-服务向 ACSE 和表示服务的映射如下：

M-服务	ACSE 或表示服务
M-ASSOCIATE	A-ASSOCIATE
M-RELEASE	P-DATA，A-RELEASE
M-DATA	P-DATA
M-U-ABORT	A-U-ABORT
M-P-ABORT	A-P-ABORT

A.1.1 M-ASSOCIATE（M-映射）服务

M-ASSOCIATE 服务的原语完全与 A-ASSOCIATE 服务的原语（参见 ISO/IEC8649）相对应。相应参数给定如下。

AARQ-apdu 的参数 Calling AP Title、Calling AE Qualifier、Calling AP Invocation-identifier 和 Calling AE Invocation-identifier 将用来给 M-ASSOCIATE 服务的 Calling Application Reference 参数提供一个值。

AARQ-apdu 的参数 Called AP Title、Called AE Qualifier、Called AP Invocation-Identifier 和 Calling AE Invocation-identifier 将用来给 M-ASSOCIATE 服务的 Called Application Reference 参数提供一个值。

AARE-apdu 的参数 Responding AP Title、Responding AE Qualifier、Responding AP Invocation-identifier 和 Responding AE Invocation-identifier 将用来给 M-ASSOCIATE 服务的 Responding Application Reference 参数提供一个值。

AARQ-apdu 的参数 Authentication Value（如果给出的话）将用来给 M-ASSOCIATE 服务的 Argument 的 Authentication Value 字段提供一个值。

AARE-apdu 的参数 Authentication Value（如果给出的话）将用来给 M-ASSOCIATE 服务的 Result 的 Authentication Value 字段提供一个值。

AARQ-apdu 的参数 User Data 将用来给 M-ASSOCIATE 服务的 Argument 的 User Data 字段提供一个值。

AARE-apdu 的参数 User Data 将用来给 M-ASSOCIATE 服务的 Result 的 User Data 字段提供一个值。

AARQ-apdu 的其他参数将用来给 M-ASSOCIATE 服务的 Argument 的 Other Communication Parameters 字段提供一个值。

AARE-apdu 的其他参数将用来给 M-ASSOCIATE 服务的 Result 的 Other Communication

Parameters 字段提供一个值。

A.1.2 M-RELEASE 服务

在 OSI 内，M-RELEASE 服务作为一个 P-DATA 服务序列(其后跟随一个 A-RELEASE 服务)来执行。过程和相应参数如下。

A.1.2.1 请求 MMS 用户

当 MMPM 接收到来自请求 MMS 用户的 Conclude. request 原语时，MMPM 发送 P-DATA 请求原语，带有作为其 User Data 字段的 Conclude-RequestPDU。

一旦接收到包含有效 Conclude-responsePDU 的 P-DATA 指示，MMPM 就发送不带用户数据的 ACSE A-RELEASE. request 服务原语。

一旦接收到 ACSE A-RELEASE. confirm 服务原语(它的用户数据被忽略)和指明成功释放该应用关联的一个结果参数时，MMPM 将向 MMS 用户传送一个 Conclude. confirm 服务原语，指明 Result(+)。如果结果参数指明该释放企图不成功，那么，MMPM 发送一个 ACSE A-ABORT. request 服务原语，同时，向 MMS 用户传送一个 Conclude. confirm 服务原语，指明 Result(+)。

A.1.2.2 应答 MMS 用户

一旦接收到包含有效 Conclude-RequestPDU 的 P-DATA 指示，MMPM 就向应答 MMS 用户发送 Conclude. indication 原语。

一旦接收到来自应答 MMS 用户的表明认可的 Conclude. response 原语，MMPM 就发送 P-DATA 请求原语，带有作为其 User Data 字段域的 Conclude-ResponsePDU。

一旦接收到 ACSE A-RELEASE. indication 服务原语(它的用户数据被忽略)，MMPM 就发送不带用户数据的 ACSE A-RELEASE. response 服务原语，连同一个结果参数集，以指示成功释放该应用关联。同时，MMPM 向请求 MMS 用户发送一个 Conclude. response 原语，指明成功。

一旦接收到来自应答 MMS 用户的指明拒绝的 Conclude. response 原语，MMPM 就发送 P-DATA 请求原语，带有作为其 User Data 字段域的 Conclude-ErrorPDU。MMPM 发送不带用户数据的 ACSE A-RELEASE. response 服务原语，连同结果参数集，以指明释放该应用关联失败。MMPM 还向请求 MMS 用户发送一个 Conclude. response 原语，指明失败。

A.2 M-DATA(M-数据)服务

M-DATA 服务直接映射到 P-DATA 服务。M-DATA 服务的 User Data 参数是 P-DATA 服务的 User Data 参数。

A.3 M-U-ABORT(M-U-异常中止)服务

M-U-ABORT 服务直接映射到 A-U-ABORT 服务。M-U-ABORT 服务的 Abort Source 参数是 A-U-ABORT 服务的 Abort Source 参数。

A.4 M-P-ABORT(M-P-异常中止)服务

M-P-ABORT 服务直接映射到 A-P-ABORT 服务。M-P-ABORT 服务的 Abort Source 参数是 A-P-ABORT 服务的 Abort Source 参数。

A.5 表示上下文的使用

OSI 支持在建立关联时对表示上下文进行协商。与第一版不同，本部分只定义一个抽象语法，所以，在本部分中不会出现多个抽象语法的问题。然而，本部分可以在多种传输语法上实施，同时，在 OSI

通信系统中操作 MMS 需要提供 ACSE 抽象语法，所以，对特定关联的表示上下文进行协商仍然是需要的。

呼叫 AE 将利用 ACSE A-Associate. request 服务原语的 Presentation Context Definition List 参数来申请一个或多个表示上下文的序列。同样地，应答 AE 将利用 ACSE A-Associate. response 服务原语的 Presentation Context Result List 参数来接收或拒绝每一个申请的元素(参见 GB/T 16688 和 GB/T 15695)。

A.6 传输语法定义

ASN.1 对象标识符和对象描述符的值{ joint-iso-ccitt asn1(1) basic-encoding(1) }和"Basic Encoding of a single ASN.1 type"(被赋于 ISO/IEC 8825 中的信息对象)可以用来作为这个抽象语法的传输语法名。

A.7 应用上下文名

为了能使用仅包含 ACSE 和 MMS(作为 ASEs)的应用，对象标识符值：
{ iso standard 9506 part(2) mms-application-context-version1(5) }和对象描述符的值：
"ISO MMS"被赋于类型为"ACSE-1. ApplicationContextName"的一个信息对象(按 GB/T 16687 中的定义)。由于这个对象标识符在本部分中指定，所以，要包含括号"part(2)"，这个应用上下文名应该涉及 GB/T 16720.1 和 GB/T 16720.2 中提出的要求。

A.7.1 ApplicationReference

本章定义 Application Reference 参数和 Authentication value 参数，它们作为 Application Association 对象的字段来使用。

```
  MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1(4)}
DEFINITIONS ::= BEGIN
EXPORTS
    ApplicationReference,
    Authentication-value;
IMPORTS
    AP-title,
    AP-invocation-identifier,
    AE-qualifier,
    AE-invocation-identifier,
    Authentication-value
    FROM ACSE-1
        { joint-iso-itu-t association-control(2) modules(0) apdus(0) version1(1)};
ApplicationReference ::= SEQUENCE {
        ap-title            [0] ACSE-1. AP-title OPTIONAL,
        ap-invocation-id    [1] ACSE-1. AP-invocation-identifier OPTIONAL,
        ae-qualifier        [2] ACSE-1. AE-qualifier OPTIONAL,
        ae-invocation-id    [3] ACSE-1. AE-invocation-identifier OPTIONAL
        }
END
```

ApplicationReference 参数应利用 GB/T 16720.1 第 6 章中的 Application Reference 服务定义，按照本部分 5.5 中提供的规则导出出来。这个 ASN.1 定义利用了 AP-title、AP- invocation-id、AE-qualifier 和 AE-invocation-id 类型，这些类型的定义在 GB/T 16687 提供的 ACSE-1 模块定义中。

在 ApplicationReference 的任一使用的实例中，该参数所用值的选取应使应用引用足以唯一地、单值地标识引用 MMS 服务所需要的 Application Process、Application Process Invocation、Application Entity 或 Application Entity Invocation。

注：加在应用层有名和寻址上的附加信息可在 GB/T 9387.3、GB/T 17176、GB/T 16688 和 GB/T 16687 中找到。

附 录 B
（规范性附录）
配置和初始化的抽象格式

本附录为25章中需要的配置和初始化信息提供编码。该编码包含在单独的ASN.1模块中，该模块可用于25章所需要的信息的通信，或者用于存储系统的初始化数据。

```
MMS-SCI-Module-1 { iso standard 9506 part(2) mms-file-record-version1(2) }
DEFINITIONS ::= BEGIN
  IMPORTS ApplicationReference
    FROM MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1(4) }
    AccessCondition,
    AdditionalCBBOptions,
    AdditionalSupportOptions,
    Address,
    AlarmAckRule,
    Control-State,
    DomainState,
    EC-Class,
    EC-State,
    EE-Duration,
    EE-Class,
    LogicalStatus,
    Modifier,
    ParameterSupportOptions,
    PhysicalStatus,
    Priority,
    ProgramInvocationState,
    ServiceSupportOptions,
    Severity,
    Transitions,
    TypeDescription
    FROM MMS-Object-Module-1 { iso standard 9506 part(1) mms-object-model1(1) }
    AlternateAccess,
    ConfirmedServiceRequest,
    AttachToEventCondition,
    AttachToSemaphore,
    Data,
    EE-State,
    Identifier,
    Integer8,
    Integer32,
    MMSString,
```

```
    MMS255String,
    ObjectName,
    TimeOfDay,
    TypeSpecification,
    Unsigned32,
    Unsigned8
    FROM ISO-9506-MMS-1 { iso standard 9506 part(2) mms-abstract-syntax-version1(1) };
  SCI-Information ::= SEQUENCE {
    partOne [0] IMPLICIT VMD-File,
    partTwo [1] IMPLICIT Service-and-Parameter-CBBs
    }
```

B.1 SCI 第 1 部分:VMD 的初始化

包含在表 1 中的信息作为 VMD-File 类型来编码

```
  VMD-File ::= SEQUENCE {
    executiveFunction
        [0] IMPLICIT ApplicationReference,
    vendorName
        [1] MMSString,
    modelName
        [2] MMSString,
    revision
        [3] MMSString,
    abstractSyntaxes
        [4] IMPLICIT OBJECT IDENTIFIER,
    —no TRANSACTIONs,
    —no APPLICATION-ASSOCIATIONs,
    accessControl
        [5] IMPLICIT Access-Control-List-instance,
    logicalStatus
        [6] IMPLICIT LogicalStatus,
    capabilities
        [7] IMPLICIT SEQUENCE OF MMSString,
    physicalStatus
        [8] IMPLICIT PhysicalStatus,
    local-detail
        [9] IMPLICIT BIT STRING,
    accessControlLists
        [10] IMPLICIT SEQUENCE OF Access-Control-List-instance,
    domains
        [11] IMPLICIT SEQUENCE OF Domain-instance,
    programInvocations
        [12] IMPLICIT SEQUENCE OF Program-Invocation-instance,
```

```
    unitControls
        [13] IMPLICIT SEQUENCE OF Unit-Control-instance
IF (vadr)
,   unnamedVariables
        [14] IMPLICIT SEQUENCE OF Unnamed-Variable-instance
ELSE
,   unnamedVariables
        [14] IMPLICIT NULL
ENDIF
IF (vnam)
,   namedVariables
        [15] IMPLICIT SEQUENCE OF Named-Variable-instance
IF (vlis)
,   namedVariableLists
        [16] IMPLICIT SEQUENCE OF Named-Variable-List-instance
ELSE
,   namedVariableLists
        [16] IMPLICIT NULL
ENDIF
,   namedTypes
        [17] IMPLICIT SEQUENCE OF Named-Type-instance
ELSE
,   namedVariables
        [15] IMPLICIT NULL,
    namedVariableLists
        [16] IMPLICIT NULL,
    namedTypes
        [17] IMPLICIT NULL
ENDIF
,   dataExchanges
        [18] IMPLICIT SEQUENCE OF Data-Exchange-instance,
    semaphores
        [19] IMPLICIT SEQUENCE OF Semaphore-instance,
    operatorStations
        [20] IMPLICIT SEQUENCE OF Operator-Station-instance,
    eventConditions
        [21] IMPLICIT SEQUENCE OF Event-Condition-instance,
    eventActions
        [22] IMPLICIT SEQUENCE OF Event-Action-instance,
    eventEnrollments
        [23] IMPLICIT SEQUENCE OF Event-Enrollment-instance
IF (cspi)
,   eventConditionLists
```

```
        [24] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ELSE

,   eventConditionLists
        [24] IMPLICIT NULL
ENDIF
,   journals
        [25] IMPLICIT SEQUENCE OF Journal-instance,
    ...
IF (csr)
,   selected-Program-Invocation CHOICE {
        selectedProgram
        [26] IMPLICIT Program-Invocation-instance,
        noneSelected
        [27] IMPLICIT NULL }
ENDIF
    }
```

B.1.1 访问控制表对象

这个产生式对表 3 中的信息进行编码。

```
  Access-Control-List-instance ::= SEQUENCE {
    name                [0] IMPLICIT Identifier,
    definition          CHOICE {
        reference           [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl
                [3] IMPLICIT Access-Control-List-instance,
            readAccessCondition
                [4] AccessCondition OPTIONAL,
            storeAccessCondition
                [5] AccessCondition OPTIONAL,
            writeAccessCondition
                [6] AccessCondition OPTIONAL,
            loadAccessCondition
                [7] AccessCondition OPTIONAL,
            executeAccessCondition
                [8] AccessCondition OPTIONAL,
            deleteAccessCondition
                [9] AccessCondition OPTIONAL,
            editAccessCondition
                [10] AccessCondition OPTIONAL,
    —The following fields are used to record lists of objects placed
    —under the control of this ACCESS-CONTROL-LIST object.
    —They will be referred to collectively as the Controlled Object Lists
```

```
            accessControlLists
                [11] IMPLICIT SEQUENCE OF Access-Control-List-instance,
            domains
                [12] IMPLICIT SEQUENCE OF Domain-instance,
            programInvocations
                [13] IMPLICIT SEQUENCE OF Program-Invocation-instance,
            unitControls
                [14] IMPLICIT SEQUENCE OF Unit-Control-instance
IF (vadr)
,           unnamedVariables
                [15] IMPLICIT SEQUENCE OF Unnamed-Variable-instance
ELSE
,           unnamedVariables
                [15] IMPLICIT NULL
ENDIF
IF (vnam)
,            namedVariables
                [16] IMPLICIT SEQUENCE OF Named-Variable-instance
IF (vlis)
,           namedVariableLists
                [17] IMPLICIT SEQUENCE OF Named-Variable-List-instance
ELSE
,           namedVariableLists
                [17] IMPLICIT NULL
ENDIF
,           namedTypes
                [18] IMPLICIT SEQUENCE OF Named-Type-instance
ELSE
,           namedVariables
                [16] IMPLICIT NULL,
            namedVariableLists
                [17] IMPLICIT NULL,
            namedTypes
                [18] IMPLICIT NULL
ENDIF
,           dataExchanges
                [19] IMPLICIT SEQUENCE OF Data-Exchange-instance,
            semaphores
                [20] IMPLICIT SEQUENCE OF Semaphore-instance,
            operatorStations
                [21] IMPLICIT SEQUENCE OF Operator-Station-instance,
            eventConditions
```

```
                [22] IMPLICIT SEQUENCE OF Event-Condition-instance,
            eventActions
                [23] IMPLICIT SEQUENCE OF Event-Action-instance,
            eventEnrollments
                [24] IMPLICIT SEQUENCE OF Event-Enrollment-instance,
            journals
                [25] IMPLICIT SEQUENCE OF Journal-instance,
            ...
IF (cspi)
,           eventConditionLists
                [26] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ENDIF
            } } }
```

B.1.2 域对象

这个产生式对表 4 中的信息进行编码。

```
  Domain-instance ::= SEQUENCE {
    name                    [0] IMPLICIT Identifier,
    definition              CHOICE {
        reference               [1] IMPLICIT OBJECT IDENTIFIER,
        details                 [2] IMPLICIT SEQUENCE {
            capabilities            [3] IMPLICIT SEQUENCE OF MMSString,
            state                   [4] IMPLICIT DomainState,
  —The aAssociation is not included
            accessControl
                [5] IMPLICIT Access-Control-List-instance,
    sharable                        [6] IMPLICIT BOOLEAN,
    programInvocations
                [7] IMPLICIT SEQUENCE OF Program-Invocation-instance
  —uploadsInProgress is not included
IF (vnam)
,   namedVariables
                [8] IMPLICIT SEQUENCE OF Named-Variable-instance
IF (vlis)
,   namedVariableLists
                [9] IMPLICIT SEQUENCE OF Named-Variable-List-instance
ELSE
,   namedVariableLists
                [9] IMPLICIT NULL
ENDIF
,   namedTypes
                [10] IMPLICIT SEQUENCE OF Named-Type-instance
ELSE
,   namedVariables
```

```
                [8] IMPLICIT NULL,
    namedVariableLists
                [9] IMPLICIT NULL,
    namedTypes
                [10] IMPLICIT NULL
ENDIF
,   eventConditions
                [11] IMPLICIT SEQUENCE OF Event-Condition-instance,
    eventActions
                [12] IMPLICIT SEQUENCE OF Event-Action-instance,
    eventEnrollments
                [13] IMPLICIT SEQUENCE OF Event-Enrollment-instance
IF (cspi)
,   eventConditionLists
                [14] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ENDIF
            } } }
```

B.1.3 预定义程序调用对象

这个产生式对表5中的信息进行编码。

```
Program-Invocation-instance ::= SEQUENCE {
    name                    [0] IMPLICIT Identifier,
    definition              CHOICE {
        reference               [1] IMPLICIT OBJECT IDENTIFIER,
        details                 [2] IMPLICIT SEQUENCE {
            programInvocationState
                [3] IMPLICIT ProgramInvocationState,
            domains
                [4] IMPLICIT SEQUENCE OF Domain-instance,
            accessControl
                [5] IMPLICIT SEQUENCE OF Access-Control-List-instance,
            reusable
                [6] IMPLICIT BOOLEAN,
            monitor
                [7] IMPLICIT BOOLEAN,
    —The following three fields shall all be present if the value of
    —monitor is true.
    —If present, the &name field of each object instance
    —shall have a value equal to the
    —&name field of this instance of the PROGRAM-INVOCATION.
        eventCondition
                [8] IMPLICIT SEQUENCE OF Event-Condition-instance OPTIONAL,
        eventAction
                [9] IMPLICIT SEQUENCE OF Event-Action-instance OPTIONAL,
```

```
        eventEnrollment
                [10] IMPLICIT SEQUENCE OF Event-Enrollment-instance OPTIONAL,
        executionArgument
                [11] MMSString,
        ...
IF (csr)
,       control
                [12] IMPLICIT Control-State,
        controlling-Program-Invocation
                [13] IMPLICIT Program-Invocation-instance,
  —The following field shall be present
  —if and only if the value of the &control field is controlling.
        controlled-Program-Invocations
                [14] IMPLICIT SEQUENCE OF Program-Invocation-instance OPTIONAL
ENDIF
            } } }
```

B.1.4 预定义单元控制对象

这个产生式对表 6 中的信息进行编码。

```
Unit-Control-instance ::= SEQUENCE {
  name              [0] IMPLICIT Identifier,
  definition        CHOICE {
      reference         [1] IMPLICIT OBJECT IDENTIFIER,
      details           [2] IMPLICIT SEQUENCE {
          accessControl
              [3] IMPLICIT Access-Control-List-instance,
          domains
              [4] IMPLICIT SEQUENCE OF Domain-instance,
          programInvocations
              [5] IMPLICIT SEQUENCE OF Program-Invocation-instance
          } } }
```

B.1.5 无名变量对象

这个产生式对表 7 中的信息进行编码。

```
Unnamed-Variable-instance ::= SEQUENCE {
  address           [0] Address,
  accessControl     [1] IMPLICIT Access-Control-List-instance,
  typeDescription   [2] TypeDescription
  }
```

B.1.6 予定义有名变量对象

这个产生式对表 8 中的信息进行编码。

```
Named-Variable-instance ::= SEQUENCE {
  name              [0] ObjectName,
  definition        CHOICE {
      reference         [1] IMPLICIT OBJECT IDENTIFIER,
```

```
        details                 [2] IMPLICIT SEQUENCE {
            accessControl
                    [3] IMPLICIT Access-Control-List-instance,
            typeDescription             [4] TypeDescription
IF ( vadr )
,           address                     [5] Address OPTIONAL
ELSE
,                                       [5] NULL
ENDIF
IF ( sem )
,           meaning                     [6] IMPLICIT VisibleString OPTIONAL
ENDIF
        } } }
```

B.1.7 预定义有名变量表对象

这个产生式对表 9 中的信息进行编码。

```
  Named-Variable-List-instance ::= SEQUENCE {
    name                [0] ObjectName,
    definition          CHOICE {
        reference           [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl
                    [3] IMPLICIT Access-Control-List-instance,
            listOfVariables
                    [4] IMPLICIT SEQUENCE OF Variable-List-Item-instance
        } } }
  Variable-List-Item-instance ::= SEQUENCE {
  —one and only one of the following two lines shall appear
IF ( vadr )
    unnamedItem     [0] IMPLICIT Unnamed-Variable-instance OPTIONAL
ELSE
    unnamedItem     [0] IMPLICIT NULL OPTIONAL
ENDIF
IF ( vnam )
,   namedItem       [1] IMPLICIT Named-Variable-instance OPTIONAL
ELSE
,   namedItem       [1] IMPLICIT NULL OPTIONAL
ENDIF
IF (valt)
    —the following specification may be included
,   alternateAccess     [2] IMPLICIT AlternateAccess OPTIONAL
ENDIF
    }
```

B.1.8 预定义有名类型对象

这个产生式对表 10 中的信息进行编码。

```
Named-Type-instance ::= SEQUENCE {
    name                [0] ObjectName,
    definition          CHOICE {
        reference           [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl       [3] IMPLICIT Access-Control-List-instance,
            typeDescription     [4] TypeDescription
IF (sem)
,           meaning             [5] IMPLICIT VisibleString OPTIONAL
ENDIF
            } } }
```

B.1.9 预定义数据交换对象

这个产生式对表 11 中的信息进行编码。

```
Data-Exchange-instance ::= SEQUENCE {
    name                [0] IMPLICIT Identifier,
    definition          CHOICE {
        reference           [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl       [3] IMPLICIT Access-Control-List-instance,
            request             [4] IMPLICIT SEQUENCE OF TypeDescription,
            response            [5] IMPLICIT SEQUENCE OF TypeDescription,
            linked              [6] IMPLICIT BOOLEAN,
  —The following attribute shall appear if an only if the value of &linked is true.
            programInvocation   [7] IMPLICIT Program-Invocation-instance
OPTIONAL
            } } }
```

B.1.10 预定义信标对象

这个产生式对表 12 中的信息进行编码。

```
Semaphore-instance ::= SEQUENCE {
    name            [0] IMPLICIT Identifier,
    definition      CHOICE {
        reference       [1] IMPLICIT OBJECT IDENTIFIER,
        details         [2] IMPLICIT SEQUENCE {
            accessControl       [3] IMPLICIT Access-Control-List-instance,
            class               [4] IMPLICIT ENUMERATED {
                token,
                pool },
  —If the value of &class is token, the following field shall appear
            numberOfTokens      [5] IMPLICIT INTEGER OPTIONAL,
  —If the value of &class is pool, the following field shall appear
            namedTokens         [6] IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
```

```
        eventCondition          [7] IMPLICIT Event-Condition-instance
        } } }
```

B.1.11 预定义操作员站对象

这个产生式对表 13 中的信息进行编码。

```
Operator-Station-instance ::= SEQUENCE {
    name                    [0] IMPLICIT Identifier,
    definition              CHOICE {
        reference               [1] IMPLICIT OBJECT IDENTIFIER,
        details                 [2] IMPLICIT SEQUENCE {
            accessControl           [3] IMPLICIT Access-Control-List-instance,
            stationType             [4] IMPLICIT ENUMERATED {
                entry,
                display,
                entry-display }
        } } }
```

B.1.12 预定义事件条件对象

这个产生式对表 14 中的信息进行编码。

```
Event-Condition-instance ::= SEQUENCE {
    name                    [0] ObjectName,
    definition              CHOICE {
        reference               [1] IMPLICIT OBJECT IDENTIFIER,
        details                 [2] IMPLICIT SEQUENCE {
            accessControl           [3] IMPLICIT Access-Control-List-instance,
            ecClass                 [4] IMPLICIT EC-Class,
            ecState                 [5] IMPLICIT EC-State,
            priority                [6] IMPLICIT Priority,
            severity                [7] IMPLICIT Severity,
            eventEnrollments        [8] IMPLICIT SEQUENCE OF
Event-Enrollment-instance,
    —The following fields shall be present
    —if and only if the value of &ecClass is monitored.
            enabled                 [9] IMPLICIT BOOLEAN OPTIONAL,
            alarmSummaryReports     [10] IMPLICIT BOOLEAN OPTIONAL,
            monitoredVariable       CHOICE {
                named                   [11] IMPLICIT Named-Variable-instance,
                unnamed                 [12] IMPLICIT Unnamed-Variable-instance,
                unspecified             [13] IMPLICIT NULL } OPTIONAL,
            evaluationInterval      [14] IMPLICIT INTEGER OPTIONAL,
            ...
IF (cspi)
,           displayEnhancement CHOICE {
IF (des)
                text                    [15] MMSString
```

```
ENDIF
IF (dei)
,               number                          [16] IMPLICIT INTEGER
ENDIF
,               none                            [17] IMPLICIT NULL
                },
            group-Priority-Override     CHOICE {
                priority                        [18] IMPLICIT Priority,
                undefined                       [19] IMPLICIT NULL
                } OPTIONAL,
            referencingEventConditionLists
                        [20] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ENDIF
            } } }
```

B.1.13 预定义事件活动对象

这个产生式对表 15 中的信息进行编码。

```
Event-Action-instance ::= SEQUENCE {
    name                [0] ObjectName,
    definition          CHOICE {
        reference           [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl               [3] IMPLICIT Access-Control-List-instance,
            confirmedServiceRequest     [4] ConfirmedServiceRequest,
            modifiers                   [5] IMPLICIT SEQUENCE OF Modifier,
            eventEnrollments            [6] IMPLICIT SEQUENCE OF
Event-Enrollment-instance
            } } }
```

B.1.14 预定义事件登录对象

这个产生式对表 16 中的信息进行编码。

```
Event-Enrollment-instance ::= SEQUENCE {
    name                [0] ObjectName,
    definition          CHOICE {
        reference           [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl               [3] IMPLICIT Access-Control-List-instance,
            eeClass                     [4] IMPLICIT EE-Class ,
            eventCondition              [5] IMPLICIT Event-Condition-instance,
            ecTransitions               [6] IMPLICIT Transitions,
—The following parameter is present if and only if the
—value of &eeClass is modifier.
            remainingDelay              CHOICE {
                time                        [7] IMPLICIT INTEGER,
                forever                     [8] IMPLICIT NULL } OPTIONAL,
```

```
  —The remaining parameters are present if and only if the
  —value of &eeClass is notification.
            eventAction                 [9] IMPLICIT Event-Action-instance OPTIONAL,
            duration                    [10] IMPLICIT EE-Duration OPTIONAL,
            clientApplication           [11] IMPLICIT ApplicationReference OPTIONAL,
            aaRule                      [12] IMPLICIT AlarmAckRule OPTIONAL,
            ...
IF (cspi)
,           displayEnhancement          CHOICE {
IF (des)
              text                          [13] MMSString
ENDIF
IF (dei)
,             number                        [14] IMPLICIT INTEGER
ENDIF
,             none                          [15] IMPLICIT NULL
             }
ENDIF
          } } }
```

B.1.15 预定义事件条件表对象

这个产生式对表 17 中的信息进行编码。

```
  Event-Condition-List-instance ::= SEQUENCE {
    name                        [0] ObjectName,
    definition                  CHOICE {
        reference                   [1] IMPLICIT OBJECT IDENTIFIER,
        details                     [2] IMPLICIT SEQUENCE {
            accessControl
                [3] IMPLICIT Access-Control-List-instance,
            eventConditions
                [4] IMPLICIT SEQUENCE OF Event-Condition-instance
IF (recl)
,           eventConditionLists
                [5] IMPLICIT SEQUENCE OF Event-Condition-List-instance,
            referencingEventConditionLists
                [6] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ENDIF
          } } }
```

B.1.16 预定义日志对象

这个产生式对表 18 中的信息进行编码。

```
  Journal-instance ::= SEQUENCE {
    name                        [0] ObjectName,
    definition                  CHOICE {
        reference                   [1] IMPLICIT OBJECT IDENTIFIER,
```

```
        details                         [2] IMPLICIT SEQUENCE {
            accessControl                   [3] IMPLICIT Access-Control-List-instance,
            entries                         [4] IMPLICIT SEQUENCE OF
Journal-Entry-instance
            } } }
  Journal-Entry-instance ::= SEQUENCE {
    journal                 [0] IMPLICIT Journal-instance,
    entry                   [1] IMPLICIT OCTET STRING,
    clientApplication       [2] IMPLICIT ApplicationReference,
    timeStamp               [3] IMPLICIT TimeOfDay,
    orderOfReceipt          [4] IMPLICIT INTEGER,
    informationType         [5] IMPLICIT ENUMERATED {
        annotation,
        event-data,
        data },
  —The following attribute shall appear if and only if the
  —value of &informationType is annotation.
    textComment             [6] MMS255String OPTIONAL,
  — The following attribute shall appear if and only if the
  —value of &informationType is event-data.
    eventTransitionRecord   [7] IMPLICIT SEQUENCE {
        name                        [8] ObjectName,
        currentState                [9] IMPLICIT EC-State
        } OPTIONAL,
  —The following attribute shall appear if and only if the
  —value of &informationType is data or event-data.
    journalVariables        [10] IMPLICIT SEQUENCE OF SEQUENCE {
            variableTag                 [11] MMS255String,
            valueSpecification          [12] Data
            } OPTIONAL
    }
```

B.2 服务和参数 CBBs

包含在 SCI 第 2 部分表 20 到表 52 中的信息由 Service-and-paramenter-CBBs 类型规定。

```
Service-and-Parameter-CBBs ::= SEQUENCE {
    services-Client             [0] IMPLICIT ServiceSupportOptions,
    services-Server             [1] IMPLICIT ServiceSupportOptions,
    parameters                  [2] IMPLICIT ParameterSupportOptions,
    nest                        [3] IMPLICIT INTEGER
IF (csr cspi)
,   extendedServices-Client     [4] IMPLICIT AdditionalSupportOptions,
    extendedServices-Server     [5] IMPLICIT AdditionalSupportOptions
ELSE
```

```
,   extendedServices-Client          [4] IMPLICIT NULL,
    extendedServices-Server          [5] IMPLICIT NULL
ENDIF
IF (cspi)
,   extendedParameters               [6] IMPLICIT AdditionalCBBOptions
ELSE
,   extendedParameters               [6] IMPLICIT NULL
ENDIF
,   generalManagement                [7] IMPLICIT GeneralManagementParameters,
    vMDSupport                       [8] IMPLICIT VMDSupportParameters,
    domainManagement                 [9] IMPLICIT DomainManagementParameters,
    programInvocation                [10] IMPLICIT ProgramInvocationManagementParameters,
    variableAccess                   [11] IMPLICIT VariableAccessParameters,
    dataParameters                   [12] IMPLICIT DataParameters,
    semaphoreManagement              [13] IMPLICIT SemaphoreManagementParameters,
    operatorCommunication            [14] IMPLICIT OperatorCommunicationParameters,
    errors                           [15] IMPLICIT ErrorParameters,
    fileManagement                   [16] IMPLICIT FileManagementParameters
    }
```

B.2.1 环境和通用管理参数

这个产生式对表 21 中的信息进行编码。

```
GeneralManagementParameters ::= SEQUENCE {
    localDetail                      [0] MMSString,
    supportForTime                   [1] IMPLICIT SEQUENCE {
      timeOfDay                      [2] IMPLICIT BOOLEAN,
      timeSequence                   [3] IMPLICIT BOOLEAN
    },
    granularityOfTime                [4] IMPLICIT INTEGER
    }
```

B.2.2 VMD 支持参数

这个产生式对表 25 中的信息进行编码。

```
VMDSupportParameters ::= SEQUENCE {
    localDetail              [0] MMSString,
    extendedDerivation       [1] MMSString
—method used to perform extended derivation
    }
```

B.2.3 域管理参数

这个产生式对表 27 中的信息进行编码。

```
DomainManagementParameters ::= SEQUENCE {
    loadDataOctet                  [0] MMSString,
—description of the format of Load Data if the octet string form is used
    loadDataSyntax                 [1] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER,
—identifier of the Abstract Syntaxes used
```

```
    maxUploads                    [2] IMPLICIT INTEGER
    }
```

B.2.4 程序调用管理参数

这个产生式对表 28 中的信息进行编码。

```
  ProgramInvocationManagementParameters ::= SEQUENCE {
    executionArgMaxSize                [0] IMPLICIT INTEGER,
    executionArgParseRules             [1] MMSString,
    executionArgSyntaxes               [2] IMPLICIT SEQUENCE OF OBJECT
    IDENTIFIER,
    programLoction                     [3] MMSString,
  —syntax of the program Location notation
    stepMode                           [4] IMPLICIT BOOLEAN
  —if true, step Mode is supported
    }
```

B.2.5 变量访问参数

这个产生式对表 32 中的信息进行编码。

```
  VariableAccessParameters ::= SEQUENCE {
    uninterruptibleAccess          [0] MMSString,
          —conditions under which it is guaranteed
    singleMode                     [1] IMPLICIT BOOLEAN,
    unnamedMode                    [2] IMPLICIT BOOLEAN
    }
```

B.2.6 变量访问参数

这个产生式对表 33 中的信息进行编码。

```
  DataParameters ::= SEQUENCE {
    bit-string          [0] IMPLICIT INTEGER OPTIONAL,
    integer             [1] IMPLICIT INTEGER OPTIONAL,
    unsigned            [2] IMPLICIT INTEGER OPTIONAL,
    floating-point      [3] IMPLICIT SEQUENCE {
        total           [4] IMPLICIT INTEGER,
        exponent        [5] IMPLICIT INTEGER } OPTIONAL,
    octet-string        [10] IMPLICIT INTEGER OPTIONAL,
    visible-string      [11] IMPLICIT INTEGER OPTIONAL,
    binary-time         [12] IMPLICIT BOOLEAN OPTIONAL,
    bcd                 [13] IMPLICIT INTEGER OPTIONAL,
    mmsString           [14] IMPLICIT INTEGER OPTIONAL
    }
```

B.2.7 信标管理参数

这个产生式对表 36 中的信息进行编码。

```
  SemaphoreManagementParameters ::= SEQUENCE {
    algorithm                    [0] IMPLICIT MMSString
        —method of processing the &priority field
    }
```

B.2.8 操作员通信参数

这个产生式对表 37 中的信息进行编码。

```
OperatorCommunicationParameters ::= SEQUENCE {
    input-time-out              [0] IMPLICIT INTEGER
    }
```

B.2.9 错误参数

这个产生式对表 47 中的信息进行编码。

```
ErrorParameters ::= SEQUENCE {
    additionalCode              [0] MMSString,
    additionalDetial            [1] IMPLICIT SEQUENCE {
        size                        [2] IMPLICIT INTEGER,
        syntax                      [3] MMSString
        }
    }
```

B.2.10 文件管理参数

这个产生式对表 50 中的信息进行编码。

```
FileManagementParameters ::= SEQUENCE {
    fileName                    [0] MMSString
    }
END
```

附 录 C
（规范性附录）
文件访问协议

C.1 引言

本章描述GB/T 16720.1(MMS服务定义)附录B(文件访问服务)所描述的服务的服务专用协议元素所描述的只有一个服务——ObtainFile服务。下列行用于介绍本附录及附录D、E中定义的模块。

```
ISO-9506-MMS-1A { iso standard 9506 part(2) mms-annex-version1(3) }
DEFINITIONS ::= BEGIN
EXPORTS
    ObtainFile-Request,
    ObtainFile-Response,
    ObtainFile-Error,
    FileOpen-Request,
    FileOpen-Response,
    FileRead-Request,
    FileRead-Response,
    FileClose-Request,
    FileClose-Response,
    FileRename-Request,
    FileRename-Response,
    FileRename-Error,
    FileDelete-Request,
    FileDelete-Response,
    FileDirectory-Request,
    FileDirectory-Response,
    ScatteredAccessDescription,
    DefineScatteredAccess-Request,
    DefineScatteredAccess-Response,
    GetScatteredAccessAttributes-Request,
    GetScatteredAccessAttributes-Response;
IMPORTS
    FileName,
    ObjectName,
    AlternateAccess,
    VariableSpecification,
    Identifier,
    Integer32,
    Unsigned32 FROM
    ISO-9506-MMS-1 { iso standard 9506 part(2) mms-abstract-syntax-version1(1) }
    ApplicationReference FROM
```

MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1 (4) };

C.2 ObtainFile(获得文件)

ConfirmedServiceRequest 类型、ServiceResponse 类型和 ServiceError 类型的 obtainFile 选择的抽象语法规定如下,其描述在随后的段落中给出。对于本章中未提供显式导出的所有参数,它们的导出在 5.5 中描述。

```
  ObtainFile-Request ::= SEQUENCE {
IF ( tpy )
    sourceFileServer [0] IMPLICIT ApplicationReference OPTIONAL,
ENDIF
    SourceFile              [1] IMPLICIT FileName,
    DestinationFile         [2] IMPLICIT FileName
    }
  ObtainFile-Response ::= NULL
  ObtainFile-Error ::= INTEGER {
    source-file             (0),
    destination-file        (1)
    } (0..1)
```

C.2.1 ObtainFile - Request

ConfirmedServiceRequest 类型的 obtainFile 选择的抽象语法是 ObtainFile - Request。

C.2.2 ObtainFile - Response

ConfirmedServiceResponse 类型的 obtainFile 选择的抽象语法是 ObtainFile - Response。

C.2.3 ObtainFile - Error

ConfirmedServiceError 类型的 serviceSpecificInformation 选择的 obtainFile 选择的抽象语法是 ObtainFile-Error,它是 ObtainFile. response 原语的 Result(—)参数的 File In Error 子参数,并且,作为 Obtain File. confirm 原语(如果发送的话)的 Result(—)参数的 File In Error 子参数出现。

附 录 D
（资料性附录）
文件管理协议

D.1 引言

本章描述 GB/T 16720.1 附录 C（文件管理服务）定义的服务的服务专用协议元素。这些服务包括：

FileOpen　　FileRename
FileRead　　FileDelete
FileClose　　FileDirectory

D.2 FileOpen（打开文件）

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 fileOpen 选择的抽象语法规定如下，其描述在随后的两节中给出。对于本章中未提供显式导出的所有参数，它们的导出在 5.5 中描述。

```
FileOpen-Request ::= SEQUENCE {
    fileName            [0] IMPLICIT FileName,
    initialPosition     [1] IMPLICIT Unsigned32 }
FileOpen-Response ::= SEQUENCE {
    frsmID              [0] IMPLICIT Integer32,
    fileAttributes      [1] IMPLICIT FileAttributes }
```

D.2.1 FileOpen-Request

ConfirmedServiceRequest 类型的 fileOpen 选择的抽象语法是 FileOpen-Request。

D.2.2 FileOpen-Response

ConfirmedServiceResponse 类型的 fileOpen 选择的抽象语法是 FileOpen-Response。

D.3 FileRead（读文件）

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 fileRead 选择的抽象语法规定如下，其描述在随后的两节中给出。对于本章中未提供显式导出的所有参数，它们的导出在 5.5 中描述。

```
FileRead-Request ::= Integer32--FRSM ID
FileRead-Response ::= SEQUENCE {
    fileData            [0] IMPLICIT OCTET STRING,
    moreFollows         [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

D.3.1 FileRead-Request

ConfirmedServiceRequest 类型的 fileRead 选择的抽象语法是 FileRead-Request。它是取自 FileRead.request 原语的 FRSM ID 参数，并作为 FileRead.indication 原语的 FRSM ID 参数出现。

D.3.2 FileRead-Response

ConfirmedServiceResponse 类型的 fileRead 选择的抽象语法是 FileRead-Response。

注：尽管本附录不是标准的，但为了阐明正确操作本附录中的协议的要求，仍使用规范语言而不是建议使用。

D.4 FileClose(关闭文件)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 close File 选择的抽象语法规定如下,其描述在随后的两节中给出。对于本章中未提供显式导出的所有参数,它们的导出在 5.5 中描述。

```
FileClose-Request ::= Integer32—FRSM ID
FileClose-Response ::= NULL
```

D.4.1 FileClose - Request

ConfirmedServiceRequest 类型的 fileClose 选择的抽象语法是 FileClose-Request。它是取自 FileClose.request 原语的 FRSM ID 参数,并作为 FileClose.Indication 原语的 FRSM ID 参数出现。

D.4.2 FileClose - Response

ConfirmedServiceResponse 类型的 fileClose 选择的抽象语法是 FileClose-Response,它是 NULL。它由 FileClose.request 原语中的 Result(+)参数指明,并作为 FileClose.confirm 服务原语中的 Result(+)参数出现。

D.5 FileRename(文件改名)

ConfirmedServiceRequest 类型、ConfirmedServiceResponse 类型和 ConfirmedServiceError 类型的 fileRename 选择的抽象语法规定如下,其描述在随后的两节中给出。对于本章中未提供显式导出的所有参数,它们的导出在 5.5 中描述。

```
FileRename-Request ::= SEQUENCE {
    currentFileName          [0] IMPLICIT FileName,
    newFileName              [1] IMPLICIT FileName }
FileRename-Response ::= NULL
FileRename-Error ::= INTEGER {
    source-file              (0),
    destination-file         (1)
    } (0..1)
```

D.5.1 FileRename - Request

ConfirmedServiceRequest 类型的 fileRename 选择的抽象语法是 FileRename-Request。

D.5.2 FileRename - Response

ConfirmedServiceResponse 类型的 fileRename 选择的抽象语法是 FileRename-Response,它由 FileRename.response 服务原语中的 Result(+)参数指明,并作为 FileRename.confirm 服务原语中的 Result(+)参数出现。

D.5.3 FileRename - Error

ConfirmedServiceError 类型的 fileRename 选择的抽象语法是 FileRename-Error,它是 File Rename.response 原语的 Result(-)参数的 FileIn Error 子参数,它将作为 FileRename.confirm 原语(如果发送的话)的 Result(-)参数的 FileIn Error 子参数出现。

D.6 FileDelete(删除文件)

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 fileDelete 选择的抽象语法规定如下,其描述在随后的两节中给出。对于本章中未提供显式导出的所有参数,它们的导出在 5.5 中描述。

```
FileDelete-Request ::= FileName
```

```
FileDelete-Response ::= NULL
```

D.6.1 FileDelete - Request

ConfirmedServiceReques 类型的 fileDelete 选择的抽象语法是 FileDelete-Request。它是取自 FileDelete. request 原语的 File Name 参数，并作为 FileDelete. indication 原语的 File Name 参数出现。

D.6.2 FileDelete - Response

ConfirmedServiceResponse 类型的 fileDelete 选择的抽象语法是 FileDelete-Response。它由 FileDelete. response 服务原语中的 Result(+)参数指明，并作为 FileDelete. confirm 服务原语中的 Result(+)参数出现。

D.7 FileDirectory(文件目录)

ConfirmedServiceReques 类型和 ConfirmedServiceResponse 类型的 fileDirectory 选择的抽象语法规定如下，其描述在随后的两节中给出。对于本章中未提供显式导出的所有参数，它们的导出在 5.5 中描述。

```
FileDirectory-Request ::= SEQUENCE {
    fileSpecification           [0] IMPLICIT FileName OPTIONAL,
    continueAfter               [1] IMPLICIT FileName OPTIONAL }
FileDirectory-Response ::= SEQUENCE {
    listOfDirectoryEntry        [0] SEQUENCE OF DirectoryEntry,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT FALSE }
DirectoryEntry ::= SEQUENCE {
    fileName                    [0] IMPLICIT FileName,
    fileAttributes              [1] IMPLICIT FileAttributes }
```

D.7.1 FileDirectory-Request

ConfirmedServiceReques 类型的 fileDirectory 选择的抽象语法是 FileDirectory-Request。

D.7.2 FileDirectory-Response

ConfirmedServiceResponse 类型的 fileDirectory 选择的抽象语法是 FileDirectory-Response。

D.7.2.1 ListOfDirectoryEntry

ListOfDirectoryEntry 字段是 FileDirectory. response 服务原语的 ListOfDirectoryEntry 参数，并作为 FileDirectory. confirm 原语的 ListOfDirectEntry 参数出现。该字段应包含 0 个或多个 DirectoryEntry 类型的值，而每一个(按表中的顺序)又包含 ListOfDirectoryEntry 参数的一个 DirectoryEntry 子参数之值。ListOfDirectoryEntry 参数的 Directory Entry 子参数的每一个值将利用 5.5，以便导出 listOfDirectoryEntry 序列的相应元素。

D.8 FileAttributes(文件属性)

FileAttributes 参数的抽象语法规定如下：

```
FileAttributes ::= SEQUENCE {
    sizeOfFile          [0] IMPLICIT Unsigned32,—in octets
    lastModified        [1] IMPLICIT GeneralizedTime OPTIONAL }
```

附 录 E
（资料性附录）
分 散 访 问

E.1 引言

下列特性已在 GB/T 16720—1996 中给出，尽管本附录是提示性的，但仍使用规范语言来重述第一版中的一些正文。

本章描述支持分散访问所必须的服务的服务专用协议元素，其中包括实现下列服务所需要的协议。

DefineScatteredAccess

GetScatteredAccessAttributes

E.1.1 规定变量访问的协议

E.1.2 VariableSpecification

分散访问对象的存在更改了 VariableSpecification 产生式（参见 14.5.2）。除了已存在的三种选择之外，增加了第四种选择：ScatteredAccessDescription。

E.1.3 ScatteredAccessDescription

ScatteredAccessDescription 参数的抽象语法规定如下，对于本章中未提供显式导出的所有参数，它们的导出在 5.5 中描述。

```
ScatteredAccessDescription ::=SEQUENCE OF SEQUENCE {
    componentName           [0] IMPLICIT Identifier OPTIONAL,
    variableSpecification   [1] VariableSpecification
IF ( valt )
,   alternateAccess         [2] IMPLICIT AlternateAccess OPTIONAL
ENDIF
    }
```

E.2 DefineScatteredAccess（定义分散访问）

ConfirmedServiceReques 类型和 ConfirmedServiceResponse 类型的 defineScatteredAccess 选择的抽象语法规定如下，其描述在随后的两节中给出。对于本章中未提供显式导出的所有参数，它们的导出在 5.5 中描述。

```
DefineScatteredAccess-Request ::= SEQUENCE {
    scatteredAccessName             [0] ObjectName,
    scatteredAccessDescription      [1] IMPLICIT ScatteredAccessDescription }
DefineScatteredAccess-Response ::= NULL
```

E.2.1 DefineScatteredAccess-Request

ConfirmedServiceRequest 类型的 defineScatteredAccess 选择的抽象语法是 DefineScatteredAccess-Request 类型。

E.2.2 DefineScatteredAccess-Response

ConfirmedServiceResponse 类型的 defineScatteredAccess 选择的抽象语法是 DefineScatteredAccess-Response 类型，它应是 NULL。

E.3 GetScatteredAccessAttributes（获取分散访问属性）

ConfirmedServiceRequest 类型和 ConfirmedServiceResponse 类型的 getScatteredAccessAttributes

选择的抽象语法规定如下，其描述在随后的两节中给出。

```
GetScatteredAccessAttributes-Request ::= ObjectName - ScatteredAccessName

GetScatteredAccessAttributes-Response ::= SEQUENCE {
    mmsDeletable                    [0] IMPLICIT BOOLEAN,
    scatteredAccessDescription      [1] IMPLICIT ScatteredAccessDescription
IF ( aco )
,   accessControlList               [2] IMPLICIT Identifier OPTIONAL
                -—Shall not appear in minor version one or two
ENDIF
    }
```

E.3.1 GetScatteredAccessAttributes-Request

ConfirmedServiceRequest 类型的 getScatteredAccessAttributes 选择的抽象语法是 GetScatteredAccessAttributes-Request 类型。

E.3.2 GetScatteredAccessAttributes-Response

ConfirmedServiceResponse 类型的 getScatteredAccessAttributes 选择的抽象语法是 GetScatteredAccessAttributes-Response 类型。

E.3.2.1 访问控制表

当且仅当 aco CBB 已经商定，accessControlList 参数才出现。

附 录 F
（资料性附录）
实数据类型

F.1 引言

下列特性虽已在于 GB/T 16720—1996 第一版中给出，但现在仍反对使用它们。在这里重述这些特性完全是为了历史上的一致性。尽管本附录是提示性的，但仍使用规范语言来重述第一版中的一些正文。

F.2 REAL(实)数据

通常在 Type Description 参数和 Data 参数中支持实数据类型。该类型引用 ISO/IEC 8824-1 中定义的 REAL 数据类型。

F.3 模块结束

以下列 END 语句结束附录 C 中开始的模块。

END

附 录 G
（资料性附录）
中英文对照表

Access Result …… 访问结果
AcknowledgeEventNotification …… 认可事件通告
AddEventConditionListReference …… 增加事件条件表引用
AdditionalService …… 附加服务
AdditionalService-Error …… 附加服务错误
Address …… 地址
AddToUnitControl …… 加入单元控制
AE-qualifier …… 应用实体-资格
AlarmAckRule …… 报警认可规则
AlarmEnrollmentSummary …… 报警登录汇总
AlarmSummary …… 报警汇总
AlterEventConditionlistMonitoring …… 变更事件条件监控表
AlterEventConditionMonitoring …… 变更事件条件监控
AlterEventEnrollment …… 变更事件登录
Alternate Access …… 替代访问
Alternate Access Selection …… 替代访问选择
AlterProgramInvocationAttributes …… 变更程序调用属性
Application Reference …… 应用引用
AP-title …… 应用过程-题目
Attach To Semaphore …… 附加信标
Cancel-ErrorPDU …… 取消-错误 PDU
Cancel-RequestPDU …… 取消-请求 PDU
Cancel-ResponsePDU …… 取消-应答 PDU
ChangeAccessControl …… 修改访问控制
Conclude-ErrorPDU …… 结束-错误 PDU
Conclude-RequestPDU …… 结束-请求 PDU
Conclude-ResponsePDU …… 结束-应答 PDU
Conditioned service Request …… 条件服务请求
Conditioned service Response …… 条件服务应答
Configuration …… 配置
Contents …… 内容
Control Element …… 控制元素
CreateJournal …… 建立日志
CreateProgramInvocation …… 建立程序调用
CreateUnitControl …… 建立单元控制
CS-EventNotification …… 确认服务-事件通告
CS-Response-Detail …… 确认服务-应答细目
Data …… 数据

Data Access Error …… 数据访问错误
DefineAccessControlList …… 定义访问控制表
DefineEventAction …… 定义事件活动
DefineEventCondition …… 定义事件条件
DefineEventEnrollment …… 定义事件登录
DefineNamedType …… 定义有名类型
DefineNamedVariable …… 定义有名变量
DefineNamedVariableList …… 定义有名变量表
DefineScatteredAccess …… 定义分散访问
DefineSemaphore …… 定义信标
DeleteAccessControlList …… 删除访问控制表
DeleteDomain …… 删除域
DeleteEventAction …… 删除事件活动
DeleteEventCondition …… 删除事件条件
DeleteEventEnrollment …… 删除事件登录
DeleteJournal …… 删除日志
DeleteNamedType …… 删除有名类型
DeleteNamedVariableList …… 删除有名变量表
DeleteProgramInvocation …… 删除程序调用
DeleteSemaphore …… 删除信标
DeleteUnitControl …… 删除单元控制
DeleteVariableAccess …… 删除变量访问
DirectoryEntry …… 目录项
DomainState …… 域状态
DownloadSegment …… 下载段
EC-Class …… EC-类别
EC-State …… EC-状态
EE-Class …… EE-类别
EE-State …… EE-状态
EntryContent …… 项内容
EN-Additional-Detail …… EN-补充细目
Event Enrollment …… 事件登录
ExchangeData …… 交换数据
File Attributes …… 文件属性
FileClose …… 关闭文件
FileDelete …… 删除文件
FileDirectory …… 文件目录
FileName …… 文件名
FileOpen …… 打开文件
FileRead …… 读文件
FileRename …… 文件更名
FileRename-Error …… 文件改名-错误
Floating Point …… 浮点

StopUnitControl …… 停止单元控制
StoreDomainContent …… 存贮域内容
StoreUnitControlToFile …… 向文件存入单元控制
TakeControl …… 取得控制
TerminateDownloadSequence …… 终止下载序列
TerminateUploadSequence …… 终止上载序列
Time …… 时间
TimeOfday …… 当日时间
Transition …… 转换
TriggerEvent …… 触发事件
Type Specification …… 类型说明
UnconfirmedService …… 无确认服务
Unconfirmed-PDU …… 无确认 PDU
UnitControlLoadSegment …… 单元控制装载段
UnitControlUpload …… 单元控制上载
Unsigned16 …… 16 位无符号
Unsigned32 …… 32 位无符号
Unsigned8 …… 8 位无符号
UnsolicitedStatus …… 非请求状态
UploadSegment …… 上载段
Variable Access Specification …… 变量访问说明
Variable Specification …… 变量说明
VMDReset …… VMD 复位
VMDStop …… VMD 停止
Write …… 写
WriteJournal …… 写日志