PROIBUS 规范-标准部分

第五部分：应用层服务定义

PROFIBUS Specification-Normative Parts

Part5:Application Layer Service Definition

第五部分 应用层服务定义

目录 **页码**

1 范围

本部分规范规定应用层协议、应用层接口以及相应的网络管理，并按 PROFIBUS 数据链路层的要求映象到位串行 PROFIBUS 系统的数据链路层。

PROFIBUS 致力于与过程密切相关的自动化应用并制定一个具有实时操作能力的简单总线接口。本规范能使不同厂家生产的现场自动化部件在分布式系统中互连，并保证部件间通信可靠。这样的系统称之为“开放系统”。此外，PROFIBUS 协议易于与较高层次的自动化系统集成（制造自动化协议，Manufacturing Automation Protocol，MAP），只需极少的互连开销。

由于本规范具有一定的自由度，即灵活性，通过对不同的应用规定专门的行规文件（面向应用的功能标准），如楼宇自动化、离散部件制造、过程控制等等，使得在不同的应用领域可以实现不同的系统组态和功能结构。

2 引用标准和附加资料

ISO 7498/1:1989	信息处理系统；开放系统互连；基本参考模型
ISO 7498/4:1989	信息处理系统；开放系统互连；基本参考模型；第四部分：管理的结构
ISO 8824:1987	信息处理系统；开放系统互连；抽象句法表达法 1 的规定（ASN.1）
ISO 8825:1987	信息处理系统；开放系统互连；抽象句法表达法 1（ASN.1）基本编码原则的规定
ISO TR 8509:1987	信息处理系统；开放系统互连--服务约定
IEEE 754:1985	IEEE 二进制浮点算法标准

3 概述

3.1 术语及缩略语

由于技术术语的多义性和关联到现有的国际总线标准的需要，因此有必要将技术术语限定在有确定定义的范围内。

缩略语	含义
.con	确认原语
.ind	指示原语
.req	请求原语
.res	响应原语
ABO	中止（abort）或 FMA7 中止
ACI	非循环的控制间隔
ACK	应答的
acyc	非循环的
AD	附加细节

ALI	应用层接口
ASIC	专用集成电路
ASN.1	抽象句法表达法 1
ASS	LLI 的有关服务
BRCT	广播通信关系
C	条件
CCI	循环控制间隔
CI	控制间隔
CN	连接
CON	确认
CREF	通信引用
CREL	通信关系
CRL	通信关系表
CRL Header	通信关系表首部
CSRD	循环地发送和请求数据需回答
“D”	已定义的连接
DIN	德国工业标准
DIS	国际标准草案
DP-OD	程序调用动态表（对象字典）
DS	未连接的站，本地 FDL/PHY 控制器不在逻辑令牌环中或从线上断开
DSAP	目的服务存取点
DTA	需应答的数据传送
DTC	需确认的数据传送
DTU	无需确认的数据传送
DV-OD	变量表的动态表（对象字典）
FC’	功能码
FDL	现场总线数据链路，第 2 层
FER	现场总线编码规则
FMA	现场总线管理
FMA1/2	现场总线管理层 1/2
FMA7	现场总线管理层 7
FMS	现场总线报文规范
FR+	最终响应
GAP	在逻辑令牌环中，从本站（TS）到下一站（NS）之间的站地址范围，不得超过最高站地址（HSA）
GAPL	GAP 表，包括在本站的 GAP 中所有站的状态
HSA	最高站地址（FDL 地址）
HW	硬件
“I”	在请求方的开放连接

ID	标识符
IDM	映象数据存储器
IEEE	电子和电气工程师协会（美国）
IL	标识表
IMA	被激活的空闲机
IND	指示
INFO	信息报告
INI	启动
IS	国际标准
ISO	国际标准化组织
IV	请求中有无效参数
IVID	调用 ID（INVOKE ID）
kbit/s	每秒千位（数据传输速率）
L_sdu	链路服务数据单元
LAS	主动站表
LG	本地生成的
LL	活动表（Live List）
LLI	低层接口
Loc_add	本地地址
LR	不可用的或不充分的本地资源
LS	在服务存取点本地服务未激活或链路服务存取点（LSAP）未激活
LSAP	链路服务存取点
LSB	最低有效位
M	强制性的
MAP	制造自动化协议
MMAC	用于非循环数据传送的‘主-主’连接
MMS	制造报文规范
MSAC	用于非循环数据传送的，非从站发起的‘主-从’连接
MSAC_SI	用于非循环数据传送的，由从站发起的‘主-从’连接
MSB	最高有效位
MSCY	用于循环数据传送的，非从站发起的‘主-从’连接
MSCY_SI	用于循环数据传送的，由从站发起的‘主-从’连接
MULT	群播通信关系
NA	无应答/响应
NE	不存在的
NIL	本地存在的值，但本规范不予确定
NOK	不行 （Not OK）
“O”	响应方的开放连接

OD	对象字典
OD-ODES	对象字典中的对象描述
OSCC	客户机未完成的服务计数器
OSCS	服务器未完成的服务计数器
PDU	协议数据单元
PEE	启用的轮询登入项
PHY	物理层
PI	程序调用
PICS	协议实现一致性声明
R+	结果（+）
R-	结果（-）
RAC	接收应答请求计数器
RC	原因代码
RCC	接收确认请求计数器
RDH	响应 FDL 高及无资源发送数据
RDL	响应 FDL 低及无资源发送数据
Rem_add	远程地址
REQ	请求
RES	响应
ROM	只读存储器
RR	无资源发送数据及无响应 FDL 数据（应答否定）
RS	在远程服务存取点无服务或 Rem_add 未激活（应答否定）
RSAP	远程服务存取点
RSV	保留
RTimer	接收定时器
RVR	远程读值
S	选择
S-OD	静态对象表
SAC	发送应答请求计数器
SAP	服务存取点
SC	状态冲突
SCC	发送确认请求计数器
SDA	发送数据需应答
SDN	发送数据无需应答
SI	从站启动
SN	符号
SRD	发送和请求数据需回答
SSAP	源服务存取点
ST-OD	对象字典中的静态类型表

Std	标准
STimer	发送定时器
SU	夏时制（标准时间/夏时制）
SW Release	软件发行版
T1	定时器 1
T_{QUI}	静止时间，即发送器降落时间（传输线状态的不确定时间）和/或中继器开关时间；发送站在启用它的接收器之前在帧结束后所必须等待的时间
T_{RR}	实际轮转时间
T_S	本站
T_{SDR}	响应方的站延迟，即该响应方在产生回答帧之前实际等待的时间
T_{SET}	建立时间，即在一个事件（例如中断 SYN 定时器到期）到必须作出反应（例如允许接收器接收数据）所需的时间
T_{SL}	时隙时间，在 PROFIBUS 系统里一个主站必须等待事务处理响应所需的最长等待时间。
T_{TR}	目标轮转时间，预期一个令牌在 PROFIBUS 系统中轮转一周所需时间，其中包括用于高低优先权事务处理、错误和 GAP 维护所需时间
U	用户选项
UE	否定的应答，远程用户接口出错
VFD	虚拟现场设备

3.2 在 ISO/OSI 分层模型中的结构和布局

PROFIBUS 的结构基于 ISO/OSI 开放通信模型（ISO 7498）。PROFIBUS 规范使用第 1 层（物理层，PHY），第 2 层（数据链路层，FDL）和第 7 层（应用层）。第 3 层到第 6 层未使用，以尽量减少系统开销并提高效率。

在 PROFIBUS 规范的前几部分中，描述了物理层（第 1 层），数据链路层（第 2 层）及相应的管理（FMA1/2）。本部分描述应用层（FMS，LLI）及相应的管理（FMA7）。

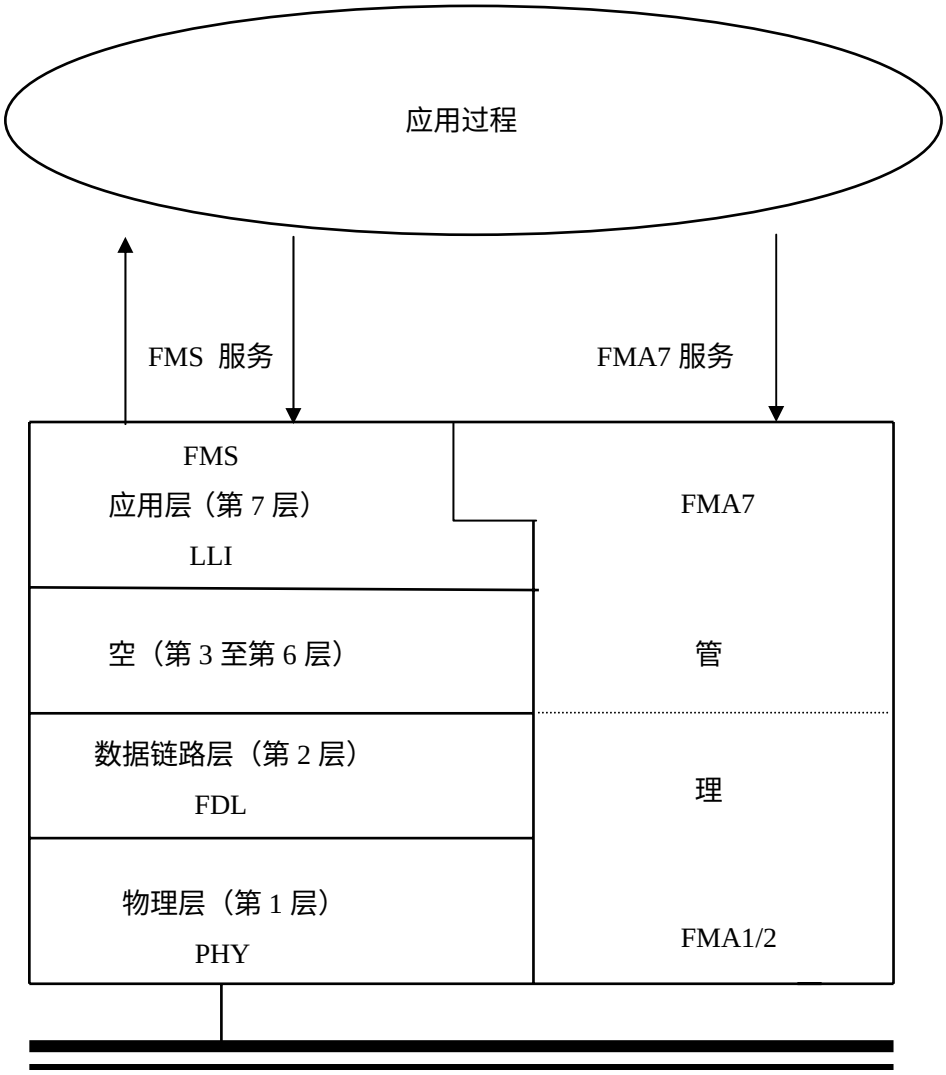


图 1. 在 ISO/OSI 分层模型中的布局

3.2.1 应用层

PROFIBUS 规范的应用层由两个实体，即 FMS（现场总线报文规范）和 LLI（低层接口）组成。

3.2.1.1 现场总线报文规范（FMS）

FMS 描述通信对象、服务、并从通信伙伴的角度（服务器行为）描述相关的模型。

3.2.1.2 低层接口（LLI）

PROFIBUS 的各种特性要求在 FDL 和 FMS/FMA7 之间有一个特殊的匹配。通过 LLI 实现这种匹配。LLI 是第 7 层的一个实体。

LLI 的主要任务是：

— 将 FMS 和 FMA7 服务映象到 FDL 服务
— 连接的建立和释放
— 连接的管理
— 数据流控制

3.2.2 现场总线管理层 7（FMA7）

FMA7 基于 ISO DIS 7498-4: 1989 的系统管理，描述对象及管理服务。通过管理服务在本地或远程控制对象。管理服务分为三组：

上下关系管理（context management）：

上下关系管理为建立和释放管理连接提供服务。

组态管理（configuration management）：

组态管理提供的服务用于标识站的通信部件，装载和读出通信关系表（CRL），存取 1/2 层的变量、计数器和参数。

故障管理（fault management）：

故障管理为识别和排除错误提供服务。

3.3 PROFIBUS 通信模型（FMS）

按通信的观点，一个应用过程包括所有程序、资源以及与通信层无关的任务。它包括操作系统、实时应用过程、应用程序和通信驱动程序（ALI，应用层接口）。

PROFIBUS 通信模型通过通信关系将分散的应用过程组合到一个总过程中（见图 2.）。应用过程可以分布在几个不同的设备上。在一个设备中可以存在一个或多个应用过程。应用过程与执行过程时所必须的过程对象（变量、程序等）一起工作。通过适用的各种属性、规则和操作对一个过程对象进行描述。PROFIBUS 通信模型支持应用过程的面向对象的操作。

用以下图例加以说明：

```
O-------------- ：有通信端点的通信关系
-------------> ：报文（PDU）                    # n      ：通信引用
+               ：应用对象                      —        ：通信对象
+/-             ：映象到通信对象的应用对象
= = = = = = =
=           =   ：设备                          ★★★★★★★
= = = = = = =                                   ★       ★   ：应用过程
: : : : : : :                                   ★★★★★★★
:           :
:           :   ：虚拟现场总线设备（VFD）
: : : : : : :
```

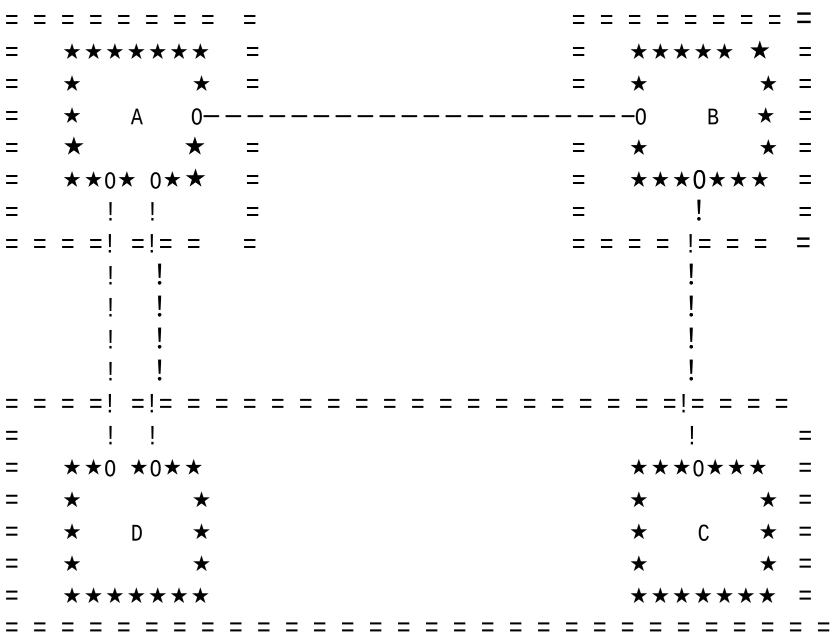


图 2．由子过程 A 至 D 组成的应用过程

3.3.1 应用过程与通信间的关系

应用过程必须利用通信端点访问通信（见 ISO 7498）。一个或多个通信端点被固定并唯一地分配给一个应用过程。应用过程借助于通信引用对这些端点寻址。通信引用是设备专用的，不由通信本身定义。在两个应用过程之间可能存在一个或多个通信关系，如图 3．所示，每一个通信关系拥有唯一的通信端点。

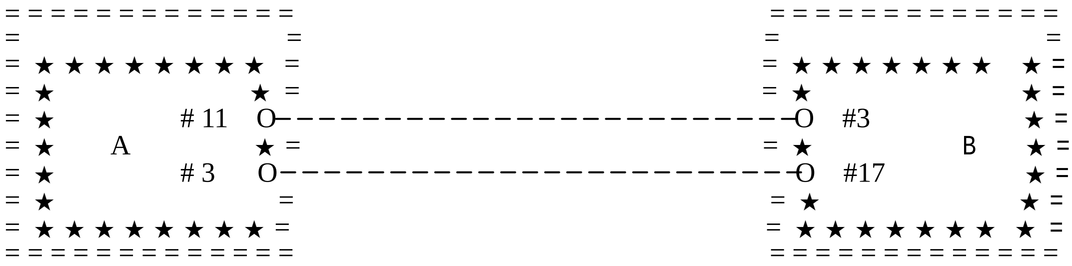


图 3．对应用过程分配通信引用

为了允许一个应用过程与另一设备的应用过程通信，必须使用通信对象。某些通信对象（虚拟对象）表示实际存在的过程对象，使应用过程对通信对象可见和可存取。利用所提供的 FMS 服务能够存取通信对象。

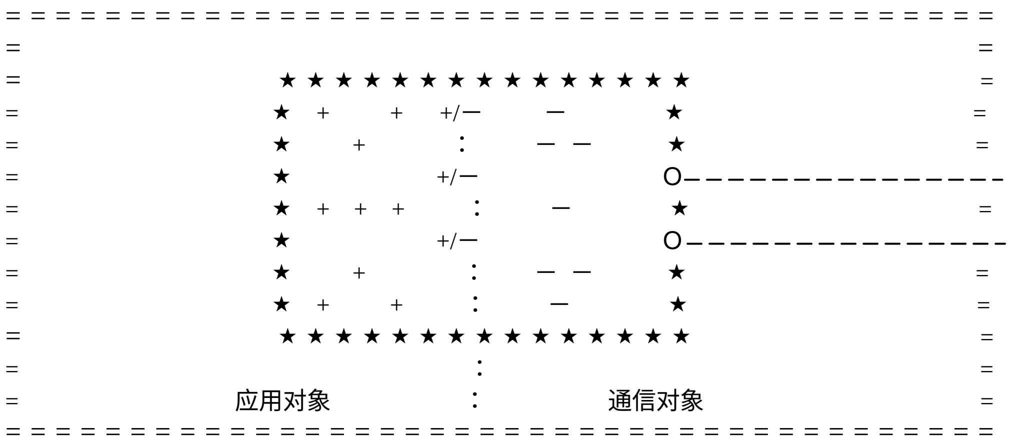


图 4．具有通信对象的应用过程

3.3.1.1　现场总线报文规范（FMS）服务

FMS 服务允许应用过程使用远程应用过程的服务器功能。通过报文（协议数据单元，Protocol Data Units ，PDUS）将 FMS 服务发送到通信伙伴。每个 FMS 服务是通信模型的要素，在使用服务时通信模型决定操作顺序和规则。在通信模型中，FMS 服务对通信对象起作用。某些服务只使用某些通信对象或只对某些通信对象起作用。

有些服务在执行之后由远程应用过程明确地确认，这种服务称之为需确认的服务；而对另一些服务，通信伙伴的应用过程不对服务的执行予以确认，这种服务称之为无需确认的服务。服务所需要的参数由应用过程提供。通过列表方式定义所需的参数，即在形式上和逻辑上对其进行描述。在表的第一列指出参数名，此后的列是特定的服务原语。由在参数行和服务原语列的登入项给定参数对服务原语的关系。

下列各种关系是可能的：

— 参数是强制性的。

— 参数是用户可选择的，可能被使用或者被忽略。

— 参数可能是从一组参数中选出。

— 一个参数的存在依赖于另一参数。

参数可以是结构式的，结构式参数的子参数名用 2 个字符定义。在此表示法中不明确地包含这样的信息，即所用的参数是否是可选的以及在具体情况下一组参数中的哪一个参数被选中。但是，在具体的接口上应传送这样的信息。

服务请求和服务指示的参数称之为变元。服务应答（Res 和 Con），在成功的情况下包含参数**结果**（＋）或在出错的情况下包含参数**结果**（－）。通常，对这些参数进行进一步的构造。

表 1．服务原语的参数

参数名	.req	.ind	.res	.con
变元（Argument）	M	M =		
请求参数 1（Request Parameter 1）	M	M =		
参数 A（Parameter_A）	S	S =		
参数 B（Parameter_B）	S	S =		
请求参数 2（Request Parameter2）	U	U =		
结果（+）（Result（+））			S	S =
应答参数 1（Acknowledge Paramenter1）			M	M =
应答参数 2（Acknowledge Paramenter2）			C	C =
结果（-）（Result（-））			S	S =
出错（Error）			M	M =

.req　　：请求服务原语

.ind　　：指示服务原语

.res　　：响应服务原语

.con　　：确认服务原语

M　　：在原语中强制性的参数

U　　：由用户可选择的参数；可能提供，可能忽略

S　　：从 2 个或 2 个以上可能的参数集合中选择的参数

C　　：以另一参数作为条件（Conditional）的参数

在代码 M、U、S 或 C 之后的代码‘ = ’表示该参数在语义上与在表中左边紧挨它的服务原语中的参数相同。（例如，在指示服务原语列中的“ M= ”与在请求服务原语列中的“ M ”， 在语义上它们是相同的。）

在上面所举的例子中，变元作为一个请求的参数被进一步构造。它由请求参数 1 和请求参数 2 组成。请求参数 1 可以是参数_A 或参数_B。在成功情况下，应答中使用结果参数（+）。该参数可以由应答参数 1 组成或者由应答参数 1 和应答参数 2 两者组成。在失败的情况下，选择结果参数(-)，它由出错参数组成。

3.3.1.2　虚拟现场总线设备（VFD）模型

虚拟现场总线设备（VFD）模型唯一地代表实际应用过程中的部分，该部分对通信是可见的和可存取的。VFD 模型规定应用过程该部分的通信行为（见下图）。VFD 模型基于 VFD 对象。VFD 对象包括所有显式和隐式的通信对象和对它们的描述。对象描述存储在对象字典（OD）中。一个 VFD 对象确切地有一个对象字典，并确切地分配给一个应用过程。一个实际的设备可以有几个 VFD 对象，每个对象通过它的通信端点对其寻址。PROFIBUS 规范对通信对象 VFD 的对象描述作出规定。

此外，VFD 模型规定了所有 FMS 服务的执行。

```
==========================
=      ★★★★★★★★★★★★★★★★★★★      =
=      ★  +      +      ::::::::::::    ★      =
=      ★              +/-    -   O——————————————
=      ★  +      +      :    -      :    ★      =
=      ★              +/-    -   O——————————————
=      ★  +      +      ::::::::::::    ★      =
=      ★★★★★★★★★★★★★★★★★★★      =
==========================
```

图 5．对应用过程分配 VFD

3.3.2 客户机与服务器之间的关系

在通信术语中，客户机是一个涉及服务的应用过程，它使用远程应用过程的功能。服务器是一个涉及服务的应用过程，它将它的 VFD 功能提供给客户机（见图 6）。原则上，一个应用过程可以既是客户机也是服务器，根据请求提供 FMS 服务，通过报文按照给定的通信关系（PDU）向通信伙伴发送请求。

需确认服务与无需确认服务是有差别的。在需确认服务的情况下，客户机发出请求，服务器通过应答（即确认）来确认请求任务的执行。

无需确认服务由服务器启动。在需确认服务的情况下，对通信关系发出的请求有一个由应用过程指定的标识（Invoke-ID）。这种请求标识使得发出的请求与其应答具有唯一的对应关系。服务的执行过程可细分为服务原语和服务时序。

```
==============                              ==============
=     客户机     =                            =     服务器     =
=  ★★★★★★★★★  =                            =  ★★★★★★★★★  =
=  ★          ★  =                            =  ★::::::::::::    ★  =
=  ★          ★  =---------->需确认的服务---------->  =  ★:---->  :    ★  =
=  ★          ★  =                            =  ★:    !  :    ★  =
=  ★       O    ---------------- ----------------------------------O    !  :    ★  =
=  ★          ★  =<------------应答----←------------  =  ★:←---    :    ★  =
=  ★          ★  =                            =  ★::::::::::::    ★  =
=  ★★★★★★★★★  =                            =  ★★★★★★★★★  =
==============                              ==============

==============                              ==============
=     服务器     =                            =     客户机     =
=  ★★★★★★★★★  =                            =  ★★★★★★★★★  =
=  ★      :::::::::★  =                            =  ★          ★  =
=  ★      :      :★  =                            =  ★          ★  =
=  ★      :      :★  =-------->无需确认的服务---------->  =  ★-------->    ★  =
=  ★      :      :★  =                            =  ★          ★  =
=  ★      :   O------------------ ----------------------------------O          ★  =
=  ★      :::::::::★  =                            =  ★          ★  =
=  ★★★★★★★★★  =                            =  ★★★★★★★★★  =
==============                              ==============
```

图 6．客户机与服务器之间的关系

3.3.2.1 服务原语

在客户机方，发出请求（需确认的服务）用请求服务原语（.req）描述，接收服务器的应答用确认服务原语（.con）。在服务器方，接收请求用指示服务原语（.ind）描述，对请求应答的返回用响应服务原语（.res）。

客户机	服务器
Service. Request (服务．请求)	Service. Indication (服务．指示)
Service. confirm(+/-)(服务．确认(+/-))	Service. response(+/-) (服务．响应（+/-）)

图 7．需确认的服务

Service. Request （.req）：

客户机用该服务原语将服务请求递交给通信。

Service. Indication（.ind）：

根据从通信接收到的此服务原语，服务器将执行此请求服务。

Service. Response（.res）：

服务器用该服务原语向通信递交对请求的应答。

Service. Confirm（.con）：

用该服务原语，通信将服务器对请求的应答递交给客户机

无需确认的服务由服务器用请求服务原语启动，用指示服务原语通知客户机的接收。

服务器	客户机
Service. Request	Service. Indication

图 8．无需确认的服务

Service. Request （.req）：

服务器用该服务原语启动无需确认服务。

Service. indication（.ind）：

用该服务原语通知接收无需确认服务。

3.3.2.2 服务时序

通过服务时序说明客户机和服务器之间的服务原语的时序关系

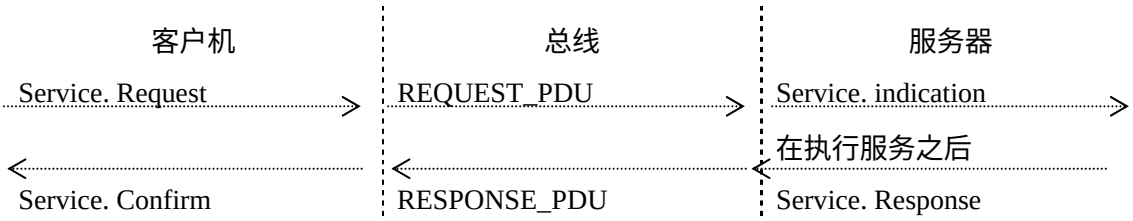


图 9．需确认服务的时序

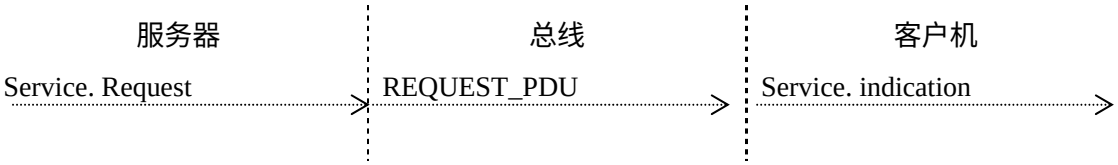


图 10．无需确认服务的时序

3.3.2.3 并行服务

在同一时间在一个连接中，如果在一个服务器中有多个来自同一个客户机的请求或者如果由一个客户机通过一个连接在同一时间对同一个服务器启动了多个请求，但还未执行，这就是说在一个通信关系中存在有并行服务。各个请求彼此之间的区别通过请求标识（Invoke-ID）来识别。

```
=================                                  =================
=     客户机      =                                  =     服务器      =
= ★★★★★★★★★★★ =                                  = ★★★★★★★★★★★ =
= ★             ★ =                                  = ★: : : : : :       ★ =
= ★             ★ =   - - ->  Service.request  - - ->  = ★- - - > :       ★ =
= ★             ★ =                                  = ★:     :       ★ =
= ★             ★ =   - - ->  Service.request  - - ->  = ★- - - > :       ★ =
= ★             ★ =                                  = ★:     :       ★ =
= ★             ★ =   - - ->  Service.request  - - ->  = ★- - - > :       ★ =
= ★            O−−−−−−−−−−−−−−−−−−−−−−−−−−−−O        ★ =
= ★             ★ =                                  = ★: : : : : :       ★ =
= ★★★★★★★★★★★ =                                  = ★★★★★★★★★★★ =
=                 =                                  =                 =
=================                                  =================
```

图 11．客户机 — 服务器/并行（需确认）服务

3.3.2.4 交互服务

如果在同一时间，对于一个通信关系而言，应用过程既是客户机又是服务器，就存在着交互（需确认的）服务。这就是说此应用过程已发出一个请求并接收到一个请求，但尚未执行。见图 12。

```
==============                                    ==============
=     客户机/服务器     =                                =     服务器/客户机
=
=  ★★★★★★★★★★  =                                =  ★★★★★★★★★★  =
=  ★  ::::::::::::  ★  =                                =  ★  ::::::::::::  ★  =
=  ★  :         :  ★  =   ----->Service. request ---->   =  ★  -------> :  ★  =
=  ★  :         :  ★  =                                =  ★  :         :  ★  =
=  ★  : <---------  ★  =   <----- Service. request <----   =  ★  :         :  ★  =
=  ★  :         O-----------------------------------------------------------O  :  ★  =
=  ★  :         :  ★  =                                =  ★  :         :  ★  =
=  ★  ::::::::::::  ★  =                                =  ★  ::::::::::::  ★  =
=  ★★★★★★★★★★  =                                =  ★★★★★★★★★★  =
=                 =                                =                 =
−−−−−−−−−−−−−−                                    −−−−−−−−−−−−−−
```

图 12．客户机 — 服务器/交互（需确认的）服务

3.3.3 应用间的通信关系

对每个通信关系进行组态，与它的使用时间无关。组态被存储在每个站的通信关系表中（CRL）。应用过程利用本地通信引用（应用专用的符号或数字）标识通信关系。通信关系分为下列三种类型：

- 一对一
- 一对多
- 一对全部

通信关系的其他属性是为站存取总线的授权。其中区分‘主-主’和‘主-从’通信关系。附加区分的特性是“面向连接”和“无连接”属性，以及“循环的”和“非循环”的连接类型。

根据所选择的通信关系，PROFIBUS 规范的应用层使用 PROFIBUS 数据链路层（FDL）的服务和优先权。FDL 为报文传输提供了两种优先权。

3.3.3.1 主站与主站以及主站与从站之间的通信关系

主站可以主动地存取总线，即独立地发送报文。从站只具有被动的总线存取能力，即仅在主站请求时可以发送报文。如果存在几个主站，必须制定授权存取总线的规则。在 PROFIBUS 系统中，通过从主站到主站传递总线存取权（即令牌）的方法达到此目的。一个主站必须在令牌持有时间期满前传出总线的存取权。

一个主站只要拥有总线存取权，它就能够轮询从站，即从站一个接着一个地接收被动总线存取（见 PROFIBUS 数据链路层协议）。PROFIBUS 规范区分‘主-主’和‘主-从’通信关系。

```
+-----------------  逻辑令牌环  -------------------+
|                                                  |
|     =========                  =========         |
|     =       =                  =       =         |
+-->=   主站   =--------------->=   主站   =---+
      =       =                  =       =
      =========                  =========
          |  |                       |  |
        |     |                   |     |
      |         |               |         |
     |   轮询    |              |   轮询    |
=====       =====            =====       =====
=    =      =   =            =   =       =   =
= 从站 =    = 从站 =         = 从站 =    = 从站 =
=    =      =   =            =   =       =   =
=====       =====            =====       =====
```

图 13．PROFIBUS 中的混合介质存取控制

3.3.3.2 通信关系的类型

在一对一的通信关系中，一个应用过程只与一个远程通信过程进行通信。在 PROFIBUS 系统中，这种通信通过面向连接的通信关系实现。

在一对多的通信关系（群播）中，一个应用过程同时与一组站的应用过程通信。确切地说，一个应用过程被一组站中的每个站寻址。

在一对全部的通信关系（广播）中，一个应用过程同时与所有站的应用过程通信。确切地说，一个应用过程被每个站寻址。

在一对多或一对全部的通信关系中，不允许对执行的请求作出应答。在 PROFIBUS 系统中，一对多或一对全部的通信通过无连接通信关系实现。

```
==========                                  ==========
= ******  =                                = ******  =
= *      O------  一对一  ----------------O      *  =
= *      *  =                             = *      *  =
= *   A  O------  一对多  --------+------O   B  *  =
= *      *  =                     |       = *      *  =
= *      O------  一对全部  --+-------O      *  =
= ******  =                     |  |    = ******  =
==========                     |  |    ==========
                               |  |
                               |  |    ==========
                               |  |    = ******  =
                               |  +----O      *  =
                               |       = *   C  *  =
                               +-------O      *  =
                               |       = ******  =
                               |       ==========
                               |
                               |       ==========
                               |       = ******  =
                               +-------O   D  *  =
                                       = ******  =
                                       ==========
```

图 14．通信关系的例子

3.3.3.2.1 面向连接的通信关系

在面向连接的通信关系中，通信双方在逻辑上建立一对一的连接。对于面向连接的通信，基本上可以划分为以下阶段：

- 连接建立阶段
- 数据传输阶段
- 连接释放阶段

在一个连接中，只有在成功地建立了连接后才能交换数据。在连接的建立阶段，向远程应用过程发出建立连接的请求（启动服务）。建立连接的请求中包含在数据传输阶段使用的服务信息。该信息包括最大的报文长度、所要求的连接类型以及其它的连接选项。

如果远程应用过程接受了建立连接的请求，它就向请求者发送对该请求的确认信息。此后，这两个应用过程都处于数据传输阶段，并可以按照约定的规则（上下关系）相互通信。一个已建立连接的有效性在 LLI 中得到控制。可以通过其连接释放即中止服务（Abort 服务）解除连接。之后，只有再次建立连接才能进行数据传输。

```
客户机                          :              服务器
-------------------------------------------------------------
<连接建立阶段>                  :
                                :
--------> 根据下列条件（上下关系）连接请求   --------------->
                                :
<-----------   根据已接受的给定条件连接请求   <---------------
<数据传输阶段>                  :
<-----------   根据约定条件的数据传输        --------------->
<连接释放阶段>                  :
------------>              连接释放              -------------->
                               或
<------------              连接释放              -------------->
                               或
<------------               出错                 -------------->
```

图 15. 面向连接的通信关系

PROFIBUS 规范定义了下列连接类型：

主-从之间的通信关系

— 非从站发起的循环数据传输的连接
— 从站发起的循环数据传输的连接
— 非从站发起的非循环数据传输的连接
— 从站发起的非循环数据传输的连接

主-主之间的通信关系

- 非循环数据传输的连接

在循环数据传输的连接中，在数据传输阶段，根据‘主站’的请求，由通信循环地执行需确认的 FMS 服务。只允许 FMS 服务的**读**和**写**操作。

此外，主站能启动带有高或低优先级的无需确认的 FMS 服务。对每一次服务请求，只能执行一次无需确认 FMS 服务；这同样也适用于循环数据传输的连接。

对从站发起的循环数据传输的连接，从站也可以启动带有高或低优先级的无需确认 FMS 服务。

如果要求在两个主站之间，建立从站发起（或非从站发起）的循环数据传输连接，考虑到第七层服务的时序，其中的一个其行为相当于一个从站。

对非循环数据传输的连接，只能执行一次需确认服务和无需确认服务。总是由主站启动服务（在‘主-主’通信关系的情况下，两者都可以）。

对从站发起的非循环数据传输的连接，从站也可以启动带有可选择优先级的无需确认 FMS 服务。

3.3.3.2.2 无连接通信关系

无连接通信是指一对多（群播）或一对全部（广播）的通信关系。对于无连接通信关系，既不建立连接，也不释放连接，它们总是处于数据传输阶段。在这种情况下，通信不可能监视连接。远程应用过程不能对请求的执行情况作出应答，即只允许无确认服务。

3.3.3.3 通信关系表（CRL）

一个 PROFIBUS 站的通信关系表（CRL）包含了该站和其他站之间所有通信关系的描述（如图 16）,该表与使用的时间无关。在组态网络时，应分别对每个 PROFIBUS 站准备 CRL 表并使用管理服务进行本地或远程装载。

对每种通信关系，其有关的通信关系类型、连接类型、上下关系、编址以及相关的 VFD 的信息都要存储在 CRL 中。

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
=                                          CRL                     =
=   ★★★★★★★★★★★      # # # # # # # # # # # #           =
=   ★                          ★      #   有关细节：      #           =
=   ★                          ★      # -  实际总线地址   #           =
=   ★          # 13        O-----→ #     -  连接类型      #>--------------------------+
=   ★                          ★      #     -  主/从          #           =          !
=   ★                          ★      #     -  上下关系      #           =          !
=   ★                          ★      #     -    VFD 指针   #           =          !
=   ★★★★★★★★★★★      # # # # # # # # # # # #           =          !
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =        !
                        I                                                               !
    应用  <---------------I----------------> 通信                                      !
                        I                                                               !
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =        !
=                                          CRL                     =          !
=   ★★★★★★★★★★★      # # # # # # # # # # # #           =          !
=   ★                          ★      #   有关细节：     #            =          !
=   ★                          ★      # -  实际总线地址   #           =          !
=   ★          # 34        O------→#     -  连接类型      #>--------------------------+
=   ★                          ★      #     -    主/从        #           =
=   ★                          ★      #     -  上下关系      #           =
=   ★                          ★      #     - VFD 的指针  #           =
=   ★★★★★★★★★★★      # # # # # # # # # # ##            =
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

图 16. 通信关系表（CRL）的例子

3.3.4 通信对象

PROFIBUS 规范为远程应用过程提供多种通信对象。对象规定了各种服务对每个通信对象所起的作用。如果某服务对一个通信对象进行操作，由规则（PROFIBUS 模型）定义通信对象的行为。

PROFIBUS 规范区分显式通信对象和隐式通信对象。隐式通信对象的对象描述由 PROFIBUS 规范定义。因此，通信既不能创建、读出和改变隐式通信对象的描述，也不能删除隐式通信对象的描述。显式通信对象用一个对象描述来指定。所有显式通信对象的对象描述均被列在站的对象字典（Object Dictionary）中。

3.3.4.1 静态通信对象及动态通信对象

显式通信对象分为两组：**静态通信对象**：简单变量，数组，记录，域和事件；

动态通信对象：程序调用，变量表。

在对象字典的静态部分登记静态通信对象。在字段区，通常静态对象很少改变，或只在某些操作模式下作一些改变。在下列的一个操作阶段中可对静态通信对象进行定义：

— 在启动或组态阶段；

— 在线；

在总线系统的规划阶段，典型的静态通信对象应在对象字典中登记。将动态通信对象登入到对象字典的动态部分。用 FMS 服务可以动态地预定义或建立和删除动态通信对象。

```
═══ ═══ ═══ ═══ ═══ ═══ ═══ ═══ ═══ ═══
=  ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★  =
=  ★                                    ★     =
=  ★  +     +                          ★  =
=  ★      +        +/-  <------  #                #  ★  =
=  ★               +/-  <------  #                #  o----------------------
=  ★  +     +       +/-  <------  #   对象字典     #  ★  =
=  ★               +/-  <------  #    (OD)        #  o----------------------
=  ★      +        +/-  <------  #                #  ★  =
=  ★  +     +                    ##################  ★  =
=  ★                                    ★     =
=  ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★  =
═══ ═══ ═══ ═══ ═══ ═══ ═══ ═══ ═══ ═══
```

图 17。.对 VFD 分配对象字典

3.3.4.2 对象编址

每个通信对象应都可能被寻址。PROFIBUS 规范定义了以下编址方法：

— 物理编址；

— 隐含编址；

— 逻辑编址或按名称编址

根据对象确定具体编址方法（Addressing scheme）。可以使用逻辑地址或名称对显式对象编址。对于显式对象，逻辑编址是强制性的，而用对象名称编址是可选的。

在逻辑编址的情况下，每个通信对象都应该使用一个可从对象字典（如图 18）中读出的逻辑地址（索引）来编址。由应用过程自身完成对过程对象的编址。由本地规则定义逻辑地址（通信）和实际地址（过程对象）之间的分配关系。若通信伙伴不能从组态中获知此地址，则可从对象字典中读到此分配关系。

在由名称编址的情况下，该名称应该是唯一的，至少在下列对象组的范围内是唯一的：

— 域

— 程序调用

— 简单变量，数组，记录

— 变量表

— 事件

名称的结构及其使用，由应用和行规文件定义。在物理编址的情况下，地址是设备专用的，因此不是由 PROFIBUS 规范定义的。可对通信对象作隐式编址，这对通信自身（VFD，事务处理对象）的管理很重要。

```
                          !索引!
                          +------+-------------------------------------+
                          ! 1  !                                     !
                          ! ·  !                                     !
                          ! ·  !                                     !
逻辑地址/                      +------+-------+--------+---------+----------+    通信对象/
 名称      17→             ! 1 7  !名称 ! 对象 ! 对象 !  内部 !→  过程对象
                          !      !     ! 代码 ! 属性 !  地址 !
                          !      !     !      !      !       !
                          +------+-------+---------+---------+----------+
                            ⋮  !                                      !
                          !  n  !                                      !
                          +------+-------------------------------------+
```

图 18. 对象字典（原理）

3.3.5 通信的对象描述模型

一个远程应用过程在对通信对象操作之前，必须知道该对象的描述。在组态期间，可以对该信息作声明。若一个应用过程在访问一个对象时不知其对象描述，应该从通信伙伴处得知此对象描述。

对象实际所在的站应对其对象描述（Source-OD）进行定义。源对象字典（Source-OD）包括一个站上所有通信对象的描述。

对每个虚拟现场设备 VFD 设计一个专门的对象字典。在本规范中所定义的结构不说明对象字典的具体实现方法，只定义了在总线上传送的对象描述的结构。此外，如果必要的对象字典信息可以构成，例如通过算法构成，则对象字典自身不必存在。因此在分布式应用中，在总线上使用不同厂家的设备，仍可提供一致的通信对象。

使用与相应通信对象相同的逻辑地址（索引）或相同的名称对一个特定的通信对象描述进行编址。对远程应用过程提供相应的 FMS 服务。使用 Get-OD 服务读对象描述。使用 Put-OD 服务，可以登记、删除或修改对象字典中的登入项。

```
客户机                          :                    服务器
<如果从组态不知道对象描述>      :
                                :
                                :
----------→获得逻辑地址 17/读名称          ---------->
            的对象描述
                                :
←---------  逻辑地址 17/名称的             <---------
            对象描述            :
                                :
                                =
                                =
                                :
----------→用逻辑地址 17/名称             ---------->
            读对象              :
                                :
←---------数据                  :         <---------
                                :
```

图 19.　读对象字典中的登入项和对象（变量对象）举例

```
客户机                          :                    服务器
<如果对象描述在通信>            :
<伙伴处产生          >          :
                                :
                                :
----------→对逻辑地址 17/名称             ---------->
            写对象描述
                                :
←---------对逻辑地址 17/名称装入的        <---------
            对象描述
                                :
```

图 20.　将登入项写入通信伙伴对象字典举例

3.3.5.1　描述

一个通信对象的对象描述包括对象代码和所分配的逻辑地址等信息，如果需要，还可包括名称。而且，在对象描述中将实际地址分配给逻辑地址或名称（如果需要）。对象描述还包括附加的对象专有属性（如图 21）。

```
+----------+-----------+----------------+----------+-----------+----------------+
! 索引     ! 对象      ! 其它的         ! 实际     ! 名称      ! 扩展           !
!          ! 代码      ! 对象           ! 对象     !           !                !
!          !           ! 属性           ! 地址     !           !                !
+----------+-----------+----------------+----------+-----------+---------------- +
                                                !
                                                !          +----------------------+
                                                !          !                      !
                                                +---------- !    通信/             !
                                                           !     过程对象         !
                                                           !                       !
                                                           !                      !
                                                           +----------------------+
```

图 21. 通信伙伴看到的一个对象登入项举例（原理）

3.3.6 保护机制

对通信端点和通信对象有保护机制。通信端点被通信保护，而通信对象被应用过程保护。

在面向连接和无连接的通信关系中，可以在通信关系表中通过组态来保护通信端点。对该通信端点的存取权限只赋予一个已定义的通信伙伴（存取控制）。

就面向连接的通信关系而言，成功地建立连接（开放通信端点）后也能够对存取进行保护。因此，多个通信伙伴能够在不同的时间内依次使用一个通信端点。对于面向连接的数据传输，几个通信伙伴不能同时使用一个通信端点。

通信对象可以在对象字典中通过组态得到保护。只有某些通信伙伴才允许存取一个通信对象。除此之外，可利用对象字典中的一个登入项去限制对一个通信对象服务的许可性（如：变量是只读的）。

其他应用过程的保护机制是设备专有的，因而它不受 PROFIBUS 规范的影响。

3.4 现场总线管理层 7（FMA7）的模型

FMA7 描述管理对象、服务和结果的模型。这些对象由 PROFIBUS 规范隐含地说明。使用对象专有的服务实现对对象的存取。应区分本地的和远程的 FMA7 服务。不能并行地执行多个 FMA7 服务。服务参数的描述类似于前面表中的说明。

FMA7 以在 PROFIBUS 数据链路层规定的 FMA1/2 功能和 LLI 与 FMS 的管理功能为基础。

3.4.1 本地管理

本地管理服务由本地提供并允许操纵本地管理对象。

管理用户通过请求服务原语发出请求。用确认服务原语接收对请求的应答。FMA7 将本地管理服务映象到 FMA1/2、LLI 和 FMS 的管理服务。

3.4.2 远程管理

远程管理服务允许操纵远程站上的管理对象。利用 FMA7_PDU，通过管理连接发送 FMA7 服务。远程管理是 LLI（类似 FMS）的用户。远程管理服务的特点在于远程用户借助于它的本地管理提供服务。远程管理是面向连接的并使用 LLI 的传输功能。

所有 PROFIBUS 站，除诊断和组态工具外，作为一个响应方只在一个管理连接中支持远程管理服务。在响应方定义管理连接的特性和编址。在多个管理连接中，组态或诊断工具还作为请求方支持管理服务。

3.4.3 缺省管理连接

通过定义缺省管理连接，为组态和诊断工具提供了对 PROFIBUS 站存取的一致性。每个 PROFIBUS 站，如果它作为响应方支持 FMA7 远程服务，它将有一个缺省管理连接，该连接在 CRL 表中登记为 CREF1。

如果要求一个站有完整的远程组态（对象字典（OD），通信关系表（CRL）和总线参数），必须在 CRL 中为组态提供两个连接：用于 CRL 组态和总线参数的管理连接和用于 OD 组态的 FMS 连接。

4 现场总线报文规范（FMS）

为了满足广泛的操作领域，PROFIBUS 通信系统为应用系统提供了多种适合于开放通信的服务。不是所有的设备和领域都要求这些服务。在应用专用的行规文件中定义了这些限制。所提供的服务遵循如下准则：

— 服务范围适应于现场总线的操作领域；
— 应合理地保证与 MMS（MAP）的兼容性；
— 应充分满足精确的时间要求；
— 用现代的面向对象的方法定义服务

4．1　服务模型

应用层的服务模型包含对对象操作的各种服务。

隐含地描述某些对象，即由 PROFIBUS 的规范定义描述。在对象字典中，对其余的对象作显式描述。使用对象专有的服务实现对对象的存取。

在对对象编址时，PROFIBUS 规范区分逻辑、物理、隐式及名称的编址。

4.1.1　服务的简短描述

Initiate（建立连接）

该项服务用于在两个通信伙伴之间建立连接。制定了关于使用通信方式的相互连接的专用协定。在使用面向连接的数据传输时，只有在成功地建立起连接后，才允许进行所有其他的通信服务。

Abort（释放连接）

通过该项服务释放两个通信伙伴之间已存在的连接。在面向连接的数据传送的情况下，只有在重新建立（Initiate）连接后，才能恢复连接。

Reject（拒绝一个不许可的服务或 PDU）

FMS 协议用 Reject 服务拒绝一个不许可的服务或接受到的一个不许可的 PDU。对于 Reject 有如下原因：

— 传输协议的冲突
— 在服务请求中出错
— 在服务响应中出错
— 服务不可执行
— 服务违背了上下关系的约定

Status（读设备/应用的状态）

该项服务用于读设备/应用的状态

UnsolicitedStatus（未经请求的状态）

该项服务用于自发地发送设备/应用的状态

Identify（读生产厂家、类型和版本的信息）

该项服务用于读供应商名称（生产厂家的名称）、类型名称（设备类型名称）及设备的版本（版本的标识）。

GetOD（读对象描述）

该项服务可从对象字典（OD）中读出单个或多个对象描述。这样，在 GetOD 服务中给出要读出的对象描述的索引或名称，或所有可以被选择的对象描述中要读出的选项。

InitiatePutOD（启动 OD 下载）

客户机使用该服务向服务器指示它要传送对象描述。

PutOD（装载对象描述）

客户机使用 PutOD 服务将对象描述传送给服务器并装入服务器的对象字典（OD）。

TerminatePutOD（终止下载 OD）

客户机使用该项服务告诉服务器已完成对象描述的传递。

InitiateDownloadSequence（启动下载序列）

利用该项服务，声明从客户机将数据块（参数，程序代码…）传送到服务器的域中。在执行下载前，PROFIBUS 规范要求在对象字典中定义服务器的域。域的特有属性也存储于 OD 中。InitiateDownloadSequence 服务不设置该属性。

DownloadSegment（传送下载的数据块）

服务器利用该项服务从客户机获取数据块。客户机传送数据和一个附加的通知，此通知指出是否有更多的数据准备发送。数据块的最大长度由 PROFIBUS 帧中的用户数据长度决定。

TerminateDownloadSequence（终止下载序列）

服务器通过该项服务通知在它的域中的数据块传送已经完成。另外，服务器还指示传输是否已成功地完成。

RequestDomainDownload（请求下载）

服务器通过该项服务请求它的客户执行下载任务。服务器请求可以包含可选的信息，该信息指出从客户的哪个文件中读出这些数据。只有在完成下载后，才传送对该请求的回答（应答）。

InitiateUploadSequence（启动上载序列）

客户机用该项服务启动服务器域中的数据块（参数，程序代码）的传送。

UploadSegment（发送一个上载的数据块）

客户机利用该项服务获取来自服务器域中的数据块。服务器传送数据和一个附加通知，此通知指出是否有更多的数据准备发送。数据块的最大长度由 PROFIBUS 帧中的用户数据长度决定。

TerminateUploadSequence（终止上载序列）

客户机利用该项服务通知已完成从服务器到客户机的数据块传送。

RequestDomainUpload （请求上载）

服务器利用该项服务请求它的客户机执行上载任务。服务器的请求可以包括可选的信息，该信息指出将上载数据写入客户机的哪个文件。在下载完成前，不发送对请求的回答（应答）。

CreateProgramInvocation（将域并入一个程序）

利用该项服务将在对象字典中描述的域（例如：代码，数据）在线地并入一个程序调用对象。使用逻辑地址对该对象寻址。

DeleteProgramInvocation（删除一个程序）

用该项服务删除一个程序调用的对象。

Start（在复位后起动一个程序）

从起始点开始执行一个程序。

Stop（停止运行一个程序）

停止一个正在运行的程序，但不设定它回到起始点。

Resume（在停止后重新开始运行一个程序）

将停止的程序设置成运行（**RUNNING**）状态，但不复位。

Kill（停止一个程序的运行）

设置一个程序调用为不可运行（**UNRUNNABLE**）状态，此状态与它的当前状态无关。

Read（读变量）

读一个在对象字典中定义的变量对象的值。

Write（写变量）

利用该服务对对象字典中描述的一个变量对象赋值。

ReadWithType（读变量，带类型）

读一个在对象字典中描述的变量对象的值及其数据类型描述。

WriteWithType（写变量，带类型）

赋值给在对象字典中描述的变量对象，此值附加了数据类型描述。

PhyRead（读物理存取对象）

用该项服务读一个物理存取对象的值。

PhyWrite（写物理存取对象）

用该项服务对一个物理存取对象赋值。

InformationReport（发送数据 — 无需应答）

为了同步的目的，将对象字典中描述的一个变量对象的值发送给所有其他站。

InformationReportWithType（发送数据 — 无需应答）

为了同步的目的，将对象字典中描述的一个变量对象的值及其数据类型描述发送给所有其他站。

DefineVariableList（将变量合并成一个表）

利用该项服务，将静态对象字典中描述的变量对象在线地合并成一个新的变量对象表。合并后，用逻辑地址对该对象寻址。

DeleteVariableList（删除变量表）

该项服务删除一个变量表。

EventNotification（通知事件）

利用该项服务，向另一通信伙伴发送一个预定义的事件消息。此外，还发送可选事件数据。

EventNotificatonWithType（通知事件，带类型）

利用该项服务，向另一通信伙伴发送一个预定义的事件消息。此外还发送可选的事件数据；在这种情况下，事件数据应附加数据类型描述。

AcknowledgeEventNotification（应答事件）

用该项服务确认接收到一个事件。

AlterEventConditionMonitoring（允许/禁止事件）

用该项服务允许或禁止一个事件的发送。

4.1.1.1 服务的边界条件

对所有作为服务器的 PROFIBUS 设备都提供以下服务：

— Initiate（启动）
— Abort（中止）
— Reject（拒绝）
— Status（状态）
— Identify（标识）
— GetOD（获得对象字典）（只用短形式，见上节 OD 服务）

通信关系中的一系列服务有如下限制：（见 LLI，通信关系章节）

— 主/主或主/从
— 面向连接（非循环，循环，有或没有从站启动的）
— 无连接的（广播，群播）

通过应用专用的定义（行规，Profile），可以进一步对各种服务加以限制。一个对象只能用一个服务寻址，该对象可以由几个对象依次组成。通过一个服务，最多可以发送大约 220 个字节的用户数据。除了对每个服务定义的专用服务参数以外，还应在‘ FMS-用户 ’接口发送通信引用，FMS 将通信引用传递给 LLI。

在需确认的服务中，还应该发送请求的标识（Invoke-ID）。例外的情况是 Initiate-Service，它不需要 Invoke-ID。

4.1.1.2 客户机和服务器

在通信术语中客户机是一个应用过程，相对一个服务来讲，它使用一个远程应用过程的 VFD 功能。服务器也是一个应用过程，相对一个服务来讲，它向客户机提供其 VFD 功能。一个应用过程可以是一个服务器，同时也可以是一个客户机。

针对请求提供各种服务。通过已定义的通信关系使用报文（PDU）将请求发送给通信伙伴。

4.1.1.2.1 需确认服务和无需确认服务

服务有“需确认”和“无需确认”之分。在需确认服务中，客户机发出请求，服务器用应答确认此请求的执行（确认）。

无需确认服务由服务器启动。对于需确认服务，按通信关系发出的请求应该从应用过程中接收一个标识（Invoke-TD）。此请求标识使发出的请求和相应的应答之间具有唯一的赋值（assignment）。通过服务原语和服务时序详细地描述服务的执行。

4.1.1.2.2 服务原语和服务序列

需确认服务（Confirmed Services）

图 22 说明了在需确认服务中客户机与服务器之间的基本时序：

```
客户机                    总线                    服务器
Service.req     !     REQUEST_PDU    !     Service.ind
---------------→! -------------------------→!-------------------------→
                !                            !
                !                            !     服务执行后
←---------------!←------------------------- !←------------------------
Service.con     !     RESPONSE_PDU   !     Service.res
```

图 22. 需确认服务的时序

在客户机方，用请求服务原语（.req）描述请求（需确认服务），用确认服务原语（.con）描述接收服务器对请求的应答

在服务器方，用指示服务原语（.ind）描述接收请求，用响应服务原语（.res）描述发送对请求的应答。

无需确认服务（Unconfirmed Services）

图 23 说明了在无需确认服务中客户机与服务器之间的基本时序：

```
服务器                    总线                    客户机
Service.req     !     REQUEST_PDU    !     Service.ind
---------------→! -------------------------→!-------------------------→
```

图 23. 无需确认服务的时序

一个无需确认服务由服务器用请求服务原语（.req）启动，用指示服务原语（.ind ）通知客户机方接收。

4.1.2 对象的简短描述

显式对象（Explicit Objects）

PROFIBUS 规范识别由对象字典显式描述的下列对象：

— 对象字典
— 域
— 程序调用
— 简单变量
— 数组
— 记录
— 变量表
— 事件

隐式对象（Implicit Objects）

PROFIBUS 规范识别隐式描述的下列对象：

— 事务处理对象
— VFD
— 描述对象字典的对象
— 数据类型
— 数据类型结构说明
— 描述域的对象
— 描述程序调用的对象
— 描述简单变量的对象
— 描述数组的对象
— 描述记录的对象
— 描述变量表的对象
— 描述事件的对象
— 物理存取对象

用下列形式描述每个子条目中的对象：

Object：对象的命名

Key Attribute：该对象的关键属性

Attribute：对象的属性

Attribute：对象的属性

属性（Attribute）

属性表示对象的某个确定的特性。

关键属性（Key Attribute）

对象的关键属性是指可以使对象被唯一寻址的特征。如果用两个关键属性的组合对一个对象寻址，那么这两个属性之间用字符“ & ”隔开。

如果用几个可交替的关键属性对一个对象寻址，那么这些属性在后续几行中作声明。第一个关键属性总是用于对象寻址，用其他关键属性的寻址只是选项。

4.1.2.1 存取权

应当防止无授权地存取对象。用于存取对象的服务只限于经授权的通信伙伴使用。

有了存取此对象的授权才能存取此对象，与存取控制相关的信息存放在 OD 的对象描述中和特

定对象的对象描述中。当建立一个连接时，用 Initiate 服务部分地发送此信息。作为对象描述自身不提供任何存取保护。

OD 对象描述：支持存取保护的属性

对象描述：	口令属性
	存取组属性
	存取权属性
Initiate 服务：	支持存取保护的参数（调用）
	口令参数（调用）
	存取组参数（调用）
	支持存取保护的参数（被调用）
	口令参数（被调用）
	存取组参数（被调用）

支持存取保护（Access Protection Supported）

在存取一个对象时，应指示是否检查存取权。

支持存取保护：假（false）

每个通信伙伴被授权可存取每一个对象。在对象描述中不存在口令、存取组和存取权的属性。

支持存取保护：真（true）

根据 Initiate 服务所发送的参数（口令，存取组）以及对象自身的属性（口令，存取组和存取权）进行授权。对象属性存取权区分如读、写、和执行的权限。

每次存取都要检查授权情况：

— 如果对象的存取权属性允许所有通信伙伴的存取，则授权存在。
— 如果对象的存取权属性允许存取组的存取，而且如果客户机至少是一个授权存取组中的一个成员，则授权存在。
— 如果对象的存取权属性允许使用口令存取，而且如果对象描述的口令与用 Initiate 服务发送的口令相同，则授权存在。

口令（Password）

PROFIBUS 规范区分在对象描述中的属性口令和 Initiate 服务中的参数口令。

— 属性口令（对象描述）
  该属性包含一个由用户定义的标识，它根据存取权属性授权存取此对象。
— 参数口令（Initiate 服务）
  该参数表明此口令对一个特定的通信关系有效。如果在一个通信关系中不提供带口令的存取，则口令参数的值应为 0。在支持带口令存取的站上，用 Initiate 服务接收的口令参数的值对所有的通信关系都应该是唯一的（口令为 0 除外）。

存取组（Access Group）

PROFIBUS 规范区分对象描述的存取组属性和 Initiate 服务的存取组参数。可能有 8 种不同的存取组。

— 存取组属性（对象描述）
  该属性包含了一个由用户定义的组，它根据存取权属性授权存取此对象。
— 存取组参数（Initiate 服务）
  该参数定义了属于一个或多个组的成员关系，这些组对一个通信关系是有效的。如果一个通信关系不处理带有存取组的存取，则存取组参数的二进制值应为 00000000。

存取权（Access Rights）

PROFIBUS 规范按照用口令存取、对存取组的存取和对所有通信伙伴的存取区分不同的权限，例如读、写和执行。这样才有可能给出不同的存取权限，如读所有通信伙伴的权限、写存取组的权限、删除单个通信伙伴的权限（口令）等。

对下列通信对象可以分配存取权限：

表 2. 对象的存取权限

对象	权限
域	上载，下载，建立 PI
程序调用	启动，停止，删除
简单变量	读，写
数组	读，写
记录	读，写
变量表	读，写，删除
事件	改变，确认

存取权限举例

OD 对象描述 通信伙伴A
支持存取保护 =true
对象字典 通信伙伴 A
index 114 objectcode =var password =0 access groups =00 hex access rights =ra index 115 objectcode =var password =134 access groups =40 hex access rights =r,rg,wg index 116 objectcode =array password =177 access groups =07 hex access rights =w,rg

OD 对象描述 通信伙伴 B
支持存取保护 =false
对象字典 通信伙伴 B
index 78 objectcode = record index 83 objectcode = var index 84 objectcode = var

注释：

var　　简单变量
r　　用口令读的权限
rg　　读存取组的权限
ra　　读所有通信伙伴的权限
w　　用口令写的权限
wg　　写存取组的权限

Initiate　服务，通信伙伴 A 作为客户机

------->	Initiate.req	支持存取保护（调用）	= true
		口令（调用）	= 0
		存取组（调用）	= 00 hex
<------	Initiate.res	支持存取保护（被调用）	= false
		口令 （被调用）	= 134
		存取组（被调用）	= 05 hex

其他服务，通信伙伴 A 作为客户机

-------> Read.req　　index = 78
<------ Read.res（+）（存取所有可能的对象）

其他服务，通信伙伴 B 作为客户机

←----- Read.req　　index = 114
-----→ Read.res（+）（都可以读对象 114）
←----- Write.req　　index = 114
-----→ Write.res（-）（不允许站写对象 114）
←----- Read.req　　index = 115
-----→ Read.res（+）（口令为 134 的可以读）
←----- Write.req　　index = 115
-----→ Write.res（-）（站 B 不属于存取组 40 hex）
←----- Read.req　　index = 116
-----→ Read.res（+）（存取组 04 hex 和 01 hex 可以读）
←----- Write.req　　index = 116
-----→ Write.res（-）（口令不正确，并且存取组没有被授权写）

符号解释：

-------> 从 A 站到 B 站的信息
<------ 从 B 站到 A 站的信息

4.1.2.2 对象的域限制

一个通信对象的有效域是该通信对象可在其中唯一编址的域。

4.1.2.3 对象的创建和删除

所有对象都在组态期间创建。此外，利用有关服务可以动态地建立和删除对象变量表及程序调用。

4.1.3 对象编址

每个对象都可以用事先定义了的寻址方案编址。以下的编址方案是允许的：

— 逻辑编址（索引）

— 物理编址
— 隐含编址
— 名称编址

4.1.3.1 逻辑编址

用逻辑地址（索引）实现对下列对象的编址：
— 描述对象字典的对象
— 描述域的对象
— 描述程序调用的对象
— 描述简单变量的对象
— 描述数组的对象
— 描述记录的对象
— 描述变量表的对象
— 描述事件的对象
— 域
— 程序调用
— 数据类型
— 数据类型结构描述
— 简单变量
— 数组
— 记录
— 变量表
— 事件

逻辑地址用 16 位（bit）无符号数的索引来表示。索引和对象之间的关系储存在对象字典的对象描述中。

4.1.3.2 物理编址

只能使用物理编址方法对物理存取对象编址。物理编址由一个 32 个位(bit)的地址来完成，由编址系统定义该地址的含义。

4.1.3.3 隐式编址

对象 VFD（虚拟现场设备）用通信引用（CREF）编址。通过 VFD 实现对对象字典对象的编址。专用于连接的事务处理对象的编址，是通过 invoke-ID 和需确认服务的通信引用来实现的。

4.1.3.4 名称编址

用名称编址是可选项。除用逻辑编址外，以下对象也可以用名称编址：
— 描述域的对象
— 描述程序调用的对象
— 描述简单变量的对象
— 描述数组的对象
— 描述记录的对象
— 描述变量表的对象
— 描述事件的对象
— 域
— 程序调用
— 简单变量

— 数组
— 记录
— 变量表
— 事件

名称用一个可视字符串表示。该可视字符串的长度是固定的，在 OD 对象描述中由名称长度属性对其定义。名称长度属性值最大为 32。

如果没有定义对象名称，则根据已定义的长度，将名称属性设定为仅由 SPACE 字符（空格）组成的可视字符串。

如果名称在对象字典中是允许的，且如果在相关通信关系的 FMS CRL 登入项中的支持 FMS 特征字段指示支持“ addressing by name（用名称编址）”的选项，此时才可能使用名称编址。

名称的结构和用途由应用和行规定义。

只有 SPACE 字符（空格）组成的名称不能被 FMS 服务用来编址对象。

至少在下列的对象组中名称应该是唯一的：

— 域
— 程序调用
— 简单变量，数组，记录
— 变量表
— 事件

用名称编址要求在服务中给出的参数“ name ”与在 OD 中相关对象组内的对象名称相匹配。

如果在 OD 对象描述中指出的长度是相同的而且每个字符都相同，那么名称是相匹配的。

4.1.4 为主站/从站和对象指定的服务

表 3. 列出的内容，概括地反映了被动站（从站）应提供的各种服务、主动站（主站）应提供的各种服务以及各种可选的服务。除此以外，还分别列出了服务器和客户机的功能。该表还区分需确认服务和无需确认服务。表 4. 列出了特定服务/对象的对应关系。

表 3. 对主站和从站分配的服务

服务	Slave		Master		cfd/
	Cli	Serv	Cli	Serv	unc
Initiate	-	M	O	M	cfd
Abort	M	M	M	M	unc
Reject	M	M	M	M	unc
Status	-	M	O	M	cfd
UnsolicitedStatus	O	O	O	O	unc
Identify	-	M	O	M	cfd
GetOD（short form）	-	M	O	M	cfd
GetOD（long form）	-	O	O	O	cfd
InitiatePutOD	-	O	O	O	cfd
PutOD	-	O	O	O	cfd
TerminatePutOD	-	O	O	O	cfd
InitiateDownloadSequence	-	-	O	O	cfd
DownloadSegment	-	-	O	O	cfd
TerminateDownloadSequence	-	-	O	O	cfd
RequestdomainDownload	-	-	O	O	cfd
InitiateUploadSequence	-	O	O	O	cfd
UploadSegment	-	O	O	O	cfd
TerminateUploadSequence	-	O	O	O	cfd
RequestdomainUpload	-	-	O	O	cfd
CreatProgramInvocation	-	O	O	O	cfd
DeleteProgramInvocaton	-	O	O	O	cfd
Start	-	O	O	O	cfd
Stop	-	O	O	O	cfd
Resume	-	O	O	O	cfd
Reset	-	O	O	O	cfd
Kill	-	O	O	O	cfd
Read	-	O	O	O	cfd
Write	-	O	O	O	cfd
ReadWithType	-	O	O	O	cfd
WriteWithType	-	O	O	O	cfd
PhysRead	-	O	O	O	cfd
PhysWrite	-	O	O	O	cfd
InformationReport	O	O	O	O	unc
InformationReportWithType	O	O	O	O	unc
DefineVariableList	-	O	O	O	cfd
DeleteVariableList	-	O	O	O	cfd
EventNotification	O	O	O	O	unc
EventNotificationWithType	O	O	O	O	unc
AcknowledgeEventNotification	-	O	O	O	cfd
AlterEventConditonMonitoring	-	O	O	O	cfd

所用的缩略语：

Cli	客户机 （Client）	cfd	需确认服务（confirmed service）
Serv	服务器（Server）	unc	无需确认服务（unconfirmed service）
M	强制性的（Mandatory）	O	可选的（Optional）
-	不允许的服务		

表 4. 服务/对象的对应关系

服务	对象										
Initiate											
Abort			T								
Reject											
Status	F		T								
UnsolicitedStatus	F										
Identify	F		T								
GetOD	O	H	T						Dt	Ds	
InitiatePutOD	O		T								
PutOD	O	H	T	D	P		V	VL	Dt	Ds	E
TerminatePutOD	O		T								
InitiateDownloadSequence	O	H	T	D							
DownloadSegment	O	H	T	D							
TerminateDownloadSequence	O	H	T	D							
RequestdomainDownload	O	H	T	D							
InitiateUploadSequence	O	H	T	D							
UploadSegment	O	H	T	D							
TerminateUploadSequence	O	H	T	D							
RequestdomainUpload	O	H	T	D							
CreateProgramInvocation	O	H	T	D	P						
DeleteProgramInvocation	O	H	T	D	P						
Start			T		P						
Stop			T		P						
Resume			T		P						
Reset			T		P						
Kill			T		P						
Read			T				V	VL			
Write			T				V	VL			
ReadWithType			T				V	VL			
WriteWithType			T				V	VL			
PhysRead			T			Y					
PhysWrite			T			Y					
InformationReport							V	VL			
InformationReportWithType							V	VL			
DefineVariableList	O	H	T					VL			
DeleteVariableList	O	H	T					VL			
EventNotification											E
EventNotificationWithType											E
AcknowledgeEventNotification			T								E
AlterEventConditionMonitoring	O		T								E

缩略语说明：

F	VFD	Y	物理存取对象
O	对象字典（OD）	V	简单变量、数组、记录
H	OD 对象描述	VL	变量表
T	事务处理对象	Dt	数据类型
D	域	Ds	数据类型结构描述
P	程序调用	E	事件

4.1.5 数据类型

在 PROFIBUS 规范中，以下数据类型确定为通信对象的标准化数据类型：

— 布尔数（boolean）
— 整数（integer）
— 无符号数（unsigned）
— 可视字符串（visible string）
— 八位位组串（octet string）
— 位串（bit string）
— 日期（date）
— 日时（time of day）
— 时差（time difference）

此外，PROFIBUS 允许对数据类型进行组态。

物理上，数据类型由一个或多个八位位组（字节）构成。一个八位位组由８个二进制位组成，其编号从１到８。第一位是 LSB（最低有效位）。对不同数据类型的编码方式见第６部分中的编码。

４.２ VFD 支持

4.2.1 模型描述

虚拟现场总线设备（VFD）是通信伙伴看到的用于描述自动化系统数据和行为的抽象模型。

VFD 模型的基础是 VFD 对象。VFD 对象包括所有可以由通信用户使用的对象和对象描述，用户通过服务使用 VFD。

对象描述存放在对象字典中（OD），每一个 VFD 精确地对应一个对象字典（OD）。

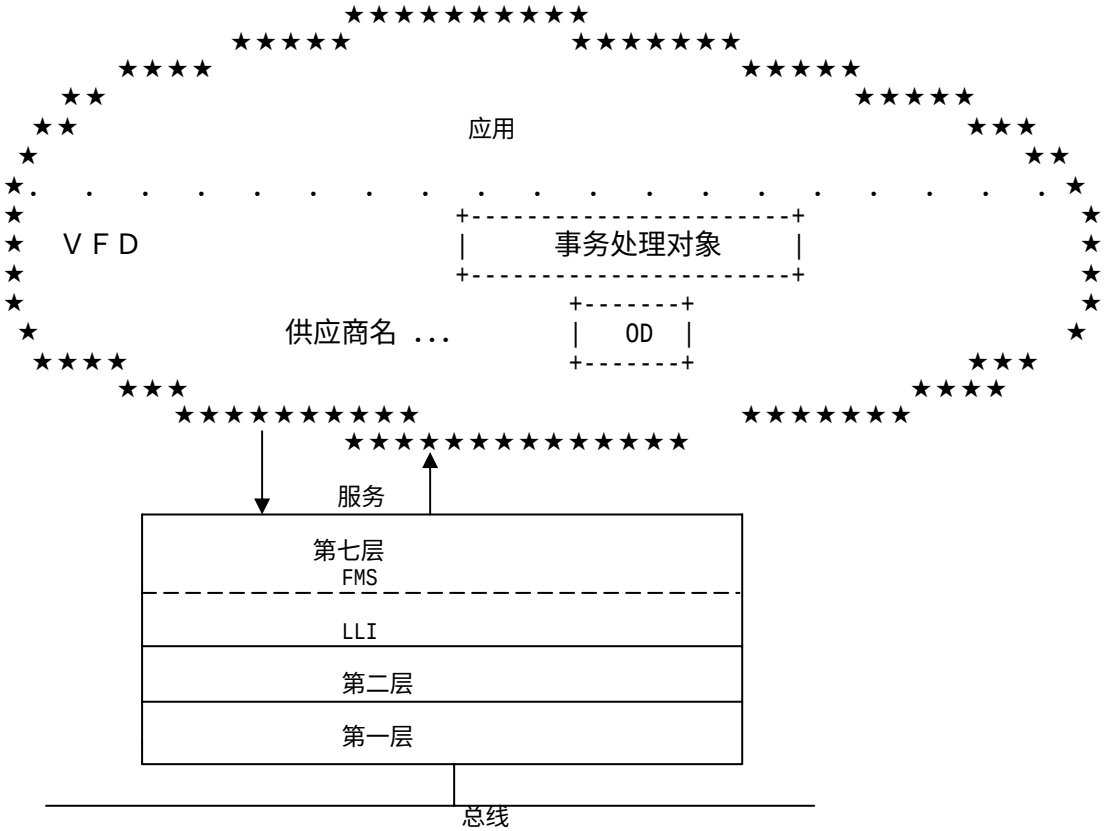


图 24. VFD 模型

第 7 层的服务不对实现定义具体的接口，而是以抽象的形式描述所提供的功能。

应用不是本规范的主题，只是指出应用如何使用抽象描述的服务。

由通信关系表（CRL）隐含地定义 VFD 对象的编址方法。原则上，每个设备只能使用一个 VFD 对象。对一个设备扩展到几个 VFD 对象也是可能的。

```
  通信关系表（FMS CRL）                          VFD 1
+----------------------------+          +-------------------------------------------------------+
!          X ---------------------+-------->!   +----------------------------+                  !
!----------------------------!    !        !   !       事务处理          !                  !
!          X ---------------------+        !   !         对象             !                  !
!----------------------------!    !        !   +----------------------------+                  !
!          X ---------------------+        !                                                     !
!----------------------------!    !        !   +---------------------------------+              !
!          X ---------------------+        !   !    OD  对象描述              !              !
!----------------------------+    !        !   +---------------------------------+              !
                                  !        !       !                                             !
                                  !        !       V                                             !
                                  !        !   +----------------------+     +----------------+   !
                                  !        !   !    OD        X-------->!    对象      !   !
                                  !        !   !                    !     +----------------+   !
                                  !        !   !                    !     +----------------+   !
                                  !        !   !                X-------->!    对象      !   !
                                  !        !   +----------------------+     +----------------+   !
                                  !        +-------------------------------------------------------+
                                  !                                VFD 2
                                  !        +-------------------------------------------------------+
                                  !-------->!                                                     !
                                           +-------------------------------------------------------+
```

图 25. 一个自动化系统（VFD）的抽象模型

定义如下支持 VFD 的服务：

```
- Status     （状态）                          \                  +----------------------------+
- UnsolicitedStatus（未经请求的状态）          >--------------->  !          VFD           !
-Identify（标识）                              /                  +----------------------------+
```

图 26. 支持 VFD 的服务

使用 Status/UnsolicitedStatus 服务，应用可以得到设备的特定状态信息。使用 Identify 服务，可以为通信关系识别合适的 VFD。

4.2.2 VFD 对象

隐含地定义 VFD 对象。不能远程读出 VFD 的对象描述。用下列属性标识 VFD 对象：

— 供应商名（VendorName）

— 型号名（ModelName）

— 版本号（Revision）

— 行规文件号（ProfileNumber）

若有多个 VFD，在 FMS CRL 的登入项中应标明哪些 VFD 将被寻址。

4.2.2.1 属性

对象：VFD

关键属性：通过通信关系隐含

属性：供应商名（VendorName）

属性：型号名（ModelName）

属性：版本号（Revision）

属性：行规文件号（ProfileNumber）
属性：逻辑状态（LogicalStatus）
属性：物理状态（PhysicalStatus）
属性：VFD 特定对象的表（List of VFD Specific Object）

通过通信关系隐含：

由 FMS 登入在通信关系表中（见 3 .4.2 节）的通信关系指向选定的 VFD 对象。

供应商名称：

该属性是可视字符串，用以识别设备的供应商。

型号名：

该可视字符串是设备的型号名，由供应商提供。

版本号：

此属性是一可视字符串，它说明此设备的版本号。该可视字符串的值由供应商提供。

行规文件号：

该行规文件标识的长度固定为 2 个八位位组，由供应商定义该属性。如果设备没有相应的行规文件，则这两个字节都被置成“ 0 ”。

逻辑状态：

该属性包括设备通信功能的状态信息。

0 <=> 通信准备就绪

2 <=> 限定服务数目

4 <=> OD-LOADING-NON-INTERACTING

5 <=> OD-LOADING -INTERACTING

通信准备就绪

- 通常所有的服务都可使用。

限定服务数目

服务器至少支持以下服务：

- Initiate（启动）
- Abort（中止）
- Reject（拒绝）
- Identify（标识）
- Status（状态）
- GetOD（获得对象字典）

OD-LOADING-NON-INTERACTING

如果对象字典处在 OD-LOADING-NON-INTERACTING 状态，则不允许执行 InitiatePutOD 服务。

OD-LOADING-INTERACTING

如果对象字典处在 OD-LOADING-INTERACTING 状态，则除了接收服务 InitiatePutOD 的连接外，其他所有连接被锁定，并且拒绝建立其他的连接。在该连接中，只允许以下服务：

— Initiate（启动）
— Abort（中止）
— Reject（拒绝）
— Status（状态）
— Identify（标识）
— PhysRead（物理读）
— PhysWrite （物理写）
— GetOD （获得对象字典）
— InitiatePutOD （启动放置对象字典）
— PutOD （放置对象字典）
— TerminatePutO（终止放置对象字典）

物理状态：

该属性给出了实际设备状态的概要。

0 <=> 可操作的

1 <=> 部分可操作的

2 <=>不可操作的

3 <=>需要维修的

VFD 的特定对象表包括所有 VFD 的 特定对象。

4.2.3 VFD 支持服务

4.2.3.1 状态（Status）

使用 Status 服务读设备/用户的状态。

表 5. 状态（Status）

参数名	.req .ind	.res .con
变元（Argument）	M	
结果（+）（Result（+））		S
逻辑状态（LogicalStatus）		M
物理状态（PhysicalStatus）		M
本地细节（LocalDetail）		U
结果（-（）Result（-））		S
出错类型（ErrorType）		M

变元（Argument）：

变元不包括服务专用的参数。

结果（+）（Result（+））：

该参数指出服务被成功地执行。

逻辑状态（LogicalStatus）：

该参数包含设备通信功能的状态信息。

物理状态（PhysicalStatus）：

该参数给出了实际设备状态的概要。

本地细节（LocalDetail）：

该参数给出设备和应用的本地状态。由行规文件定义各个位的含义。

结果（-）（Result（-））：

参数 Result（-）指出服务没有被成功地执行。

出错类型（ErrorType）：

包含服务执行失败的原因。

4.2.3.2 未经请求的状态（UnsolicitedStatus）

用户使用 UnsolicitedStatus 服务的目的是希望自发地发送设备/用户状态。

表 6．UnsolicitedStatus

参数	.req .ind
变元	M
优先权	M
逻辑状态	M
物理状态	M
本地细节	U

变元：

变元包含服务调用所专用的服务参数。

优先权：

该参数给出服务的优先权，它的值应是下列 2 者之一：
true：高优先权
false：低优先权

逻辑状态：

该参数包含设备通信功能的状态信息。

物理状态：

该参数给出了实际设备状态的概要

本地细节：

该参数给出设备和应用的本地状态。由行规文件定义各个位的含义。

4.2.3.3 标识（Identify）

使用标识服务读出 VFD 的标识信息。通过 Initiate 服务 发送属性 VFD 行规文件号。

表 7. 标识（Identify）

参数名	.req .ind	.res .con
变元	M	
结果（+）		S
供应商名		M
型号名		M
版本名		M
结果（-）		S
出错类型		M

变元:

变元不包括服务专用的参数。

结果（+）：

该参数指出参数可读。

供应商名：

该参数包含 VFD 属性中的供应商名。

型号名：

该参数给出设备的型号名。

版本号：

该参数给出 VFD 的版本属性。

结果（-）：

参数 Result（-）指出服务不能执行。

出错类型:

包含服务不能执行的原因信息。

4.3 对象字典（OD）管理

4.3.1 模型描述

对象字典（OD）包括了以下通信对象的对象描述：

— 数据类型（DataType）
— 数据类型结构描述（DataTypeStructureDescription）
— 域（Domain）
— 程序调用 （ProgramInvocation）
— 简单变量 （SimpleVariable）
— 数组（Array）
— 记录（Record）
— 变量表（VarialeList）
— 事件（Event）

此外，OD 可以包含 Null（空）对象。在下列情况下定义空对象：

— 在 OD 中组态一个空对象作为占位用（placeholder）。
— 如果在 OD 中存在的一个对象被 PutOD 服务删除，就创建一个空对象。
— 如果一个标准的数据类型不被支持，那么组态一个空对象以替代此数据类型。

（请阅读有关数据类型对象定义的章节）。

OD 对象描述唯一地赋予对象字典。OD 对象描述包含对象字典的结构信息。该对象描述用一个唯一的 16 位（bit）无符号数索引来标记。

一个类型为可视字符串的名称也可以分配给下列对象；

— 域（Domain）
— 程序调用（ProgramInvocation）
— 简单变量（SimpleVariable）
— 数组数组（Array）
— 记录（Record）
— 变量表（VarialeList）
— 事件（Event）

索引或名称可作为对象与对象描述的关键字。

服务按原则寻址服务器的对象，因为在服务器上存在实际的对象。

在一个远程通信伙伴可以存取一个对象之前，他应该知道该对象的描述。在远程存取对

象时，可以使用对象的逻辑地址（索引）或名称。

在系统组态期间可以定义对象描述，而且在此后的任何时候，可以在站之间传送对象描述。对象描述的定义应在对象实际存在的那个站上（源 OD）实现。不同的对象可以有不同长度的对象描述。通信伙伴持有远程对象描述（远程 OD）的全部或部分拷贝。

这样，每个站有一个描述本地存在的通信对象的源 OD，和一个或多个远程 OD。远程 OD 是由连接规定的，即对每个连接有一个远程 OD 是有效的。

OD 模型为读、写源 OD 提供了相关的服务。

如果一个源 OD 被远程的通信伙伴修改，应当通知其他所有使用该相同数据库（即源 OD）的通信伙伴。为此，用户释放与其他所有通信伙伴的连接。在修改该 OD 之后，再次建立连接。在建立连接的过程中，新的版本号（Version OD）发送给每个通信伙伴用来通知相关的源 OD 已被修改。允许在不释放连接的情况下增加对象描述。

```
                 总线站 A              !                 总线站 B
                ----------------       !                -------------------
                 远程 OD               !                   源 OD
                ……................     !               +----------------+
                . 对象描述    .        !               !          X-----------→ 对象 x
                .     的      .←-------------------------- !                !
                .    拷贝     .        !               !          X-----------→ 对象 y
                ……................     !               +----------------+
                                       !
                    源 OD              !                 远程 OD
                +----------------+     !               ……................
对象 v <-----------X          !        !               . 对象描述    .
                !             ! ------------------------ →.     的      .
对象 w <----------  X         !        !               .    拷贝     .
                +----------------+.    !               ……................
```

图 27. 源 OD/远程 OD

在下面说明的 OD 结构不描述一个 OD 的具体的实现。它只是定义了要通过总线传送的对象描述的结构。如果所需的信息能从别的方法得到（例如通过一个算法），则 OD 不必存在。

4.3.2 OD 的结构

对象字典（OD）分为几个部分：

— “ 对象字典 ” 的对象描述（OD 对象描述）
  包括 OD 的结构信息（见 OD 对象描述）。

— 静态类型字典（ST-OD）
  包括数据类型和数据类型结构的描述。

— 静态对象字典（S-OD）
  包括简单变量、数组、记录、域、和事件的对象描述。

— 变量表的动态表（DV-OD）
  包括变量表的对象描述。

— 程序调用的动态表（DP-OD）
  包括程序调用的对象描述。

索引	对象字典（OD）
0	OD　对象描述（OD_ODES）
l … i	静态类型表（ST-OD）
k … n	静态对象字典（S-OD）
p … t	变量表的动态表 （DV-OD）
v … z	程序调用的动态表 （DP-OD）

图 28. 对象字典的结构

4.3.2.1　静态类型表（ST-OD）

静态类型表（ST-OD）包括数据类型的对象描述和 S-OD 的变量存取对象的数据类型结构描述 。一个对象描述精确地分配给一个索引，该分配是静态的而且不能删除，ST-OD 可以由远程的伙伴装载（见 OD 服务）。

在对象字典中，静态类型表总是从索引 l 开始。其余的对象描述按索引的递增依次排列。在对象字典中，分配给 “ 索引<=>对象描述 ” 的数目作为 ST-OD 的长度登入。

ST-OD 分为两段。第一段包括标准 PROFIBUS 数据类型（标准数据类型）的对象描述，该描述从索引 l 开始。紧接着是第二段，第二段包括数据类型的对象描述和数据类型结构描述，可以分别对二者进行定义。用户组在行规文件的帮助下，可以更进一步说明第二段。OD 中对象描述“ 数据类型 ”和“ 数据类型结构描述 ”的含义可在变量存取对象（Variable Acess Objects）的定义中找到。

索引	静态类型表（ST-OD）
1 … c	包含对象：PROFIBUS 规范定义的数据类型
d … i	包含对象：数据类型 和数据类型结构描述

图 29．静态类型表的结构

4.3.2.2　静态对象字典（S-OD）

静态对象字典（S-OD）包括简单变量、数组、记录、域和事件的对象描述。

一个对象描述精确地分配给一个索引。在对象字典的对象描述中，登入的第一个索引对应于第一个对象描述，其余的对象描述按递增的索引依次排列。在对象字典的对象描述中，分配给 “ 索引<=>对象描述 ” 的数目作为 S-OD 的长度登入。

除了“ 索引<=>对象描述 ”的分配外，也可以有一个附加的“ 名称<=>对象描述 ” 的分配。名称的长度可以从 0 到 32 个字节。名称长度在 OD 对象描述的 Name Length Field（名称长度字段）中登入。如果该字段只含有一个字符“ 0 ”，则没有名称存在。对 S-OD 中定义的对象可提供存取权限。在 OD 对象描述的 AccessProtectionSupported 字段中登入是否有存取权限。

在 S-OD 中的对象描述包括一个扩展字段，该扩展字段包括对象的专用定义。连同域对象，变量存取对象，事件对象一起，定义 S-OD 中对象描述的表达。

索引	静态对象字典（S-OD）
k … n	包括对象：简单变量、数组、记录、域和事件

图 30. 静态对象字典的结构

4.3.2.3 变量表的动态表（DV-OD）

变量表的动态表包括变量表的对象描述。在 DV-OD 中的“ 变量表 ”对象和该对象描述由 DefineVariableList 服务动态地建立，由 DeleteVariableList 服务删除。在建立对象时，可为对象指定存取权限。而且，可以用装载整个 OD 的方法来建立这些对象（见 OD 管理）。

分配给变量表的对象描述的第一个索引和变量索引的数目被登入在 OD 对象描述中。变量表的基本信息也登入在对象描述中，例如存取权限、变量存取对象的数目和 S-OD 中变量存取对象的逻辑地址。

除了“ 索引<=>对象描述 ”的分配以外，也可存在一个附加的“ 名称<=>对象描述 ” 的分配（见对象的静态表）。

DV-OD 中变量表的对象描述的表达可在变量存取对象（Variable Access Objects）的定义中找到。

索引	变量表的动态表（DV-OD）
p … t	包含对象：变量表

图 31. 变量表的动态表结构

4.3.2.4 程序调用的动态表（DP-OD）

程序调用的动态表包括程序调用对象的对象描述。

在 DP-OD 中的“ 程序调用 ”对象和该对象描述由 Create Program Invocaton 服务动态地建立，由 Delete Program Invocation 服务删除。在建立对象时，可为对象指定存取权限。而且，可以用装载整个 OD 的方法来建立这些对象。（见 OD 管理）

分配给程序调用的对象描述的第一个索引和变量索引的数目登入到 OD 对象描述中。一个程序调用的对象描述包括如下信息，如存取权限、域对象的数目及其逻辑地址（索引）。

除了“索引<=>对象描述”的分配，也可存在一个附加的“名称<=>对象描述” 的分配（见对象的静态表）。

DP-OD 可以包括一个预组态的程序调用段。

DP-OD 中程序调用的对象描述的含义可在程序调用的定义中找到。

索引	程序调用的动态表（DP-OD）
v … w	预定义的程序调用
x … z	动态程序调用

图 32.　程序调用的动态表结构

4.3.3　OD 中的对象描述

对象描述被登录在 OD 中。每个对象描述包含索引、对象代码、附加的对象属性、系统对这个真实对象所规定的引用，如有需要，还包含名称和扩展。

通过索引寻址对象，使用对象描述来识别对象。对象代码是对象的标识符，它指出这个对象属于哪个类别。附加的对象属性是这个对象所特有的属性。

本地地址用于对象的内部编址。除索引外，名称选项也可用于对象的编址。扩展选项包含用于扩展选项和可选的附加对象属性的长度信息。长度信息作为第一字段被传送。长度信息的值包含扩展选项的长度，而不包含长度字段自己。长度信息的值可以在 0 到 200 个字节中变化。长度信息的值为 0 表示没有附加的对象属性。

各个对象描述映象到参数“对象描述”（打包的数据类型），而此参数用于 GetOD 服务和 PutOD 服务，此映象关系以图表方式显示在相应的子条目中。它的表达方式有两行，上一行是对象的一些属性名，下一行是登入项实例。如果行的长度不够，可以用后续行来表达。

这些属性被编码，并与它们的数据类型相一致（见 FMS 编码）。

一个对象描述的登入项的长度限制在最大的 PDU 范围内。

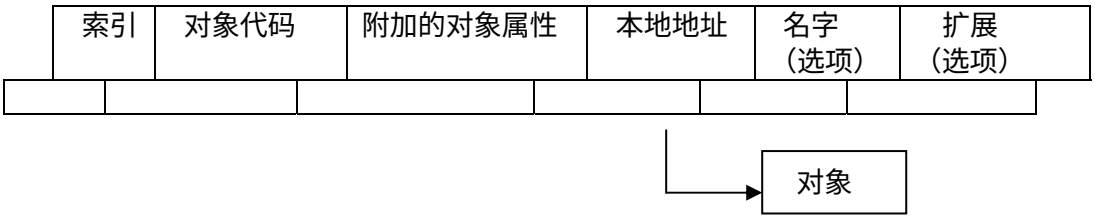


图 33.　OD 中的对象描述的结构

4.3.4 OD 对象描述

对象字典的结构通过 “ 对象字典 ” 对象描述（OD 对象描述）来描述。它在对象字典中具有 “ 0 ” 索引。通过读 OD 对象描述可以获得这个 OD 的结构描述和版本号。

4.3.4.1 属性

对象：OD 对象描述

关键属性：索引（Index）
属性：ROM/RAM 标记（Flag）
属性：名称长度 （NameLength）
属性：存取保护支持 （AccessProtectionSupported）
属性：OD 的版本号 （VersionOD）
属性：本地地址-OD-ODES （LocalAddress-OD-ODES）
属性：ST-OD-长度 （ST-OD-Length）
属性：本地地址-ST-OD （LocalAddress-ST-OD）
属性：第一索引-S-OD （FirstIndex-S-OD）
属性：S-OD-长度 （S-OD-Length）
属性：本地地址-S-OD （LocalAddress-S-OD）
属性：第一索引-DV-OD （FirstIndex-DV-OD）
属性：DV-OD-长度 （DV-OD-Length）
属性：本地地址-DV-OD （LocalAddress-DV-OD）
属性：第一索引-DP-OD （FirstIndex-DP-OD）
属性：DP-OD-长度 （DP-OD-Length）
属性：本地地址-DP-OD （LocalAddress-DP-OD）

索引 （Index）

对象描述对象字典（Object-Description Object Dictionary）作为对象，它的索引总是等于零。

ROM/RAM 标记 （ROM/RAM-Flag）

此属性是布尔数，用来说明 OD 是否允许被更改。

假 （False） <=> OD 不允许被更改
真 （True） <=> OD 允许被更改

名称长度 （NameLength）

这个字段给出名称的长度，长度可取值范围是 “ 0…..32”。
0<=>说明没使用名称。

存取保护支持 （AccessProtectionSupported）

此属性状态说明存取权限。

真（True）： 支持口令，保护存取组和所有通信伙伴的存取权
假（False）： 允许每个通信伙伴存取所有对象

OD 的版本号 （VersionOD）

给出 OD 的版本号。

本地地址-OD-ODES （LocalAddress-OD-ODES）

此属性是对真实 OD 对象描述的系统专用引用，用于内部编址。
如果没有这类表达，则将此属性设置为十六进制值 FFFFFFFF。

ST-OD-长度 （ST-OD-Length）
此属性给出在静态类型表中有效登入项的最大数。静态类型表的第一个可利用的索引值为 1，静态类型表的最高可利用的索引值等于它的长度。

本地地址-ST-OD （LocalAddress-ST-OD）
此属性是对真实 ST-OD 的系统专用引用，用于内部编址。
如果没有这类表达式，则将此属性设置为十六进制值 FFFFFFFF。

第一索引-S-OD （FirstIndex-S-OD）
这是静态对象字典中第一个可用的索引。

S-OD-长度 （S-OD-Length）
此属性给出在静态对象字典中有效登入项的最大数。
最高可用的索引值等于：起始索引值 + 长度 – 1 。
本地地址-S-OD （LocalAddress-S-OD）
此属性是对真实 S-OD 的系统专用引用，用于内部编址。
如果没有这类表达式，则将此属性设置为十六进制值 FFFFFFFF。

第一索引-DV-OD （FirstIndex-DV-OD）
这是在变量表的动态表中第一个可用的索引。
0<=>说明没有变量表的动态表。

DV-OD-长度 （DV-OD-Length）
此属性给出在变量表的动态表中有效登入项的最大数。
最高可用索引值等于：起始索引值 + 长度 – 1。

本地地址-DV-OD （LocalAddress-DV-OD）
此属性是对真实 DV-OD 的系统专用引用，用于内部编址。
如果没有这类表达式，则将此属性设置为十六进制值 FFFFFFFF。

第一索引-DP-OD （FirstIndex-DP-OD）
这是在程序调用的动态表中第一个可用的索引。
0<=>说明没有程序调用的动态表。

DP-OD-长度 （DP-OD-Length）
此属性给出在程序调用的动态表中有效登入项的最大数。
最高可用索引值等于：起始索引值 + 长度 – 1。
本地地址-DP-OD （LocalAddress-DP-OD）
此属性是对真实的 DP-OD 的系统专用引用，用于内部编址。
如果没有这类表达式，则将此属性设置为十六进制值 FFFFFFFF。

4.3.4.2 在传送中 OD 的表达方式

Index	ROM/RAM Flag	Name Length	Access Protection Supported	Version OD	Local Address OD-ODES

//

ST-OD Length	Local Address ST-OD	First Index S-OD	S-OD Length	Local Address S-OD

//

First Index DV-OD	DV-OD Length	Local Address DV-OD	First Index DP-OD	DP-OD Length	Local Address DP-OD

图 34. OD 对象描述的结构

4.3.4.3 空对象字典

因为对象字典是可以装载的，所以如果一个对象字典还没被装载，那末至少有一个空对象字典应存在。

用预置一个 OD 对象描述的属性（如下表所示）和 ST-OD 来定义一个空对象描述。在 ST-OD 中所有标准数据类型组态为空对象（Null Object）。

表 8. 一个空对象字典的 OD 对象描述

属性	值
ROM/RAM-Flag	True
NameLength	0
AccessProtectionSpported	False
VersionOD	0
ST-OD-Length	14
FirstIndex-S-OD	15
S-OD-Length	0
FirstIndex-DV-OD	0
DV-OD-Length	0
FirstIndex-DP-OD	0
DP-OD-Length	0

4.3.5 OD 对象

4.3.5.1 属性

对象：OD

关键属性：由 VFD 隐含(implicit by VFD)

属性：OD 对象描述 (OD object description)

属性：对象描述表(list of object description)

由 VFD 隐含

登录在 FMS 通信关系表中的通信关系（见上下关系管理），它指向指定的对象字典。

OD 对象描述

包含对象字典的对象描述。

对象描述表

包含以下对象的对象描述：

- 域
- 程序调用
- 简单变量
- 数组
- 记录
- 变量表
- 数据类型
- 数据类型结构描述
- 事件
- 空（Null）

4.3.6 OD 服务

OD 服务对如下对象的对象描述起作用，这些对象是：域、程序调用、简单变量、数组、记录、变量表、事件、对象字典、数据类型和数据类型结构描述。

定义如下的 OD 服务：

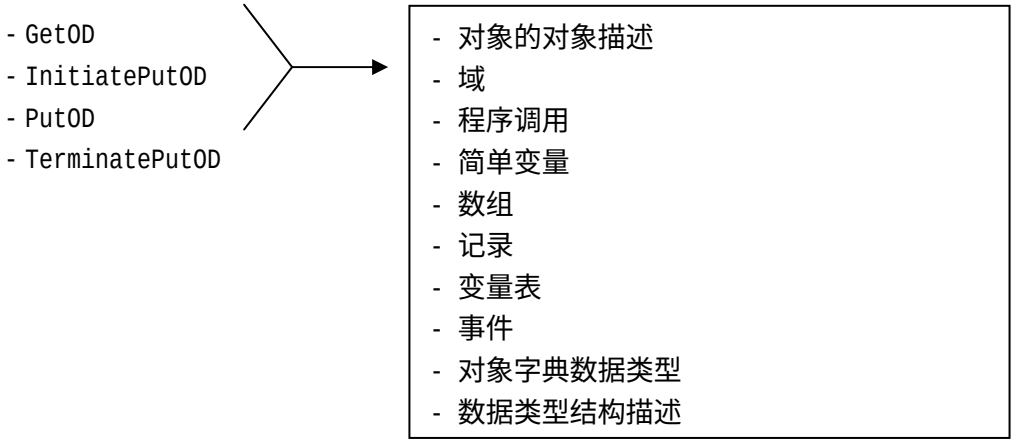


图 35. OD 服务

GetOD 服务用于读一个或几个对象描述。PutOD 服务用于写一个或几个对象描述。其中，能用 GetOD 或 PutOD 服务传递的对象描述的数目取决于它们的长度和最大有效 PDU 的大小。

InnitiatePutOD 服务用于启动对 OD 的写，而 TerminatePutOD 服务用于终止对 OD 的写。

4.3.6.1 获得对象字典（GetOD）服务

GetOD 服务用于读一个或几个对象描述。GetOD 服务区分为短形式和长形式。GetOD 服务的长形式是可选择的。为了读单个对象描述，需要给出它的索引或它的名称。为了读几个对象或全部对象的描述，需要给出第一个所要求的对象描述的索引（起始索引）。为了读出整个 OD，需要重复使用 GetOD 服务。

表 9. GetOD

参数名	.req .ind	.res .con
变元（Argument）	M	
AllAttributes	M	
AccessSpecification	M	
Index	S	
VariableName	S	
VariableListName	S	
DomainName	S	
PI-Name	S	
EventName	S	
StartIndex	S	
结果（+）　（Result +）		S
List of Object Description		M
MoreFollows		M
结果（-）　（Result - ）		S
ErrorType		M

变元（Argument）

变元包含服务调用的服务专用参数。

全部属性（AllAttributes）

此参数说明对象描述的传送是短形式（false）还是长形式（true）。

在短形式中不包括以下对象属性：

- Description
- Password
- AccessGroups
- AccessRights
- LocalAddress
- Name
- LocalAddress-OD-ODES
- LocalAddress-ST-OD
- LocalAddress-S-OD
- LocalAddress-DV-OD
- LocalAddress-DP-OD
- Extension

存取说明　（AccessSpecification）

此参数说明哪一个对象描述被存取。

索引（Index）

对象描述的索引。

变量名（VariableName）

变量名。

变量表名（VariableListName）

变量表名。

PI-名称（PI-Name）

程序调用名。

域名（DomainName）

域名。

事件名（EventName）

事件名。

起始索引（StartIndex）

开始被读的那个对象描述的索引。

结果（+）（Result （+））

此参数表示服务执行成功。

对象描述表（List of object description）

对逻辑存取和按名称存取的一个或多个完整的对象描述，对象描述的数量取决于许可的最大 PDU 值和单个对象描述的长度。如果存取权限不允许（存取保护为‘假’(false)），那么属性口令、存取组和存取权都是没有意义的。如果在 OD 中被选择的索引是个空（Null）对象，则返回适合这个空对象的代码。如果使用起始索引，那么只有在空对象代表不被支持的数据类型时，才返回这个空对象的代码，而在所有其它情况下，空对象可以被忽略。

更多的数据（MoreFollows）

此参数说明在读多个对象描述（Start Index）后是否存在有其它的对象描述（Further Object Description）。对于单个对象描述的存取，这个参数始终为‘假’(false)。

结果（-） （Result （-））

结果（-）参数表示这个服务没有成功地执行。

出错类型（ErrorType）

此参数包含服务没有成功执行的原因信息。

4.3.6.2 放置对象字典（PutOD） 服务

用 PutOD 服务写一个或多个对象描述。在开始更改 OD 前，用户可以采取适当的测定（例如把一个过程进入安全状态）。通信不测试新对象字典的似真性（Plausibility）。

写 OD 采用一个序列 InitiatePutOD 服务、一个或多个 PutOD 服务和一个 TerminatePutOD 服务来实现。为此，我们要区分自由交互方式的装载和非自由交互方式的装载（OD-LOADING-NON-INTERACTING 和 OD-LOADING-INTERACTING）。而非自由交互方式的装载区分新装载（完整的）和后装载（部分的）。如果必要的话，使用 TerminatePutOD 服务建立新的通信对象和它们的状态机。

自由交互的装载（OD-LOADING-NON-INTERACTING）

客户机用参数 Consequence = 0 与服务器通信时，不管是更改还是扩展，对于其它的通信伙伴都是自由交互的。在这种情况时，通信关系是连续的。

例如:　自由交互的更改是：

- 在静态段中增添新的对象描述。
- 删除那些不再在通信系统中的站点的专用对象描述。

通信并不检查对 OD 的写是否是自由交互的。自由交互的装载的启动使用 InitiatePutOD 服务，它的参数 Consequence = 0。接着，使用 PutOD 服务装载对象描述。只要还没有使用 TerminatePutOD 服务来终止这个装载过程，那么，再一次的 InitiatePutOD 服务将遭拒绝。

非自由交互的装载（OD-LOADING-INTERACTING）

非自由交互的装载使用 InitiatePutOD 服务启动，它的参数 Consequence = 1（后装载），或者等于 2（新装载）。这时，用户将释放所有其它的通信关系（具有原因代码 6 的中止将被发出），接着，使用 PutOD 服务装载对象描述。作为客户机用户的任务还要在 OD 中登录新的版本号。通过 TerminatePutOD 服务结束这个装载。此后，才可以重新建立所有的通信关系。在通信关系重新建立的同时，传送新的 OD 版本号。于是，通信伙伴能够识别已经更改的 OD。对于后装载而言，装载了 OD 的一些单独的部分。在接收到 InitiatePutOD 服务，并且它的参数 Consequence = 2 时，已存在的 OD 被删除，再进行完整的新的 OD 装载。

4.3.6.2.1　启动放置对象字典（InitiatePutOD）

InitiatePutOD 服务用于打开自由交互的装载过程或非自由交互的装载过程。

表 10.　InitiatePutOD

参数名	.req .ind	.res .con
变元	M	
后果	M	
结果　(+)		S
结果　(-)		S
出错类型		M

变元

变元包含此服务调用的服务专用参数。

后果（Consequence）

此参数表示对 OD 的更改是对其它通信伙伴使用自由交互装载（Consequence = 0）还是使用非自由交互的部分 OD 再装载（Consequence = 1）或是使用非自由交互的全部 OD 新装载（Consequence = 2）。

0 < = > 自由交互的装载

1 < = > 再装载，非自由交互

2 < = > 新装载，非自由交互

结果 （+）

结果（+）参数表示服务执行成功。

结果 （-）

结果（-）参数表示此服务没有成功地执行。

出错类型

此参数包含了服务没有成功执行的原因信息。

4.3.6.2.2 放置对象字典（PutOD）

PutOD 服务用于把一个或多个对象描述写入 OD 中。

表 11. Put OD

参数名	.req .ind	.res .con
变元	M	
对象描述表	M	
结果 (+)		S
结果(-)		S
出错类型		M

变元

变元包含此服务调用的服务专用参数。

对象描述表

被登录到 OD 中的一个或多个对象描述。如果存取权未被许可（存取保护为“假”），那么，属性口令、存取组和存取权是没有意义的。用 PutOD 服务也能够删除对象描述，这时，要传送一个空的对象描述，它的结构如下：

- 被删除的对象描述的索引
- 对象代码 = Null 对象。

索引	对象代码
245	Null

图 36. 空对象描述 （例）

结果（+）

结果（+）参数表示服务成功执行。

结果（-）

结果（-）参数表示此服务没有成功地执行。这时，没有对象描述写入到 OD 中。

出错类型

此参数包含服务没有成功执行的原因信息。

4.3.6.2.3 终止放置对象字典（Terminate PutOD）

TerminatePutOD 服务用于结束 OD 的装载过程。生成带有状态机的对象。

表 12. TerminatePutOD

参数名	.req .ind	.res .con
变元	M	
结果 （+）		S
结果 （-）		S
出错类型		M
索引		C

变元

变元没有包含服务专用参数。

结果（+）

结果（+）参数表示服务成功执行。

结果（-）

结果（-）参数表示此服务没有成功地执行。如果出错发生在对象生成期间，那么原有的状态被保存。

出错类型

此参数包含服务没有成功执行的原因信息。

索引

此参数给出没能成功地生成的对象的索引。

4.3.7 状态机

4.3.7.1 状态机描述

OD-LOADING-NON-INTERACTING

在这个状态时，不允许执行 InitiatePutOD 服务。对 DV-OD 或 DP-OD 的 PutOD 服务是不允许的。

OD-LOADING-INTERACTING

除了已经接收到 InitiatePutOD 服务的连接以外，其它所有的连接均被中止，并且用原

因代码 6 的中止来拒绝建立其它的连接。在这个连接上只允许以下的服务：

- Initiate
- About
- Reject
- Status
- Identify
- PhysRead
- PhysWrite
- GetOD
- InitiatePutOD
- PutOD
- TerminatePutOD

OD-READY

在此状态时可以正常地使用所有的连接和服务。

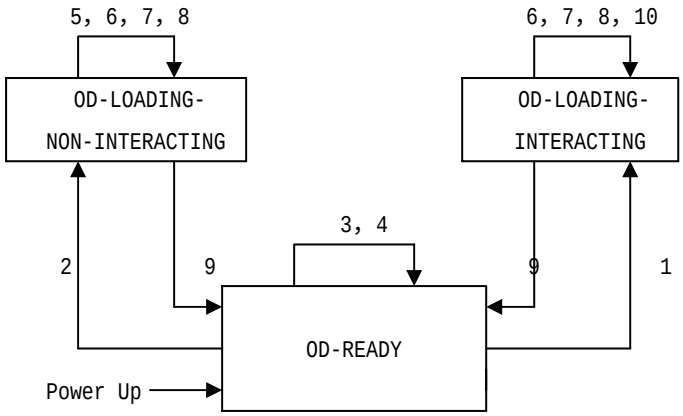


图 37．状态机

4.3.7.2 状态转换

```
    事件
      \退出条件
      =>采取的动作
-----------------------------------------------------------------------------
1   InitiatePutOD.ind
    \Consequence > 0
    = > .res (+)
2   InitiatePutOD.ind
    \Consequence = 0
    = > .res (+)
3   PutOd.ind
    = > .res (-)     对象状态冲突
4   TerminatePutOD.ind
    = > .res (-)     对象状态冲突
5   InitiatePutOD.ind
    \Consequence > 0
    = > .res (-)     对象状态冲突
6   InitiatePutOD.ind
    \Consequence = 0
    = > .res (-)     对象状态冲突
7   PutOd.ind
    = > .res (+)
8   TerminatePutOD.ind
    \OD 不能使用
    = > .res (-) 操作问题
9   TerminatePutOD.ind
    \OD 没问题
    = > .res (+)
10.Initiate PutOD.ind
    \Consequence>0
    = >.res(+)
```

4.3.8 放置对象字典（PutOD）时序举例

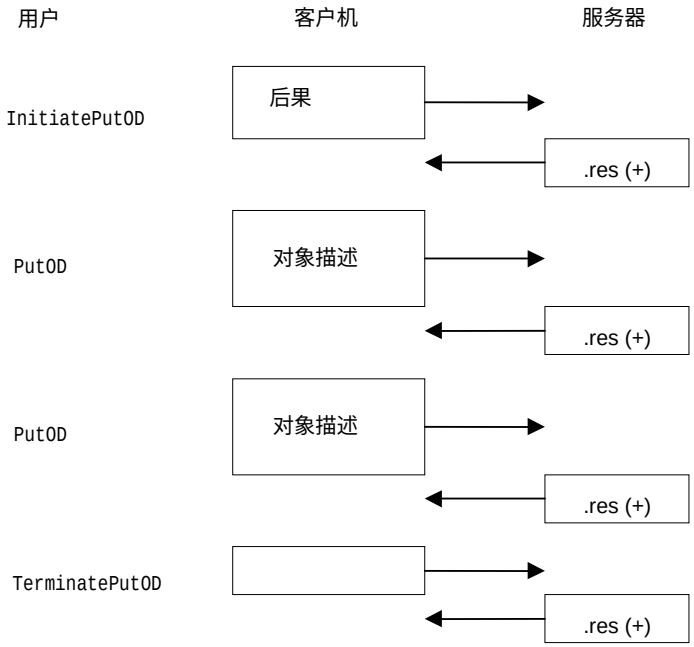


图 38. PutOD 时序举例

4.4 上下关系管理

4.4.1 模型描述

上下关系包含所有有关通信关系的约定。上下关系管理服务用于建立连接、释放连接和拒绝不适当的服务。上下关系所需要的数据保存在 OD 对象描述和 FMS 的通信关系表里。事务处理对象提供给各个通信关系。事务处理对象用于需确认服务原语的管理。

4.4.2 FMS 通信关系表（FMS CRL） 对象

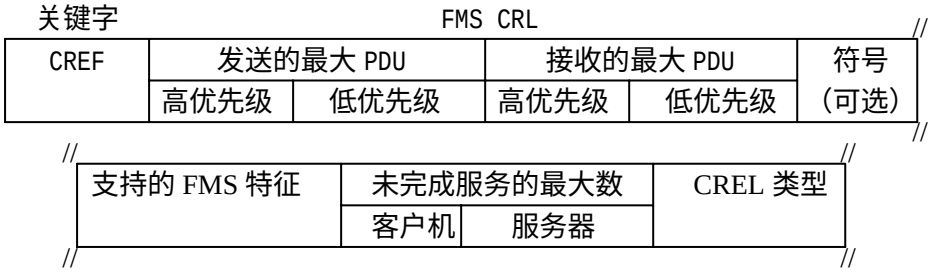
FMS 通信关系表（FMS CRL）包含一个设备所有通信关系的 FMS 专用描述，与使用的时间无关。FMS CRL 由登入项组成。各个登入项通过通信关系的关键索引（CREF）寻址。各个登入项由静态部分和动态部分所组成。登入项包含相关联的通信关系的全部 FMS 专用描述。

FMS CRL 首部包含 FMS CRL 结构的信息。FMS CRL 首部使用 CREF 0 寻址。

关键字	FMS CRL 首部		
0	FMS CRL 登入项的数目	符号长度	支持的 VFD 指针

图 39. FMS CRL 首部的结构

FMS CRL 登入项的静态部分



FMS CRL 登入项的动态部分

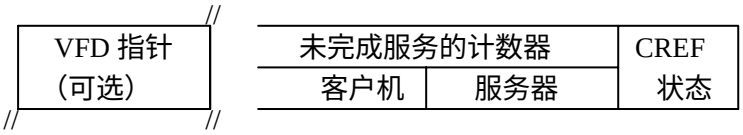


图 40.FMS CRL 登入项的结构

FMS CRL 首部（FMS CRL Header）：

FMS CRL 登入项的数目（Number of FMS CRL entries）

此属性规定了在 FMS CRL 中 FMS CRL 登入项的数目。

符号长度（Symbol Length）

此属性规定在 FMS CRL 中符号的长度。它的值只允许为 0...32。

0<=> 没有符号

支持的 VFD 指针（VFD Pointer Supported）

此属性表示在 FMS CRL 中是否支持多个 VFD。

FMS CRL 登入项的静态部分（Static part of the FMS CRL entry）：

CREF

通信引用（CREF）是通信关系在本地的唯一地址。

高优先级发送的最大 PDU（Max PDU Sending High Prio）

此属性规定了能在此通信关系中被处理的带高优先级发送的 FMS PDU 的最大长度。

低优先级发送的 PDU（Max PDU Sending Low Prio）

此属性规定了能在此通信关系中被处理的带低优先级发送的 FMS PDU 的最大长度。

高优先级接收的最大 PDU（Max PDU Receiving High Prio）

此属性规定了能在此通信关系中被处理的带高优先级接收的 FMS PDU 的最大长度。

低优先级接收的最大 PDU（Max PDU Receiving Low Prio）

此属性规定了能在此通信关系中被处理的带低优先级接收的 FMS PDU 的最大长度。

支持的 FMS 特征（FMS Features Supported）

此属性将识别在通信关系中可选择的 FMS 服务和可选择的支持。各个可选择服务有两位（bit），专门用于指定所支持的服务原语。对这些选择专用的两位用于确定哪一个服务原语的选择被许可。当选择 FMS 特征时，将会看到对主/从和对象，以及对 OD 对象所赋予的服务定义。

表 13.　支持的 FMS 特征属性

服务	原语	位[n]	原语	位[m]
GetOD （Long form）	.req , .con	0	.ind , .res	24
UnsolicitedStatus	.req	1	.ind	25
InitiatePutOD	.req , .con	2	,ind , .res	26
PutOD	.req , .con	2	.ind , .res	26
TerminatePutOD	.req , .con	2	.ind , .res	26
InitiateDownloadSequence	.req , .con	3	.ind , .res	27
DownloadSegment	.ind , .res	3	.req , .con	27
TerminateDownloadSequence	.ind , .res	3	.req , .con	27
InitiateUploadSequence	.req , .con	4	.ind , .res	28
UploadSegment	.req , .con	4	.ind , .res	28
TerminateUploadSequence	.req , .con	4	.ind , .res	28
RequestDomainDownload	.req , .con	5	.ind , .res	29
RequestDomainUpload	.req , .con	6	.ind , .res	30
CreateProgramInvocation	.req , .con	7	,ind , .res	31
DeleteProgramInvocation	.req , .con	7	.ind , .res	31
Start	.req , .con	8	.ind , .res	32
Stop	.req , .con	8	.ind , .res	32
Resume	.req , .con	8	.ind , .res	32
Reset	.req , .con	8	.ind , .res	32
Kill	.req , .con	9	.ind , .res	33
Read	.req , .con	10	.ind , .res	34
Write	.req , .con	11	.ind , .res	35
ReadWithType	.req , .con	12	.ind , .res	36
WriteWithType	.req , .con	13	.ind , .res	37
Physread	.req , .con	14	.ind , .res	38
PhysWrite	.req , .con	15	.ind , .res	39
InformationReport	.req	16	.ind	40
InformationReportWithType	.req	17	.ind	41
DefineVariableList	.req , .con	18	.ind , .res	42
DeleteVariableList	.req , .con	18	.ind , .res	42
EventNotification	.req ,	19	.ind	43
EventNotificationWithType	.req ,	20	.ind	44
AcknowledgeEventNotification	.req , .con	21	.ind , .res	45
AlterEventConditionMonitoring	.req , .con	22	.ind , .res	46
选项	原语	位[n]	原语	位[m]
用名称编址	.req	23	.ind	47

注释:
[n] : 0 到 23
[m] : 24 到 47

未完成的客户机服务的最大数目（Max Outstanding Services Client）

此属性规定了在此通信关系中作为客户机所允许的未完成的需确认服务的最大数目。

未完成的服务器服务的最大数目（Max Outstanding services server）

此属性规定了在此通信关系中作为服务器所允许的未完成的需确认服务的最大数目。

CREL 类型（CREL Type）

此属性表示通信关系的类型。

- 真 （True） ： 面向连接 （connection-oriented）
- 假 （False）： 无连接 （connectionless）

符号（Symbol）：

此属性规定了通信引用的符号名。在 FMS CRL 首部定义这个符号的存在和长度。

VFD 指针（VFD Pointer）

此指针应指向指定的 VFD。

FMS CRL 登入项的动态部分（Dynamic part of the FMS CRL entry）：

客户机用于未完成服务的计数器（Outstanding Services Counter Client（OSCC））

此属性规定了在通信关系中作为客户机当前未完成的需确认服务的数目。

服务器用于未完成服务的计数器（Outstanding services Counter Server（OSCS））

此属性规定了在通信关系中作为服务器当前未完成的需确认服务的数目。

CREL 状态（CREL State）

此属性规定通信关系的状态。

一个面向连接的通信关系可以是下面状态中的一个：

- CONNECTION-NOT-ESTABLISHED （连接未建立）
- CONNECTION-ESTABLISHING（CALLING） （在调用的连接正在建立）
- CONNECTION-ESTABLISHING（CALLED） （被调用的连接正在建立）
- CONNECTION-ESTABLISHED （连接已建立）

一个无连接的通信关系总是具有 CONNECTIONLESS-CLIENT（无连接客户机）或 CONNECTIONLESS-SERVER（无连接服务器）的状态。（见上下关系管理状态机）。

4.4.3 事务处理对象

当接收到一个需确认服务的指示原语时，要建立一个事务处理对象。这个事务处理对象与相应的服务原语有着唯一的关系。在相应的响应原语已被发送后，要删除这个事务处理对象。通过 Invoke ID 和通信引用两者结合唯一地标识这个事务处理对象。

用于服务器的各个通信关系的事务处理对象的最大数目由在 FMS CRL 中的 Max Outstanding Services Server 属性来规定。

下面的服务对事务处理对象起作用：

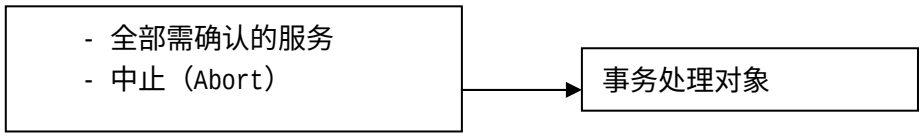


图 41. 事务处理对象

4.4.3.1 属性

对象：事务处理对象
关键属性：Invoke ID 和 CREF
属性：需确认的服务指示

Invoke ID
此属性标识在通信关系中的事务处理对象。

CREF
此属性标识被事务处理对象使用的通信关系。

需确认的服务指示（confirmed service indication）
服务标识符和挂起服务（pending service）的变元.

4.4.3.2 状态机

状态机的描述

NON-EXISTENT
事务处理对象不存在。

PENDING
一个需确认的服务指示原语已被接收，但是相应的响应原语还没被发送。

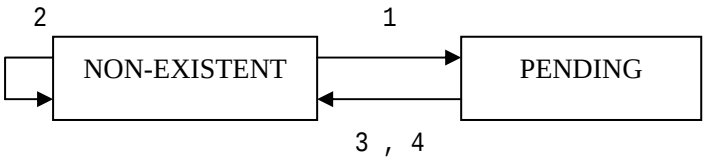


图 42. 事务处理对象状态机

状态转换

1 需确认的 service.ind （事务处理对象数 < max ）
2 需确认的 service.ind （事务处理对象数 = max ）
3 需确认的 service.res
4 Abort.ind

4.4.4 上下关系管理服务
上下关系管理服务的组成
- 启动 （Initiate）
- 中止 （Abort）
- 拒绝 （Reject）

4.4.4.1 启动

这项服务用于在两个通信伙伴间建立一个连接并交换有关信息,信息包括支持的服务、支持的选择、最大的 PDU 长度和当前 OD 的版本。

表 14. 启动

参数名	.req	.ind	.res	.con
变元	M	M=		
OD 的版本 *)	M	M=		
行规号 *)	M	M=		
支持的存取保护 *)	M	M=		
口令 *)	M	M=		
存取组 *)	M	M=		
高优先级发送的最大 PDU *)				
低优先级发送的最大 PDU *)				
高优先级接收的最大 PDU *)				
低优先级接收的最大 PDU *)				
支持的 FMS 特征 *)				
结果 (+)			S	S=
OD 版本 **)			M	M=
行规号 **)			M	M=
支持存取保护 **)			M	M=
口令 **)			M	M=
存取组 **)			M	M=
结果 (-)			S	S=
出错代码			M	M
高优先级发送的最大 PDU **)				M
低优先级发送的最大 PDU **)				M
高优先级接收的最大 PDU **)				M
低优先级接收的最大 PDU **)				M
支持的 FMS 特征 **)				M
备注： *) 调用 (Calling) **) 被调用 (Called)				

变元 (Arqument) :

变元传送服务请求的服务专用参数。

OD 的版本 (Version OD)

此参数规定客户机的对象字典的版本号。

行规号 (调用) (Profile Number (Calling))

此参数规定客户机的行规号。

支持的存取保护 (调用) (Access Protection Supported (Calling))

此参数包含客户的 OD 对象描述所支持的存取保护属性。

口令（调用）（Password （Calling））

此参数规定在此通信关系上存取服务器的所有对象所用的口令。如果用口令的存取在此通信关系上未被使用，那么口令的参数值应等于 0。

存取组（调用）（Access Groups （Calling））

此参数规定在特定存取组里的客户机成员。成员适用于在此通信关系上对服务器的所有对象的存取。

高优先级发送的最大 PDU（调用）（Max PDU Sending High Prio （Calling））

此参数规定能在此通信关系上被处理的高优先级发送的 FMS PDU 的最大长度。

它被调用的 FMS 发送，不是原语的一部分。

低优先级发送的最大 PDU（调用）（Max PDU Sending Low Prio （Calling））

此参数规定能在此通信关系上被处理的用低优先级发送的 FMS PDU 的最大长度。

它被调用的 FMS 发送，不是原语的一部分。

高优先级接收的最大 PDU（（调用）（Max PDU Receiving High Prio （Calling））

此参数规定能在此通信关系上被处理的用高优先级接收的 FMS PDU 的最大长度。

它被调用的 FMS 发送，不是原语的一部分。

低优先级接收的最大 PDU（调用）（Max PDU Receiving Low Prio （Calling））

此参数规定能在此通信关系上被处理的用低优先级接收的 FMS PDU 的最大长度。

它被调用的 FMS 发送，不是原语的一部分。

支持的 FMS 特征（调用）（FMS Features Supported （Calling））

此参数标识可选的 FMS 服务和由客户机支持的选项（见 FMS CRL）。此参数由调用的 FMS 发送，它不是原语的一部分。

结果（+）（Result（+））：

此参数表示请求的服务已成功地执行。

OD 版本（被调用）（Version OD （Called））

此参数规定服务器的对象字典的版本。

行规号（被调用）（Profile Number（Called））

此参数规定服务器的行规号。

支持存取保护(被调用)（Access Protection Supported （Called））

此参数包含服务器的 OD 对象描述所支持的存取保护的属性。

口令（被调用）（Password（Called））

此参数规定在此通信关系上存取客户机的所有对象所用的口令。如果带口令的存取在此通信关系上未被使用，那么口令的参数值应等于 0。

存取组（被调用）(Access Group （Called))

此参数规定在特定存取组里的服务器的成员。成员适用于在这个通信关系上对客户机的所有对象的存取。

结果（-）(Result（-)）:

此参数表示这个服务请求失败。

出错代码（Error Code）

此参数提供失败的原因。

-Max PDU Size Insufficient

最大的 PDU 长度不能满足通信需要。

- Feature Not Supported

  请求的服务或选项不能被服务器所支持。

- User Initiate Denied

  FMS 用户拒绝建立连接。

- Version OD incompatible

  通信伙伴的对象字典（源 OD 和 远程 OD）的版本不兼容。

- Password Error

  已经存在用相同口令建立的通信关系。

- Profile Number Incompatible

  客户机的行规不被服务器所支持。

- Other

  没有列入上面的其它原因。

高优先级发送的最大 PDU（被调用）(Max PDU Sending High Prio （Called))

此参数规定能在此通信关系上被处理的用高优先级发送的 FMS PDU 的最大长度。它由被调用的 FMS 发送，并且它只是 Initiate.con 原语的一部分。

低优先级接收的最大 PDU（被调用）(Max PDU Sending Low Prio （Called))

此参数规定能在此通信关系上被处理的用低优先级发送的 FMS PDU 的最大长度。此参数由被调用的 FMS 发送，并且它只是 Initiate.con 原语的一部分。

高优先级接收的最大 PDU（被调用）(Max PDU Receiving High Prio （Called))

此参数规定能在此通信关系上被处理的用高优先级接收的 FMS PDU 的最大长度。此参数被调用的 FMS 发送，并且它只是 Initiate.con 原语的一部分。

低优先级接收的最大 PDU（被调用）(Max PDU Receiving Low Prio （Called))

此参数规定能在此通信关系上被处理的用低优先级接收的 FMS PDU 的最大长度。此参数由被调用的 FMS 发送，并且它只是 Initiate.con 原语的一部分。

支持的 FMS 特性（被调用）(FMS Feature Supported （Called))

此参数标识可选的 FMS 服务和服务器支持的选项（见 FMS CRL)。此参数通过被调用的 FMS 发送，且它只是 Initiate.con 原语的一部分。

4.4.4.2 中止

此服务用于释放在两个通信伙伴之间已经存在的通信关系。客户机和服务器都可以释放此连接。

表 15. 中止

参数名	.Req	.ind
变元	M	M
本地生成的		M
中止标识符	M	M=
原因代码	M	M=
中止细节	U	C

变元（Argument）

变元传送服务请求的服务专用参数。

本地生成的（Locally Generated）

此参数表示这个中止服务是由本地生成的还是由通信伙伴生成的。

如果 Abort Identifier 参数的值是 FMS，并且 Reason Code 参数的值是 FMS CRLError，那么此参数的值不允许是‘假’(false)。

中止标识符（Abort Identifier）

此参数表示在哪里已检测到中止的原因。

0 < = > USER （用户）

1 < = > FMS

2 < = > LLI

3 < = > LAYER2 （层 2）

原因代码（Reason Code）

此参数规定中止的原因。

如果 Abort Identifier 参数的值是 USER，那么将适用下面的值：

- ABT_RC1 < = > Disconnect
  由 FMS 用户释放了此连接。
- ABT_RC2 < = > Version OD incompatible
  通信伙伴的对象字典（源 OD 和远程 OD）的版本不兼容。
- ABT_RC3 < = > Password Error
  已经存在用相同口令建立的通信关系。
- ABT_RC4 < = > Profile Number Incompatible
  服务器的行规不被客户机所支持。
- ABT_RC5 < = > Limited Services Permitted
  VFD 处在逻辑状态 LIMITED-SERVICES-PERMITTED 中
- ABT_RC6 < = > OD-Loading-interacting
  当前在活动的是非自由交互方式的 PutOD 服务。

如果 Abort Identifier 参数的值是 FMS，那么将适用下面的值：

- ABT_RC1 < = > FMS CRL Error
  有错误的 FMS CRL 登入项。
- ABT_RC2 < = > User Error
  接收到来自 FMS 用户的不正确的、未知的或有错误的服务原语。
  ABT_RC3 < = > FMS PDU Error
  接收到来自 LLI 的未知的或有错误的 FMS PDU。
- ABT_RC4 < = > Cnnection State Conflict LLI
  不正确的 LLI 服务原语。
- ABT_RC5 < = > LLI Error
  未知的或有错误的 LLI 服务原语。
- ABT_RC6 < = > PDU Size
  PDU 长度超过最大的 PDU 长度。
- ABT_RC7 < = > Feature Not Supported
  接收到来自 LLI 的 SERVICE_REQ_PDU,但作为服务器不支持此服务或选项。
  （见 FMS CRL 中 FMS 特征支持属性）
- ABT_RC8 < = > Invoke ID Error Response
  接收到来自 FMS 用户的需确认的 service.res，但 Invoke ID 不存在或者接收到来自 LLI 的 CONFIRMED-SERVICE_RES_PDU, 但 Invoke ID 不存在。
- ABT_RC9 < = >Max Services Overflow
  接收到来自 LLI 的 CONFIRMED-SERVICE_REQ_PDU,但未完成的服务计数器服务器大于等于未完成的服务服务器（Outstanding Services Counter Server >= Outstanding services Server ）。
- ABT_RC10 < = > Connection State Conflict FMS
  接收到来自 LLI 的 INITIATE_REQ_PDU 或 INITIATE_RES_PDU。
- ABT_RC11 < = > Service Error
  在响应中的服务与在指示中的服务不匹配，或者在确认中的服务与在请求中的服务不匹配。
- ABT_RC12 < = > Invoke ID Error Request
  接收到来自 LLI 的 CONFIRMED-SERVICE_REQ_PDU,但 Invoke ID 已经存在。
- ABT_RC13 < = > FMS disabled
  FMS 没有为数据发送作好准备。

如果 Abort Identifier 参数的值为 LLI 或 LAYER2，那么 LLI 将提供原因代码（Reason Code）参数。

中止细节（Abort Detail）

此参数包含有关中止原因的附加信息（最大为 16 个八位位组）。在行规中定义了来自应用的出错报告的含义。

4.4.4.3 拒绝

FMS 使用拒绝服务拒绝不正确的 PDU，或者拒绝来自 FMS 用户的不正确的 FMS-Service_Request 或 FMS-Service_Response。

表 16. 拒绝

参数名	.ind
变元	
本地检测	M
初始的调用 ID	C
拒绝 PDU 类型	M
拒绝代码	M

变元（Argument）：

变元传送此服务请求的服务专用参数。

本地检测（Detected Here）

此参数表示是否在本地已经检测到出错（‘真’即‘true’）。仅当 Reject PDU Type 参数的值为 Confirmed-Response-PDU 并且 Reject Code 参数的值为 PDU Size 时，此参数的值才可以为‘假’（即‘false’）。

初始的调用 ID（Original Invoke ID）

此参数是被拒绝的 PDU 或被拒绝的 FMS 服务的初始调用 ID。

拒绝 PDU 类型（Reject PDU Type）

此参数表示被拒绝的 PDU 的类型或由被拒绝的 FMS 服务所引起的 PDU 的类型。此参数许可的值如下：

1 < = > Confirmed-Request-PDU　3 < = > Unconfirmed-PDU

2 < = > Confirmed-Response-PDU　4 < = > Unknown type of PDU

拒绝代码（Reject Code）

此参数规定拒绝的原因。

1 < = > Invoke-ID-Exists
来自 FMS 用户的需确认的 service.req 已经收到，并且调用 ID 已存在。

2 < = > Max-Services-Overflow
来自 FMS 用户的需确认的 service.req 已经收到，并且未完成的服务计数器客户机大于等于未完成的服务客户机。

3 < = > Feature-Not-Supported-connection-Oriented
来自 FMS 用户的 Service.req 已经收到，并且作为客户机不支持此服务或选项。（见在 FMS CRL 中 FMS Feature Supported 属性）

4 < = > Feature-Not-Supported-Connectionless
来自 FMS 用户的无需确认的 service.req 已经收到，并且作为客户机不支持此服务或选项（见在 FMS CRL 中 FMS Feature Supported 属性），或者来自 FMS 用户的需确认的 service.req 已经收到。

5 < = > PDU-Size
PDU 的长度超过了最大的 PDU 长度。

6 < = > User-Error-Connectionless
从 FMS 用户收到不正确的或有错误的服务原语。

0 < = > Other
除上面以外的其它的原因。

4.4.5　连接建立中的测试

4.4.5.1 在 FMS 中的上下关系测试

在接收到 INITIATE_REQ_PDU 时，服务器的 FMS 将检查通信伙伴的 FMS 上下关系（远程上下关系），即检查远程上下关系是否与为此连接在 FMS CRL 中定义的它自身的 FMS 上下关系（本地上下关系）兼容。假定本地的 FMS 上下关系是正确的。下图表定义了本地与远程上下关系的兼容性（见图 43）。在图 43 中无含意的字段留作空格（例如本地上下关系“ Max PDU Sending High Prio”与远程上下关系“ FMS Feature Supported”的组合），这些组合不检查。

<table>
<tr><th rowspan="4">远程上下关系</th><th colspan="6">本地上下关系</th></tr>
<tr><th colspan="4">最大 PDU</th><th colspan="2" rowspan="2">支持的 FMS 特性</th></tr>
<tr><th colspan="2">发送</th><th colspan="2">接收</th></tr>
<tr><th>高优先级</th><th>低优先级</th><th>高优先级</th><th>低优先级</th><th>.req [n]
0 1</th><th>.ind [m]
0 1</th></tr>
<tr><td>Max PDU Sending High Prio</td><td colspan="2" rowspan="2"></td><td colspan="2">≥</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Max PDU Sending Low Prio</td><td colspan="2">≥</td></tr>
<tr><td>Max PDU Receiving High Prio</td><td colspan="2">≤</td><td colspan="2" rowspan="2"></td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Max PDU Receiving Low Prio</td><td colspan="2">≤</td></tr>
<tr><td>支持的 FMS 特征
.req [n] 0
.req [n] 1
.ind [m] 0
.ind [m] 1</td><td colspan="2" rowspan="4"></td><td colspan="2" rowspan="4"></td><td colspan="2" rowspan="4"></td></tr>
<tr><td colspan="2">X X</td></tr>
<tr><td colspan="2">- X</td></tr>
<tr><td colspan="2">X -
X X</td></tr>
<tr><td colspan="8">注释：
.req 0 : 作为客户机时不用此特性
.req 1 : 作为客户机时用此特性
.ind 0 : 作为服务器时不支持此特性
.ind 1 : 作为服务器时支持此特性
≤ : 本地值小于或等于远程值
≥ : 本地值大于或等于远程值
X : 兼容
- : 不兼容（出错情况）
[n] : 0 至 23
[m] : 24 至 47</td></tr>
</table>

图 43．本地 FMS 上下关系与远程 FMS 上下关系的兼容性

4.4.5.2 在 FMS 用户中的测试

- 口令测试

如果 FMS 用户支持用口令存取，则将检查此口令是否单义。这就是说，此口令应该与所有其他通信关系的口令不同。

- 如果口令不是单义的，而且 FMS 用户是启动服务的服务器，它应发布一个 Initiate.response 原语,该原语带有参数 Result（-）和出错代码“ Password Error ”。
- 如果口令不是单义的，而且 FMS 用户是启动服务的客户机，它将用 Abort 服务和用原因代码“ Password Error ”释放此连接。

- OD 版本测试

如果 FMS 用户对此连接有远程 OD，它可以检验接收到的参数“ Version OD ”与相关的远程 OD 的属性“ Version OD ”是否兼容。

- 如果它们不兼容，而且该 FMS 用户是启动服务的服务器，它可发布 Initiate.response 原语，该原语中带有参数 Result（-）和出错代码“ Version OD incompatible ”。
- 如果它们不兼容，而且该 FMS 用户是启动服务的客户机，它将用 Abort 服务和用原因

代码“Version OD incompatible”释放此连接。

- **行规号测试**

FMS 用户可以检验所接收的参数“Profile Number”是否与 VFD 的属性“Profile Number”兼容。

- 如果它们不兼容，而且该 FMS 用户是启动服务的服务器，它就可发布 Initiate.response 原语，该原语带有参数 Result（-）和出错代码“Profile Number incompatible”。
- 如果它们不兼容，而且该 FMS 用户是启动服务的客户机，它将用 Abort 服务和用原因代码“Profile Number incompatible”释放此连接。

4.4.6 面向连接的通信关系的状态机

4.4.6.1 状态机的描述

CONNECTION – NOT – ESTABLISHED（连接没有建立）

此连接没有建立。仅服务原语 Initiate.req，ASS.ind，Abort.req 和 ABT.ind 得到许可。用中止服务（Abort service）拒绝其他所有的服务。

CONNECTION – ESTABLISHING（CALLING）（连接正在建立（正在调用））

本地 FMS 用户希望建立连接。仅服务原语 ASS.con（+），ASS.con（-），Abort.req 和 ABT.ind 得到许可。用中止服务（Abort service）拒绝其他所有的服务。此状态缩写为 CON-ESTABLISHING-CALLING。

CONNECTION – ESTABLISHING（CALLED）（连接正在建立（被调用））

远程 FMS 用户希望建立连接。仅服务原语 Initiate.res（+），Initiate.res（-），Abort.req 和 ABT.ind 得到许可，用中止服务（Abort service）拒绝其他所有的服务。此状态缩写为 CON-ESTABLISHING-CALLED。

CONNECTION – ESTABLISHED（连接被建立）

建立了通信关系。不许可的服务原语 Initiate.req，Initiate.res（+），Initiate.res（-）和 DTU.ind，用中止服务（Abort service）拒绝执行这些服务。

为了复位通信引用（Reset CREF），将进行下列操作：

- 清存储器内容
- 设置 FMS CRL（动态部分）的属性“未完成服务的计数器客户机”和“未完成服务的计数器服务器”为 0。
- 设置通信关系状态为 CONNECTION – NOT – ESTABLISHED

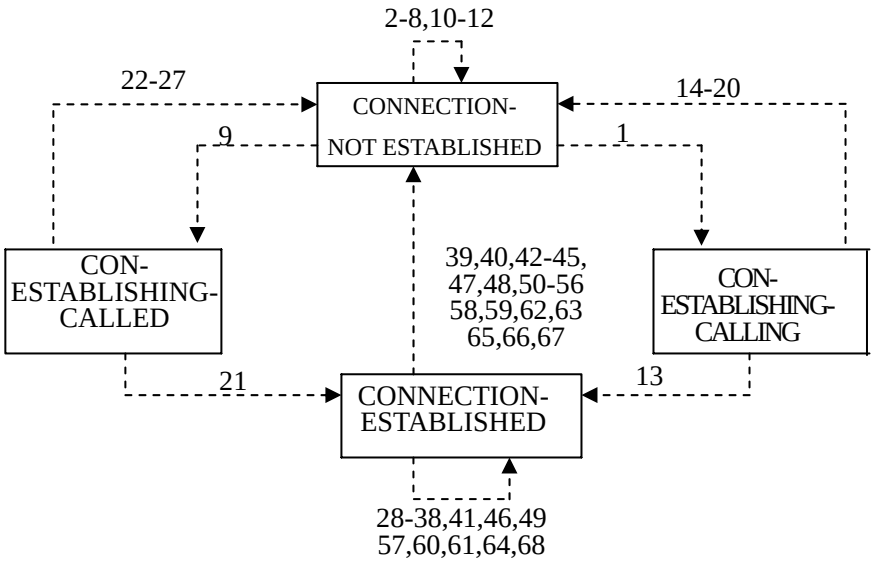


图 44.状态机

4.4.6.2 状态转换

在连接建立/释放过程中的状态转换：

当前状态 事件 \退出条件 => 采取的动作	转换	下一状态
CONNECTION-NOT-ESTABLISHED 从 FMS 用户接收到 Initiate.req \FMS CRL 登入项有效 => 将 INITIATE_REQ_PDU, (ASS.req) 发送到 LLI	**1**	**CON-ESTABLISHING-CALLING**
CONNECTION-NOT-ESTABLISHED 从 FMS 用户接收到 Initiate.req \FMS CRL 登入项无效 => 向 FMS 用户发送 Abort.ind <原因代码=ABT_RC1>	**2**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-NOT-ESTABLISHED 从 FMS 用户接收到 Abort.req =>忽略	**3**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-NOT-ESTABLISHED 从 FMS 用户接收到不允许、未知的或错误的服务原语 =>向 FMS 用户发送 Abort.ind <原因代码=ABT_RC2>	**4**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-NOT-ESTABLISHED 从 LLI 接收到 ABT.ind =>忽略	**5**	**CONNECTION-NOT-ESTABLISHED**

当前状态 事件 \退出条件 =>采取的动作	转换	下一状态
CONNECTION-NOT-ESTABLISHED 从 LLI 接收到错误的或不知道的服务原语 =>向 LLI 发送 ABT.req <原因代码=ABT_RC5>	**6**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-NOT-ESTABLISHED 从 LLI 接收到不允许的 LLI 服务原语 =>向 LLI 发送 ABT.req <原因代码=ABT_RC4>	**7**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-NOT-ESTABLISHED 接收到不允许,不知道或错误的 FMS PDU（ASS.ind） =>向 LLI 发送 ABT.req <原因代码=ABT_RC3>	**8**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-NOT-ESTABLISHED 从 LLI 接收到 INITIATE_REQ_PDU（ASS.ind） \FMS CRL 登入项有效 AND FMS 上下关系测试结果是肯定的 => 向 FMS 用户发送 Initiate.ind	**9**	**CON-ESTABLISHING-CALLED**
CONNECTION-NOT-ESTABLISHED 从 LLI 接收到 INITIATE_REQ_PDU（ASS.ind） \FMS CRL 登入项无效 => 向 LLI 发送 ABT.req< 原因代码=ABT_RC1 >	**10**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-NOT-ESTABLISHED 从 LLI 接收到 INITIATE_REQ_PDU（ASS.ind） \FMS CRL 登入项有效 AND 最大 PDU 长度测试结果是否定的 =>向 LLI 发送 INTIATE_ERROR_PDU（ASS.res（-）） < 出错代码=1 >	**11**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-NOT-ESTABLISHED 从 LLI 接收到 INITIATE_REQ_PDU（ASS.ind） \FMS CRL 登入项有效 AND 最大 PDU 长度测试结果是肯定的 AND 所支持特性测试的结果是否定的 =>向 LLI 发送 INTIATE_ERROR_PDU（ASS.res（-）） < 出错代码=2 >	**12**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLING 从 LLI 接收到 INITIATE_RES_PDU（ASS.con（+）） =>向 FMS 用户发送 Initiate.con	**13**	**CONNECTION-ESTABLISHED**
CON-ESTABLISHING-CALLING 从 LLI 接收 INITIATE_ERROR_PDU（ASS.con（-）） =>向 FMS 用户发送 Initiate.con（-） CREF 复位	**14**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLING 从 LLI 接收到 ABT.ind =>向 FMS 用户发送 Abort.ind,< 原因代码 ABT.ind > CREF 复位	**15**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLING 从 FMS 用户接收到 Abort.req =>向 LLI 发送 ABT.req< 原因代码由用户给出> CREF 复位	**16**	**CONNECTION-NOT-ESTABLISHED**

当前状态 事件 \退出条件 =>采取的动作	转换	下一状态
CON-ESTABLISHING-CALLING 从 LLI 接收到错误的或不知道的服务原语 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC5 > 向 FMS 用户发送 Abort.ind< 原因代码 = ABT_RC5 > CREF 复位	**17**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLING 从 LLI 接收到不允许的服务原语 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC4 > 向 FMS 用户发送 Abort.ind< 原因代码 = ABT_RC4 > CREF 复位	**18**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLING 从 LLI 接收到不允许的，错误的或不知道的 FMS PDU（ASS.con） =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC3 > 向 FMS 用户发送 Abort.ind< 原因代码 = ABT_RC3 > CREF 复位	**19**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLING 从 FMS 用户接收到不允许的，错误的或不知道的服务原语 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC2 > 向 FMS 用户发送 Abort.ind< 原因代码 = ABT_RC2 > CREF 复位	**20**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLED 从 FMS 用户接收到 Initiate.res（+） => 将 INITIATE_RES_PDU,（ASS.res（+））发送到 LLI	**21**	**CONNECTION-ESTABLISHED**
CON-ESTABLISHING-CALLED 从 FMS 用户接收到 Initiate.res（-） => 将 INITIATE_RES_PDU,（ASS.res（-））发送到 LLI CREF 复位	**22**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLED 从 FMS 用户接收到 Abort.req =>向 LLI 发送 ABT.req< 原因代码由用户给出> CREF 复位	**23**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLED 从 LLI 接收到 ABT.ind =>向 FMS 用户发送 Abort.ind,< 原因代码不在 ABT.ind 中> CREF 复位	**24**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLED 从 LLI 接收到错误的或不知道的服务原语 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC5 > 向 FMS 用户发送 Abort.ind< 原因代码 = ABT_RC5 > CREF 复位	**25**	**CONNECTION-NOT-ESTABLISHED**
CON-ESTABLISHING-CALLED 从 LLI 接收到不允许的 LLI 服务原语 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC4 > 向 FMS 用户发送 Abort.ind< 原因代码 = ABT_RC4 > CREF 复位	**26**	**CONNECTION-NOT-ESTABLISHED**

当前状态 事件 \退出条件 =>采取的动作	转换	下一状态
CON-ESTABLISHING-CALLED	**27**	**CONNECTION-ESTABLISHED**
从 FMS 用户接收到不允许的，错误的或不知道的服务原语 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC2 > 向 FMS 用户发送 Abort.ind< 原因代码 = ABT_RC2 > CREF 复位		
CONNECTION-ESTABLISHED	**28**	**CONNECTION-ESTABLISHED**
从 FMS 用户接收到需确认的 Service.req \OSCC < 最大未完成服务客户机 AND 不存在调用 ID AND PDU 长度≤以低优先级发送的最长 PDU AND 所支持的特性测试结果（客户机）是肯定的 => 向 LLI 发送带有 DTC.req 的需确认的 service_REQ_PDU OSCC := OSCC + 1		
CONNECTION-ESTABLISHED	**29**	**CONNECTION-ESTABLISHED**
从 FMS 用户接收到需确认的 Service.req \OSCC ≥ 最大未完成服务客户机 => 将 Reject.ind 发送到 FMS 用户< 拒绝代码 = 2 >		
CONNECTION-ESTABLISHED	**30**	**CONNECTION-ESTABLISHED**
从 FMS 用户接收到需确认的 Service.req \OSCC < 最大未完成服务客户机 AND 存在 Invoke ID => 将 Reject.ind 发送到 FMS 用户< 拒绝代码 = 1 >		
CONNECTION-ESTABLISHED	**31**	**CONNECTION-ESTABLISHED**
从 FMS 用户接收到需确认的 Service.req \OSCC < 最大未完成服务客户机 AND 不存在调用 ID AND PDU 长度 > 以低优先级发送的最长 PDU => 将 Reject.ind 发送到 FMS 用户< 拒绝代码 = 5 >		
CONNECTION-ESTABLISHED	**32**	**CONNECTION-ESTABLISHED**
从 FMS 用户接收到需确认的 Service.req \OSCC < 最大未完成服务客户机 AND 不存在调用 ID AND PDU 长度≤ 以低优先级发送的最长 PDU AND 所支持的特性测试结果（客户机）是否定的 => 将 Reject.ind 发送到 FMS 用户< 拒绝代码 = 3>		
CONNECTION-ESTABLISHED	**33**	**CONNECTION-ESTABLISHED**
从 FMS 用户接收到无需确认的 Service.req<priority = false> \PDU 长度≤以低优先级发送的最长 PDU AND 所支持的特性测试结果（客户机）是肯定的 => 将无需确认的 Service_REQ_PDU （DTA.req< priority = low >）发送到 LLI		
CONNECTION-ESTABLISHED	**34**	**CONNECTION-ESTABLISHED**
从 FMS 用户接收到无需确认的 Service.req<priority = false> \PDU 长度 > 以低优先级发送的最长 PDU => 将 Reject.ind 发送到 FMS 用户< 拒绝代码 = 5>		

当前状态 事件 \退出条件 =>采取的动作	转换	下一状态
CONNECTION-ESTABLISHED 从 FMS 用户接收到无需确认的 Service.req<priority = false> \PDU 长度≤以低优先级发送的最长 PDU AND 所支持的特性测试结果（客户机）是否定的 => 将 Reject.ind 发送到 FMS 用户< 拒绝代码 = 3>	**35**	**CONNECTION-ESTABLISHED**
CONNECTION-ESTABLISHED 从 FMS 用户接收到无需确认的 Service.req<priority = true> \PDU 长度≤以高优先级发送的最长 PDU AND 所支持的特性测试结果（客户机）是肯定的 => 将无需确认 Service_REQ_PDU （DTA.req <priority=high>） 发送到 LLI	**36**	**CONNECTION-ESTABLISHED**
CONNECTION-ESTABLISHED 从 FMS 用户接收到无需确认的 Service.req<priority = true> \PDU 长度 > 以高优先级发送的最长 PDU => 将 Reject.ind 发送到 FMS 用户< 拒绝代码 = 5>	**37**	**CONNECTION-ESTABLISHED**
CONNECTION-ESTABLISHED 从 FMS 用户接收到无需确认的 Service.req<priority = true> \PDU 长度 ≤ 以高优先级发送的最长 PDU AND 所支持的特性测试结果（客户机）是否定的 => 将 Reject.ind 发送到 FMS 用户< 拒绝代码 = 3>	**38**	**CONNECTION-ESTABLISHED**
CONNECTION-ESTABLISHED 从 FMS 用户接收到 Abort.req =>向 LLI 发送 ABT.req< 原因代码由用户给出> CREF 复位	**39**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 FMS 用户接收错误的,不知道的或不允许的服务原语 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC2 > 将 Abort.ind 发送给 FMS 用户< 原因代码 = ABT_RC2 > CREF 复位	**40**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到需确认的 Service_REQ_PDU（DTC.ind） \PDU 长度≤ 以低优先级接收的最长 PDU AND OSCS < 最大未完成的服务服务器 AND Invoke ID 不存在 AND 所支持的特性测试结果（服务器）是肯定的 => 向 FMS 用户发送需确认的 Service.ind OSCS:= OSCS + 1	**41**	**CONNECTION-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到需确认的 Service_REQ_PDU（DTC.ind） \PDU 长度 > 以低优先级接收的最长 PDU =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC6 > 将 Abort.ind 发送给 FMS 用户< 原因代码 = ABT_RC6 > CREF 复位	**42**	**CONNECTION-NOT-ESTABLISHED**

当前状态 事件 \退出条件 =>采取的动作	转换	下一状态
CONNECTION-ESTABLISHED 从 LLI 接收到需确认的 Service_REQ_PDU（DTC.ind） \PDU 长度≤以低优先级接收的最长 PDU AND OSCS ≥ 最大未完成的服务服务器 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC9 > 将 Abort.ind 发送给 FMS 用户< 原因代码 = ABT_RC9 > CREF 复位	**43**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到需确认的 Service_REQ_PDU（DTC.ind） \PDU 长度≤以低优先级接收的最长 PDU AND OSCS < 最大未完成的服务服务器 AND 调用 ID 已经存在 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC12 > 将 Abort.ind 发送给 FMS 用户< 原因代码 = ABT_RC12 > CREF 复位	**44**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到需确认的 Service_REQ_PDU（DTC.ind） \PDU 长度≤以低优先级接收的最长 PDU AND OSCS < 最大未完成的服务服务器 AND 调用 ID 不存在 AND 所支持的特性测试结果（服务器）是否定的 =>向 LLI 发送 ABT.req< 原因代码 = ABT_RC7 > 将 Abort.ind 发送给 FMS 用户< 原因代码 = ABT_RC7 > CREF 复位	**45**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到无需确认的 Service_REQ_PDU <priority = low>（DTA.ind） \PDU 长度≤ 以低优先级接收的最长 PDU AND 所支持的特性测试结果（服务器）是肯定的 => 将无需确认 Service.ind <priority = false> 发送给 FMS 用户	**46**	**CONNECTION-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到无需确认的 Service_REQ_PDU <priority = low> （DTA.ind） \PDU 长度 > 以低优先级接收的最长 PDU => 向 LLI 发送 ABT.req< 原因代码 = ABT_RC6 > 将 Abort.ind 发送给 FMS 用户< 原因代码 = ABT_RC6 > CREF 复位	**47**	**CONNECTION-NOT-ESTABLISHED**

当前状态 事件 \退出条件 =>采取的动作	转换	下一状态
CONNECTION-ESTABLISHED 从 LLI 接收到无需确认的 Service_REQ_PDU <priority = low>（DTA.ind） \PDU 的长度 ≤ 以低优先级接收的最长 PDU AND 所支持的特性测试结果（服务器）是否定的 => 向 LLI 发送 ABT.req <原因代码=ABT_RC7> Abort.ind 到 FMS 用户<原因代码=ABT_RC7> 复位 CREF	**48**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到无需确认的 Service_REQ_PDU <priority = high>（DTA.ind） \PDU 的长度≤ 以高优先级接收的最长 PDU AND 所支持的特性测试结果（服务器）是肯定的 => 向 FMS 用户发送无需确认 Service.ind<优先级=真>	**49**	**CONNECTION- ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到无需确认 Service_REQ_PDU （priority=high）（DTA.ind） \PDU 的长度 > 以高优先级接收的最长 PDU => ABT.req 到 LLI<原因代码=ABT_RC6> Abort.ind 到 FMS 用户<原因代码=ABT_RC6> 复位 CREF	**50**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到无需确认 Service_REQ_PDU （priority=high）（DTA.ind） \PDU 的长度≤ 以高优先级接收的最长 PDU AND 支持的特性测试结果是肯定的 => 向 LLI 发送 ABT.req <原因代码=ABT_RC7> 向 FMS 用户发送 Abort.ind <原因代码=ABT_RC7> 复位 CREF	**51**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到 ABT.ind => 向 FMS 用户发送 Abort.ind<ABT.ind 之外的原因代码> 复位 CREF	**52**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到 INITIATE_REQ_PDU（ASS.ind） 或 INITIATE_RES_PDU（ASS.con） => 向 LLI 发送 ABT.req<原因代码=ABT_RC10> 向 FMS 用户发送 Abort.ind<原因代码=ABT_RC10> 复位 CREF	**53**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到错误的或未知的服务原语 => 向 LLI 发送 ABT.req <原因代码=ABT_RC5> 向 FMS 用户发送 Abort.ind<原因代码=ABT_RC5> 复位 CREF	**54**	**CONNECTION-NOT-ESTABLISHED**

当前状态 事件 \退出条件 =>采取的动作	转换	下一状态
CONNECTION-ESTABLISHED 从 LLI 接收到不允许的 LLI 服务原语 => 向 LLI 发送 ABT.req<原因代码=ABT_RC4> 向 FMS 用户发送 Abort.ind<原因代码=ABT_RC4> 复位 CREF	**55**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收不允许的、未知的或错误的 FMS PDU => 向 LLI 发送 ABT.req <原因代码=ABT_RC3> 向 FMS 用户发送 Abort.ind<原因代码=ABT_RC3> 复位 CREF	**56**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 FMS 用户接收到需确认的 Service.res (+) \作为服务器存在调用 ID AND 来自.ind 的服务与来自.res 的服务是一致的 AND PDU 的长度 ≤ 低优先级发送的最长的 PDU => 向 LLI 发送需确认的带 DTC.res 的 Servic_RES_PDU OSCS:=OSCS-1	**57**	**CONNECTION-ESTABLISHED**
CONNECTION-ESTABLISHED 从 FMS 用户接收到需确认的 Service.res (-) \作为服务器存在调用 ID AND 来自.ind 的服务与来自.res 的服务是一致的 AND PDU 的长度 ≤ 低优先级发送的最长的 PDU => 向 LLI 发送需确认的带 DTC.res 的 Error_PDU OSCS:=OSCS-1	**68**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 FMS 用户接收到需确认的 Service.res \作为服务器不存在调用 ID => 向 LLI 发送 ABT.req <原因代码=ABT_RC8> 向 FMS 用户发送 Abort.ind<原因代码=ABT_RC8> 复位 CREF	**58**	**CONNECTION-NOT-ESTABLISHE**
CONNECTION-ESTABLISHED 从 FMS 用户接收到需确认的 Service.res \ 作为服务器存在调用 ID AND 来自.ind 的服务与来自.res 的服务是不一致的 => 向 LLI 发送 ABT.req <原因代码=ABT_RC11> 向 FMS 用户发送 Abort.ind <原因代码=ABT_RC11> 复位 CREF	**59**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 FMS 用户接收到需确认的 Service.res \作为服务器存在调用 ID AND 来自.ind 的服务与来自.res 的服务是一致的 AND PDU 的长度 >以低优先级发送的最长的 PDU => 向 LLI 发送带 DTC.res 的 REJECT_PDU<拒绝代码=5> OSCS:=OSCS-1	**60**	**CONNECTION-ESTABLISHED**

当前状态 事件 \退出条件 =>采取的动作	转换	下一状态
CONNECTION-ESTABLISHED 从 LLI 接收到需确认的 Service_RES_PDU 或需确认的 Error_PDU（DTC.con） \ PDU 的长度 ≤ 以低优先级接收的最长的 PDU AND 作为客户机存在调用 ID AND 来自.con 的服务与来自.req 的服务是一致的 => 向 FMS 用户发送需确认的服务 Service.con, OSCC:=OSCC-1	**61**	**CONNECTION-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到需确认服务 Service_RES_PDU 或需确认的 Error_PDU（DTC.con） \ PDU 的长度 > 以低优先级接收的最长的 PDU => 向 LLI 发送 ABT.req 到<原因代码=ABT_RC6> 向 FMS 用户发送 Abort.ind<原因码=ABT_RC6> 复位 CREF	**62**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到需确认的 Service_RES_PDU 或需确认的 Error_PDU（DTC.con） \ PDU 的长度 ≤ 以低优先级接收的最长的 PDU AND 作为客户机不存在调用 ID => 向 LLI 发送 ABT.req <原因代码=ABT_RC8> 向 FMS 用户发送 Abort.ind<原因代码=ABT_RC8> 复位 CREF	**63**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到需确认的 Service_RES_PDU 或需确认的 Error_PDU（DTC.con） \ PDU 的长度 ≤ 以低优先级接收的最长的 PDU AND 作为客户机存在调用 ID AND 来自.con 的服务与来自.req 的服务是不一致的 => 向 LLI 发送 ABT.req <原因代码=ABT_RC11> 向 FMS 用户发送 Abort.ind<原因代码=ABT_RC11> 复位 CREF	**67**	**CONNECTION-NOT-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到 REJECT_PDU（DTC.con） \作为客户机存在初始的调用 ID AND（<拒绝代码=5>或<拒绝代码=0>） => 向 FMS 用户发送的 Reject.ind<拒绝代码=5> OSCC:=OSCC-1	**64**	**CONNECTION-ESTABLISHED**
CONNECTION-ESTABLISHED 从 LLI 接收到 REJECT_PDU（DTC.con） \作为客户机不存在初始的调用 ID => 向 LLI 发送 ABT.req <原因代码=ABT_RC8> 向 FMS 用户发送的 Abort.ind<原因代码=ABT_RC8> 复位 CREF	**65**	**CONNECTION-NOT-ESTABLISHED**

当前状态 事件 \退出条件 =>采取的动作	转换	下一状态
CONNECTION-ESTABLISHED 从 LLI 接收到 REJECT_PDU（DTC.con） \作为客户机存在初始的调用 ID AND（<拒绝码代≠5>或<拒绝代码≠0>） => 向 LLI 发送 ABT.req <原因代码=ABT_RC3> 向 FMS 用户发送 Abort.ind<原因代码=ABT_RC3> 复位 CREF	**66**	**CONNECTION-NOT-ESTABLISHED**

4．4．7 无连接通信关系的状态机

4．4．7．1 客户机方的状态机

状态机描述

CONNECTIONLESS-CLIENT（无连接客户机）

为操作准备好通信关系。

只允许无需确认的服务请求（Service.req）原语，其他的服务应被拒绝或忽略。

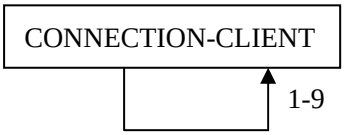


图 45．状态机

4.7.7.1.1 状态转换

当前状态 事件 \退出条件 => 采取的动作	转换	下一状态
CONNECTIONLESS-CLIENT 从 FMS 用户接收到无需确认的 Service.req<priority=false> \PDU 长度≤≤ 以低优先级发送的最大的 PDU 长度 AND 所支持的特性测试（客户机）是肯定的 =>向 LLI 发送无需确认的 Service_REQ_PDU (DTU.req< priority=low >)	1	CONNECTIONLESS-CLIENT
CONNECTIONLESS-CLIENT 从 FMS 用户接收到无需确认的 Service.req<priority=false> \PDU 长度 > 以低优先级发送的最长的 PDU => 向 FMS 用户发送 Reject.ind（拒绝代码=5）	2	CONNECTIONLESS-CLIENT
CONNECTIONLESS-CLIENT 从 FMS 用户接收到无需确认的 Service.req<priority=false> \PDU 长度≤以低优先级发送的最长的 PDU AND 所支持的特性测试（客户机）是否定的 => 向 FMS 用户发送 Reject.ind（拒绝代码=4）	3	CONNECTIONLESS-CLIENT
CONNECTIONLESS-CLIENT 从 FMS 用户接收到无需确认 Service.req<priority=true> \PDU 长度≤ 以低优先级发送的最长的 PDU AND 所支持的特性测试（客户机）是肯定的 => 向 LLI 发送无需确认的 Service_REQ_PDU（DTU.req< priority=true >）	4	CONNECTIONLESS-CLIENT
CONNECTIONLESS-CLIENT 从 FMS 用户接收到无需确认的 Service.req<priority=true> \PDU 长度 > 以高优先级发送的最大的 PDU => 向 FMS 用户发送 Reject.ind（拒绝代码=5）	5	CONNECTIONLESS-CLIENT
CONNECTIONLESS-CLIENT 从 FMS 用户接收到无需确认的 Service.req< priority=true > \PDU 长度≤ 以高优先级发送的最长的 PDU AND 所支持的特性测试（客户机）是否定的 => 向 FMS 用户发送 Reject.ind（拒绝代码=4）	6	CONNECTIONLESS-CLIENT
CONNECTIONLESS-CLIENT 从 FMS 用户接收到需确认的 Service.req => 向 FMS 用户发送的 Reject.ind（拒绝代码=4）	7	CONNECTIONLESS-CLIENT
CONNECTIONLESS-CLIENT 从 FMS 用户接收到不允许的、未知的或错误的服务原语 => 向 FMS 用户发送 Reject.ind（拒绝代码=6）	8	CONNECTIONLESS-CLIENT
CONNECTIONLESS-CLIENT 接收到 LLI 服务原语 => 忽略	9	CONNECTIONLESS-CLIENT

4.4.7.2 服务器方的状态机

4.4.7.2.1 状态机描述

CONNECTIONLESS-SERVER（无连接服务器）

为操作准备好通信关系。

只允许从 LLI 来的 DTU.ind，其他的服务应被拒绝或忽略。

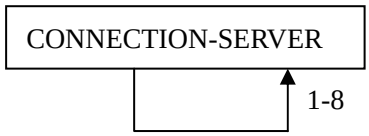


图 46．状态机

状态转换

当前状态 事件 \退出条件 => 采取的动作	转换	下一状态
CONNECTIONLESS-SERVER 从 LLI 接收到无需确认的 Service_REQ_PDU （DTU.ind<priority=low>） \PDU 长度≤ 以低优先级接收的最长的 PDU AND 所支持的特性测试（客户机）是肯定的 => 向 FMS 用户发送无确认的 Service.ind priority=false >	1	CONNECTIONLESS-SERVER
CONNECTIONLESS-SERVER 从 LLI 接收到无需确认的 Service_REQ_PDU （DTU.ind< priority=low > ） \PDU 长度 > 以低优先级接收的最长的 PDU => 忽略	2	CONNECTIONLESS-SERVER
CONNECTIONLESS-SERVER 从 LLI 接收到无需确认的 Service_REQ_PDU （DTU.ind< priority=low >） \PDU 长度 ≤ 以低优先级接收的最长的 PDU AND 所支持的特性测试（客户机）是否定的 => 忽略	3	CONNECTIONLESS-SERVER
CONNECTIONLESS-SERVER 从 LLI 接收到无需确认的 Service_REQ_PDU （DTU.ind< priority=high >） \PDU 长度 ≤ 以高优先级接收的最长的 PDU AND 所支持的特性测试（客户机）是肯定的 => 向 FMS 用户发送无需确认的 Service.ind <priority=true>	4	CONNECTIONLESS-SERVER
CONNECTIONLESS-SERVER 从 LLI 接收到无需确认的 Service_REQ_PDU （DTU.ind< priority=high >） \PDU 长度 > 以高优先级接收的最长的 PDU => 忽略	5	CONNECTIONLESS-SERVER

当前状态 事件 \退出条件 =>采取的动作	转换	下一动作
CONNECTIONLESS-SERVER 从 LLI 接收到无需确认的 Service_REQ_PDU （DTU.ind< priority=high >） \PDU 长度 ≤ 以高优先级接收的最长的 PDU AND 所支持的特性测试（客户机）是否定的 => 忽略	**6**	**CONNECTIONLESS-SERVER**
CONNECTIONLESS-SERVER 不允许的，未知的或错误的 LLI 服务原语 OR 接收到不允许的，未知的或错误的 FMS PDU => 忽略	**7**	**CONNECTIONLESS-SERVER**
CONNECTIONLESS-SERVER 从 FMS 用户接收服务原语 => 向 FMS 用户发送 Reject.ind<拒绝代码=6>	**8**	**CONNECTIONLESS-SERVER**

4.5 域管理

4.5.1 模型描述

域是存储器的一部分，它可以包含程序或数据。域的数据类型是“八位位组串”(octet string)。在一个域中，八位位组串的最大数在对象描述中定义。无论何时，在域中只能进行一个上载服务或一个下载服务，不能两者同时进行。

下列服务可以对域对象进行操作：

```
                                        +-------------------------------+
- OD Service    ----------------------> !  对象描述                     !
- InitiateDownloadSequence      \       !  域                           !
- DownloadSegment               :       !                               !
- TerminateDownloadSequence     :       !  +----------------------+    !
- InitiateUploadSequence        >-------> !          域           !    !
- UploadSegment                 :       !  +------------- ---------+    !
- TerminateUploadSequence       :       +-------------------------------+
- RequestDomainDownload         :
- RequestDomainUpload           /
```

图 47. 域服务

4.5.2 域对象

4.5.2.1 属性

对象：域

关键属性：索引（Index）
关键属性：域名（Domain Name）
属性：最大八位位组数（Max Octets）
属性：口令（Password）
属性：存取组（Access Groups）
属性：存取权（Access Rights）
属性：本地地址（Local Address）
属性：域状态（Domain State）
属性：上载状态（Upload State）
属性：计数器（Counter）
属性：扩展（Extension）

索引

在 OD 中对象的逻辑地址

域名

该属性规定域的符号名，域名的存在和长度在 OD 对象描述中定义。

最大八位位组数

该属性规定域的八位位组串的最大数

口令

该属性为存取权规定口令。

存取组

该属性规定在特定的存取组中的对象成员。如果相关的位被设置，则对象是存取组的成员。

表 17．域的存取组

位	含义
7	存取组 1
6	存取组 2
5	存取组 3
4	存取组 4
3	存取组 5
2	存取组 6
1	存取组 7
0	存取组 8

存取权

该属性规定了存取对象的权。如果相关位被设置，则对应的存取权被许可。

表 10．域的存取权限

Bit	名字	含义
7	R	读已注册口令的权
6	W	写已注册口令的权
5	U	在一个 PI 中使用已注册口令的权
3	Rg	读存取组的权
2	Wg	写存取组的权
1	Ug	在一个 PI 中使用存取组的权
15	Ra	读所有通信伙伴的权
14	Wa	写所有通信伙伴的权
13	Ua	在一个 PI 中使用所有通信伙伴的权

本地地址

该属性是系统对实际对象的特有的引用。可利用该属性在内部对对象编址。如果没有提供此类引用，则本地地址属性的值为十六进制数 FFFFFFFF。

域状态

该属性规定了**域对象**的状态。

1<=> EXISTENT（存在）　2<=> LOADING（装载）　3<=> INCOMPLETE（未完成）

4<=> COMPLETE（完成）`　5<=> READY（准备好）　6<=> IN-USE（在使用中）

上载状态

该属性规定在域上载时的状态机的状态。

0<=> NON-EXISTENT（不存在）

1<=> UPLOADING（正在上载）

2<=> UPLOADED（上载完成）

计数器

该属性规定当前使用此域的程序调用数。程序调用管理管理该计数器，而域管理只能解释该计数器。如果计数器属性值大于 0，则该域正在使用中，而且不能用 Download 服务重写该计数器。

扩展

该属性包含行规所规定的信息。

4.5.2.2 在传输中 OD 的对象描述

索引	对象代码	最大八位位组数	口令	存取组	存取权
123	Domain	513	12		R W U

本地地址	域状态	上载状态	计数器	域名	扩展
B49A hex				Volz	

图 48. S-OD 中一个登入项的结构

对象代码(Object code)

域对象的标识。除了上述对象属性外，在 GetOD 服务和 PutOD 服务中也应该传输此对象标识。

4.5.3 下载服务

使用下载（Download）服务可以把数据从客户机装载到服务器的域中。

4.5.3.1 启动下载序列（InitiateDownloadSequence）

启动下载序列（InitiateDownloadSequence）服务用于开始对域的装载，域的索引或域名包含在服务请求中。

表 19.启动下载序列

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
域名	S	
结果(+)		S
结果(-)		S
出错类型		M

变元

变元提交所请求服务的专用参数

存取说明

该参数规定是用索引还是用域名对对象寻址。

索引

该参数规定域的逻辑地址。

域名

该参数规定域名。

结果（+）

结果（+）参数表示所请求的服务执行成功。

结果（-）

结果（-）参数表示服务执行没有成功。

出错类型

该参数提供了失败原因。

4.5.3.2 下载数据段（DownloadSegment）

下载数据段（DownloadSegment）服务用于向服务器的域传送一个数据段。在服务响应中传送该数据段。

表 20.下载数据段

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
域名	S	
结果（+）		S
装载数据（Load Data）		M
更多的数据（More Follows）		M
结果（-）		S
出错类型		M

变元

变元提交所请求服务的专用参数

存取说明

该参数规定是用索引还是用域名对对象寻址。

索引

该参数规定域的逻辑地址。

域名

该参数规定域名。

结果（+）

此参数表示所请求的服务执行成功。

装载数据

该参数包含要被下载的数据。

更多的数据

该参数指示是否还有更多的数据要发送。

结果（-）

此参数表示服务执行没有成功。

出错类型

该参数提供失败的原因。

4.5.3.3 终止下载序列（TerminateDownloadSequence）

终止下载序列（TerminateDownloadSequence）服务用于结束对域的装载，该域的索引或域名包含在服务请求中。

表 21．终止下载序列

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
域名	S	
最后结果（Final Result）	M	
结果（+）		S
结果（-）		S
出错类型		M

变元

变元提交所请求服务的专用参数

存取说明

该参数规定是用索引还是用域名对对象寻址。

索引

该参数规定域的逻辑地址。

域名

该参数规定域名。

最后结果

该参数通知客户机，服务器是否成功地结束了装载任务。

结果（+）

结果（+）参数表示请求的服务执行成功。

结果（-）

结果（-）参数表示服务执行没有成功。

出错类型

该参数提供了失败原因。

4.5.3.4 请求域下载（RequestDomainDownload）

服务器利用请求域下载（RequestDomainDownload） 服务向客户机请求执行对域的下载任务，该域的索引或域名包含在服务请求中。在下载序列没有完全结束之前，不发送带有结果（+）的服务响应。

表 22．请求域下载

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
域名	S	
附加信息	U	
结果（+）		S
结果（-）		S
出错类型		M

变元

变元提交所请求服务的专用参数

存取说明

该参数规定是用索引还是用域名对对象寻址。

索引

该参数规定域的逻辑地址。

域名

该参数规定域名。

附加信息（AdditionalInformation）

该参数包含有关域名的附加信息。例如该信息可以包含一个文件名。域名和文件名之间的相互关系是一个本地事件，由应用作具体规定。

结果（+）

结果（+）参数表示请求的服务执行成功。

结果（-）

结果（-）参数表示服务执行没有成功。

出错类型

该参数提供了失败原因。

4.5.4 上载服务

上载（Upload）服务用于从服务器的域向客户机传送数据。

4.5.4.1 启动上载序列（InitiateUploadSequence）

启动上载序列（InitiateUploadSequence）服务用于开始对域的上载，该域的索引或域名包含在此服务请求中。

表 23．启动上载序列

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
域名	S	
结果（+）		S
结果（-）		S
出错类型		M

变元

变元提交所请求服务的专用参数

存取说明

该参数规定是用索引还是用域名对对象寻址。

索引

该参数规定域的逻辑地址。

域名

该参数规定域名。

结果（+）

结果（+）参数表示请求的服务执行成功。

结果（-）

结果（-）参数表示服务执行没有成功。

出错类型

该参数提供失败原因。

4.5.4.2 上载数据段（UploadSegment）(上载数据段)

上载数据段（UploadSegment）服务用于向客户机传送服务器域中的一个数据段。

表 24.上载数据段

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
域名	S	
结果（+）		S
上载数据		M
更多的数据		M
结果（-）		S
出错类型		M

变元

变元提交所请求服务的专用参数

存取说明

该参数规定是用索引还是用域名对对象寻址。

索引

该参数规定域的逻辑地址。

域名

该参数规定域名。

结果（+）

结果（+）参数表示请求的服务执行成功。

装载数据

该参数包含要上载的数据。

更多的数据

该参数指示是否还有更多的数据要传送。

结果（-）

结果（-）参数表示服务执行失败。

出错类型

该参数提供失败原因。

4.5.4.3 终止上载序列（TerminateUploadSequence）

终止上载序列（TerminateUploadSequence）服务用于结束上载序列。

表 25. 终止上载序列

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
域名	S	
结果（+）		S
结果（-）		S
出错类型		M

变元

变元提交所请求服务的专用参数

存取说明

该参数规定是用索引还是用域名对对象寻址。

索引

该参数指定域的逻辑地址。

域名

该参数指定域的名称。

结果（+）

结果（+）参数表示请求的服务执行成功。

结果（-）

结果（-）参数表示服务执行失败。

出错类型

该参数提供失败原因。

4.5.4.4 请求域上载（RequestDomainUpload）

服务器使用请求域上载（RequestDomainUpload） 服务请求把指定域的内容传送给客户机。直到完全完成上载序列后，才发送带有结果（+）的服务响应。

表 26．请求域上载

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
域名	S	
附加信息	U	
结果（+）		S
结果（-）		S
出错类型		M

变元

变元提交所请求服务的专用参数

存取说明

该参数规定是用索引还是用域名对对象寻址。

索引

该参数指定域的逻辑地址。

域名

该参数指定域的名称。

附加信息

该参数应包括关于域名的附加信息，例如它可以包括一个文件名。域名和文件名之间的相互关系是一个本地事件，由应用作具体规定。

结果（+）

结果（+）参数表示请求的服务执行成功。

结果（-）

结果（-）参数表示服务执行失败。

出错类型

该参数提供失败原因。

4.5.5 下载的状态机

4.5.5.1 状态机描述

EXISTENT(存在)

域存在，但未定义内容。

LOADING（在装载）

域的装载正在进行。

INCOMPLETE（未完成）

装载失败，内容不完整。

COMPLETE（完成）

已完成域内容的传送，但域还没有释放。

READY（准备好）

域已释放，可以使用。

IN-USE（在使用）

域被一个程序调用所占用，在该状态下不允许下载。

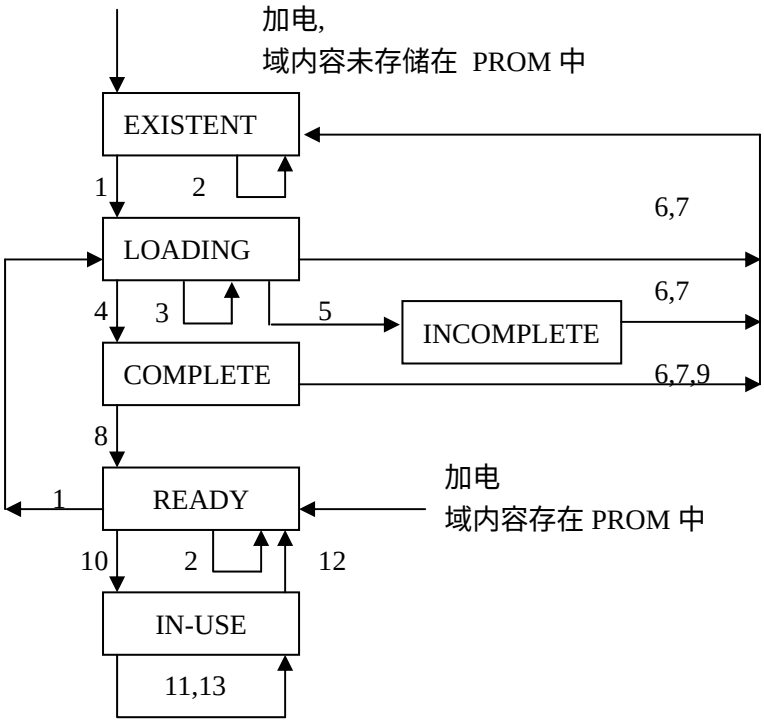


图 49．域下载状态机（服务器）

不能在同一时间内对同一个域进行 Download 和 Upload 操作。

4.5.5.2 状态转换

事件

\ 退出条件

=> 采取的动作

---

1 InitiateDownloadSequence.ind

\Upload 状态机处在 NON-EXISTENT 状态

=> .res（+）

2 InitiateDownloadSequence.ind

\Upload 状态机不处于 NON-EXISTENT 状态

=> .res（-） 对象约束冲突

3 DownloadSegment.con（+）
\更多的数据 = true
4 DownloadSegment.con（+）
\更多的数据 = false
5 DownloadSegment.con（-）
6 TerminateDownloadSequence.req
\最后结果 = false
\.con（+/-）
7 Abort
8 TerminateDownloadSequence.req
\最后结果 = true
\.con（+）
9 TerminateDownloadSequence.req
\最后结果 = true
\.con（-）
参见程序调用的状态转换图
10 Counter increment（Start/Resume）（计数器增加（开始/恢复））
=>counter :=1
11 Counter increment（Start/Resume）（计数器增加（开始/恢复））
=>counter :=counter+1
12 Counter decrement（Stop/Kill/End of Program/Program Stop）（计数器减少（停止/削除/程序结束/程序停止））
\counter=1
=>counter:=0
13 Counter decrement（Stop/Kill/End of Program/Program Stop）（计数器减少（停止/削除/程序结束/程序停止））
\counter>1
=>counter:=counter-1

4.5.6 上载的状态机

4.5.6.1 状态机描述

NON-EXISTENT（不存在）

域存在，但没有运行 Upload

UPLOADING（在上载）

域正在上载，不允许下载

UPLOADED（已上载）

域的内容已全部被传送，但终止上载序列（TerminateUploadSequence）服务仍在执行中。

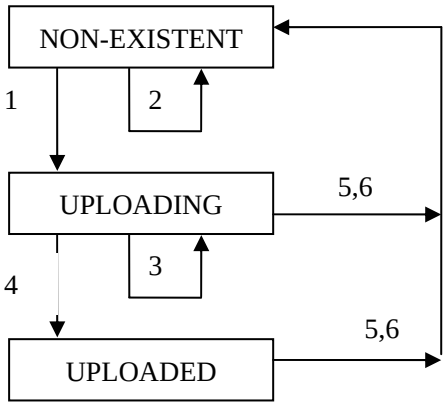


图 50. 域上载状态机（服务器）

不能在同一时间内对同一个域进行 Download 和 Upload 操作。

4.5.6.2 状态转换

事件

\退出条件

=>采取的动作

---

1 InitiateUploadSequence.ind

\Download 状态机处在 EXISTENT 或 READY 或 IN-USE 状态

=> .res (+)

2 InitiateUploadSequence.ind

\Download 状态机处在 LOADING 或 INCOMPLETE 或 COMPLETE 状态

=> .res (-) 对象约束冲突

3 UploadSegment.ind

\更多的数据 = true

=> .res (+)

4 UploadSegment.ind

\更多的数据 = false

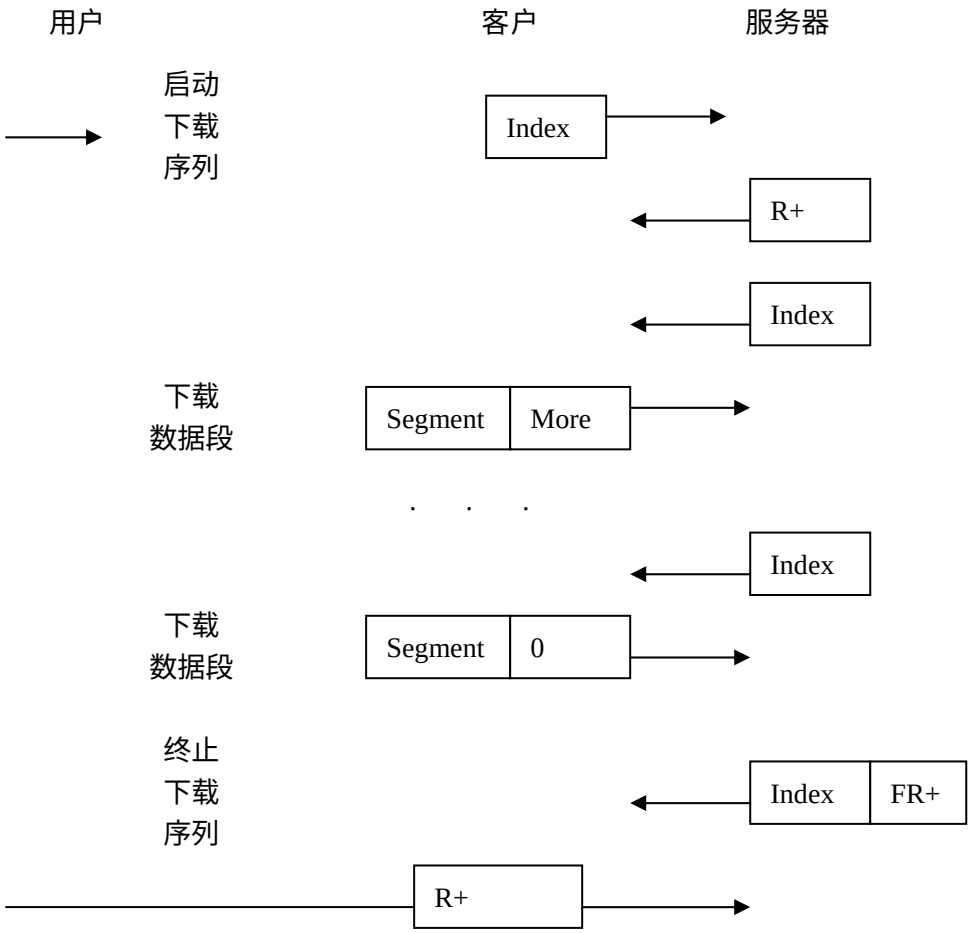
=> .res (+)

5 TerminateUploadSequence.res (+/-)

6 Abort.ind

4.5.7 举例

4.5.7.1 下载 时序举例



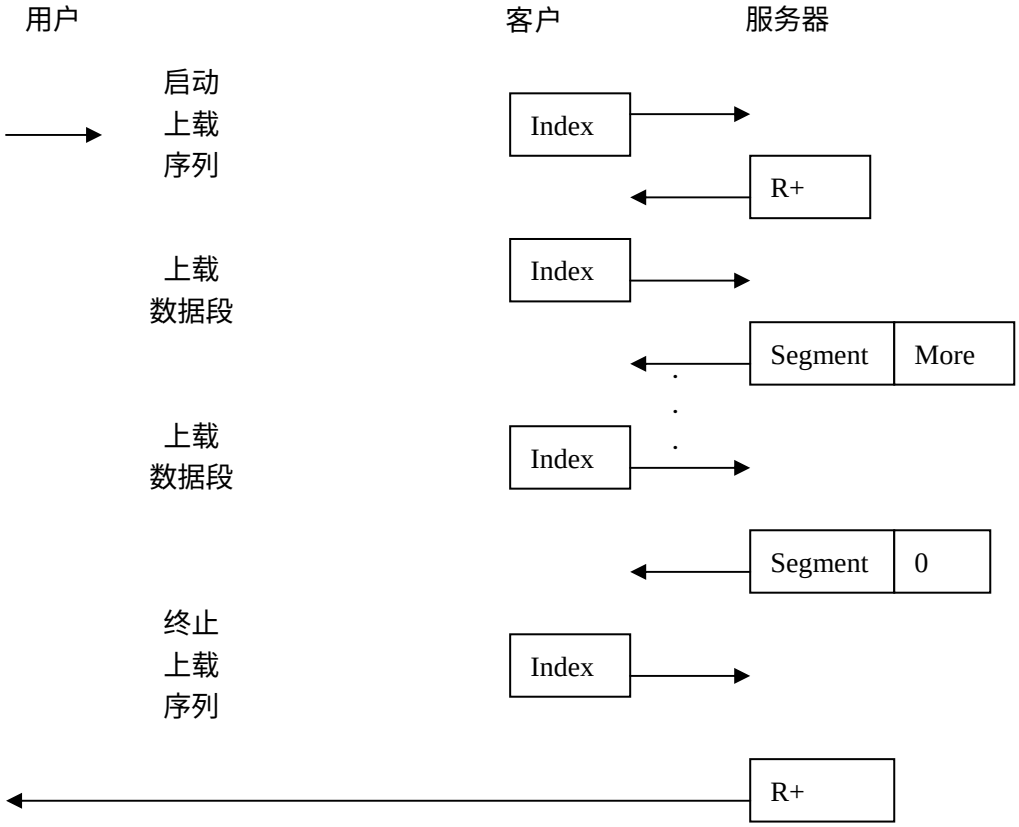
R+: 肯定的应答

More: 标识，表示还有更多的数据段要传送

FR+：发送最终结果

图 51. 下载序列的举例

4.5.7.2 上载序列举例



R+: 肯定的应答

More: 标识，表示还有更多的数据段要传送

图 52. 上载序列的举例

4.6 程序调用管理

对程序调用对象的服务：

- OD 服务
- CreateProgramInvocation
- DeleteProgramInvocation

- Start
- Stop
- Resume
- Reset
- Kill

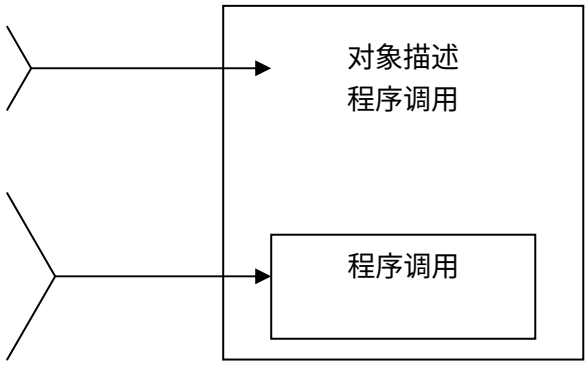


图 53. 程序调用服务

4.6.1 模型描述

程序调用模型提供的服务有：将域连接到一个程序、开始该程序、停止该程序以及删除

该程序。在一个设备中可以建立多个程序调用。通过 DP-OD 中的登入项定义程序调用。

4．6．2 程序调用（ PI）对象

程序调用是一个对象，在其中具有代码和数据的域被组合成一个可执行程序。程序调用可以预定义或在线创建。当重新加载对象字典（OD）时，所有的程序调用对象就被删除。

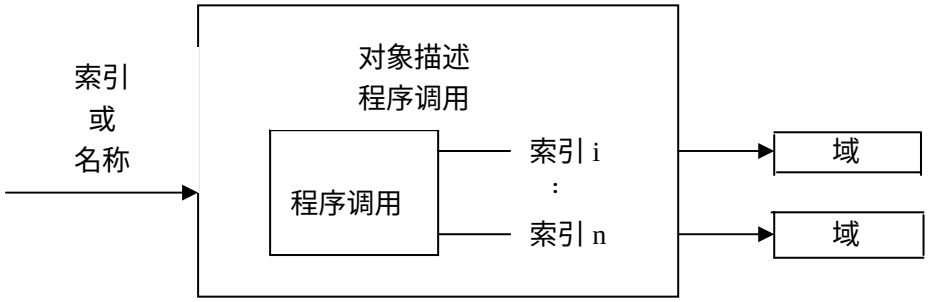


图 54．程序调用概要

4.6.2.1 属性

对象：程序调用（Program Invocation）

关键属性：索引（Index）
关键属性：程序调用名称（PI Name）
属性：域的数目（Number of Domains）
属性：口令（Password）
属性：存取组（Access Groups）
属性：存取权（Access Rights）
属性：可删除的（Deletable）
属性：可再用的（Reusable）
属性：程序调用状态（PI State）
属性：域索引表（List of Domain Index）
属性：索引（Index）
属性：扩展（Extension）

索引

OD 中登入项的逻辑地址

PI 名

对象名。在 OD 对象描述中定义该名称的存在和长度。

域的数目

进入该程序调用的域的数目。该数受创建程序调用服务的最大 PDU 长度限制。

口令

该属性包含存取权限的口令。

存取组

该属性将对象与特定的存取组相关联。如果相关的位被置位，则此对象属于相关存取组。

表 27．程序调用的存取组

位	含义
7	存取组 1
6	存取组 2
5	存取组 3
4	存取组 4
3	存取组 5
2	存取组 6
1	存取组 7
0	存取组 8

存取权

该属性包含有关存取权限的信息。如果相关的位置位，则特定的存取权限就存在。

表 28．PI 的存取权

位	名字	含义
7	S	为已经注册的口令（起动，恢复，复位）起动 PI 的权
6	H	为已经注册的口令（停止）停止 PI 的权
5	D	为已经注册的口令（削除，删除 PI）删除 PI 的权
3	Sg	开始存取组的权
2	Hg	停止存取组的权
1	Dg	删除存取组的权
15	Sa	开始所有的通信伙伴的权
14	Ha	停止所有的通信伙伴的权
13	Da	删除所有的通信伙伴的权

可删除的（Deletable）

该属性表明是否可以使用删除程序调用服务来删除此程序调用对象。

True <=> deletable（可删除的）

False <=> not deletable（不可删除的）

可再用的（Reusable）

该属性表示程序调用在执行后，是否转变到‘空闲’（IDLE）状态或‘不可运行’（UNRUNNABLE）状态。

True <=> 转换到 IDLE 状态

False <=> 转换到 UNRUNNABLE 状态

PI 状态（PI State）

该属性表示该程序的状态。

1 <=> UNRUNNABLE（不可运行）

2 <=> IDLE（空闲）

3 <=> RUNNING（在运行）

4 <=> STOPPED（已停止）
5 <=> STARTING（在起动）
6 <=> STOPPING（在停止）
7 <=> RESUMING（在恢复）
8 <=> RESETING（在复位）

域索引表（List of Domain Index）

该属性包含了多个域的索引，这些域被组合成此程序调用。在程序调用中，域索引的登入项的次序与在创建程序调用服务中域的次序相对应。索引表中的第一个域应包含有一个可执行程序。

扩展（Extension）

该属性包含行规的特定信息。

4.6.2.2 在传输中 OD 的对象描述

索引	对象代码	域的数量	口令	存取组	存取权	可删除的	可再用的
130	PI	7	17		S H D	true	True

PI 状态	域索引（1）	域索引（2）	域索引（3）	域索引（4）	域索引（5）	域索引（6）
	34	55	56	66	67	68

域索引（7）	PI 名	扩展
77	P. Thiessmeier	

图 55. DP-OD 中程序调用的对象描述（举例）

对象代码（Object Code）

程序调用的标签，在 GetOD 服务和 PutOD 服务中，该标签与对象属性一起发送。

4.6.3 程序调用服务

4.6.3.1 建立程序调用（CreateProgramInvocation）

通过该服务，把已在 OD 中定义了的域（例如程序、数据区）组合成一个程序，并在线地定义在 DP-OD 中。使用逻辑地址或名称对该程序寻址。一个用 CreateProgramInvocation 服务建立的程序调用，它的 Deletable 属性设置为 true。对象描述中的 Number of Domains 属性由服务器自动建立。域表的第一个域应包含一个可执行的程序。

只有对这些域有使用权限的，才能建立程序调用，否则响应 Result（-）。如果已经存在与要建立的程序调用有相同存取权限的同名程序调用，则不再建立新的对象。在这种情况下，将已存在对象的逻辑地址（索引）传送给客户机。

表 29．建立程序调用

参数名	.req .ind	.res .con
变元（Argument）	M	
口令（Password）	M	
存取组（Access Groups）	M	
存取权（Access Rights）	M	
可再用的（Reusable）	M	
域表（List of Domains）	M	
域索引（Domain Index）	S	
域名（Domain Name）	S	
PI 名（PI Name）	U	
扩展（Extension）	U	
结果（+）（Result（+））		S
索引（Index）		M
结果（-）（Result（-））		S
出错类型（Error Type）		M

变元

该参数包含服务请求的服务专用参数。如果不支持存取权限（Access Protection Supported = false），则参数口令、存取组和存取权限无效。

口令

该参数说明口令属性应设置的值。

存取组

该参数说明存取组属性应设置的值。

存取权

该参数说明存取权属性应设置的值。

可再用的

该参数说明可再用的属性应设置的值。

域表

该参数包含域的地址，这些域在程序调用中被组合成一个程序。每个地址可以是逻辑地址或名称。

域索引

域的索引

域名

域的名称

PI 名

该参数说明程序调用对象的名称属性应设置的值。

扩展

该参数包含扩展属性。

结果（+）

结果（+）参数表示请求的服务执行成功。

索引

该索引与建立的程序调用相关联。

结果（-）

结果（-）参数表示服务执行没有成功。

出错类型

该参数包含服务未曾成功完成的原因的信息。

4.6.3.2 删除程序调用（DeleteProgramInvocation）

使用删除程序调用（DeleteProgramInvocation）服务删除已在 DP-OD 中定义的程序调用。只有程序调用对象的可删除属性值为 true 时，才能删除程序调用。

表 30. 删除程序调用

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
PI 名	S	
结果（+）		S
结果（-）		S
出错类型		M

变元

变元包括服务请求的服务专用参数

存取说明

该参数指出是否使用索引或域名寻址。

索引

该参数是与服务相关的在 OD 中的程序调用的逻辑地址。

PI 名

该参数是程序调用名。

结果（+）

结果（+）参数表示请求服务执行成功。

结果（-）

结果（-）参数表示服务执行失败。

出错类型

该参数包含服务未曾成功完成的原因信息。

4.6.3.3 起动（Start）

通过 Start 服务起动一个程序。程序从头开始运行。所使用域的计数器加 1。在起动前，要对所有相关的域进行检查，判断它们是处在准备好（READY）状态还是在使用中（IN-USE）状态。

表 31．起动

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
PI 名	S	
执行变元	U	
结果（+）		S
结果（-）		S
出错类型		M
PI 状态		M

变元

变元包含服务请求的服务专用参数

存取说明

该参数指出是否用索引或域名寻址。

索引

该参数是与服务相关的在 OD 中的程序调用的逻辑地址。

PI 名

该参数是程序调用的名称。

执行变元（Execution Argument）

该可选的参数是八位位组串类型，它包括提供给程序调用的用于开始的数据。

结果（+）

结果（+）参数表示请求的服务执行成功。

结果（-）

结果（-）参数表示服务执行失败。

出错类型

该参数包括服务未曾成功完成的原因信息。

PI 状态（PI State）

该参数指示程序调用的状态。

4.6.3.4 停止（Stop）

将一个正在运行的程序停止，并不使其回到开始状态。所用域的计数器减 1。

表 32. 停止

参数名	·req ·ind	·res ·con
变元 存取说明 索引 PI 名	M M S M	
结果（+）		S
结果（-） 出错类型 PI 状态		S M M

变元

变元包含服务请求的服务专用参数

存取说明

此参数表述是否有索引或名被用于寻址

索引

此参数是在与此服务有关的 OD 内程序调用的逻辑地址

PI 名

此参数是程序调用名

结果（+）

结果（+）参数将指示服务已成功完成

结果（-）

结果（-）将指示服务请求未曾成功完成

出错类型

此参数包含此服务未曾成功完成的原因信息

PI 状态

此参数指示程序调用的状态

4.6.3.5 恢复（Resume）

将一个已经停止运行的程序设置为 RUNNING（运行）状态，并非复位。所用域的计数器加 1。在此前检查了相应的域是否处于准备好（READY）或处于在用（IN USE）状态。

表 33. 恢复

参数名	·req ·ind	·res ·con
变元 存取说明 索引 PI 名 执行变元	M M S S U	
结果（+）		S
结果（-） 出错类型 PI 状态		S M M

变元

变元包含服务请求的服务专用参数

存取说明

此参数表述是否有索引或名称被用于寻址

索引

此参数是在与此服务有关的 OD 内程序调用的逻辑地址

PI 名

此参数是程序调用名

执行变元

此八位位组串类型的可选参数包含为起动程序调用所给出的数据。

结果（+）

结果（+）参数将指示服务已成功完成

结果（-）

结果（-）将指示服务请求未曾成功完成

出错类型

此参数包含此服务未曾成功完成的原因信息

PI 状态

此参数指示程序调用的状态

4.6.3.6 复位（Reset）

将一个具有可再用(Reusable)属性为真且已停止运行的程序设置为开始状态。程序调用转为空闲（IDLE）状态。若可再用属性为假，程序调用对象转为不可运行（UNRUNNABLE）状态。

表 34. 复位

参数名	·req ·ind	·res ·con
变元 存取说明 索引 PI 名	M M S S	
结果（+）		S
结果（-） 出错类型 PI 状态		S M M

变元

变元包含服务请求的服务专用参数

存取说明

此参数表述是否有索引或名被用于寻址

索引

此参数是在与此服务有关的 OD 内程序调用的逻辑地址

PI 名

此参数是程序调用名

结果（+）

结果（+）参数将指示服务已成功完成

结果（-）

结果（-）将指示服务请求未曾成功完成

出错类型

此参数包含有关服务未曾成功完成的原因信息

PI 状态

此参数指示程序调用的状态

4.6.3.7 削除（Kill）

将一个程序调用设置为不可运行（UNRUNNABLE）状态，这与当前状态无关。若程序调用的状态为运行（RUNNING）或停止（STOPPING），所用域的计数器减 1。

表 35. 削除

参数名	·req ·ind	·res ·con
变元 存取说明 索引 PI 名	M M S S	
结果（+）		S
结果（-） 出错类型		S M

变元

变元包含服务请求的服务专用参数

存取说明

此参数表述是否有索引或名被用于寻址

索引

此参数是在与此服务有关的 OD 内程序调用的逻辑地址

PI 名

此参数是程序调用名

结果（+）

结果（+）参数将指示服务已成功完成

结果（-）

此参数将指示服务请求未曾成功完成

出错类型

此参数包含有关服务未曾成功完成的原因信息

4.6.4 状态机（State Machine）

4.6.4.1 状态机描述

NON-EXISTENT（不存在）

若无另外定义，在上电后或在删除后，程序调用处于不存在状态。

IDLE（空闲）

通过让一个程序调用进入 DP-OD 内，使之转为空闲状态。在一次成功的服务复位后，以及在程序结束之后，程序调用也转为空闲状态。可预先定义程序调用，这类程序调用在上电后转为空闲状态。

STARTING（起动中）

程序准备起动的中间状态

RUNNING（运行中）

在此状态程序正在运行。在此程序调用中被定义的域，现在处于在用（INUSE）状态。

STOPPING（停止中）

程序准备停止的中间状态

STOPPED（已停止）

在此状态程序被停止

RESUMING（恢复中）

程序准备恢复的中间状态

RESETTING（复位中）

程序被设置为一种已充分定义的初始状态的中间状态

UNRUNNABLE（不可运行）

在此状态下程序被停止，且不可能再起动，它仅可能被删除。

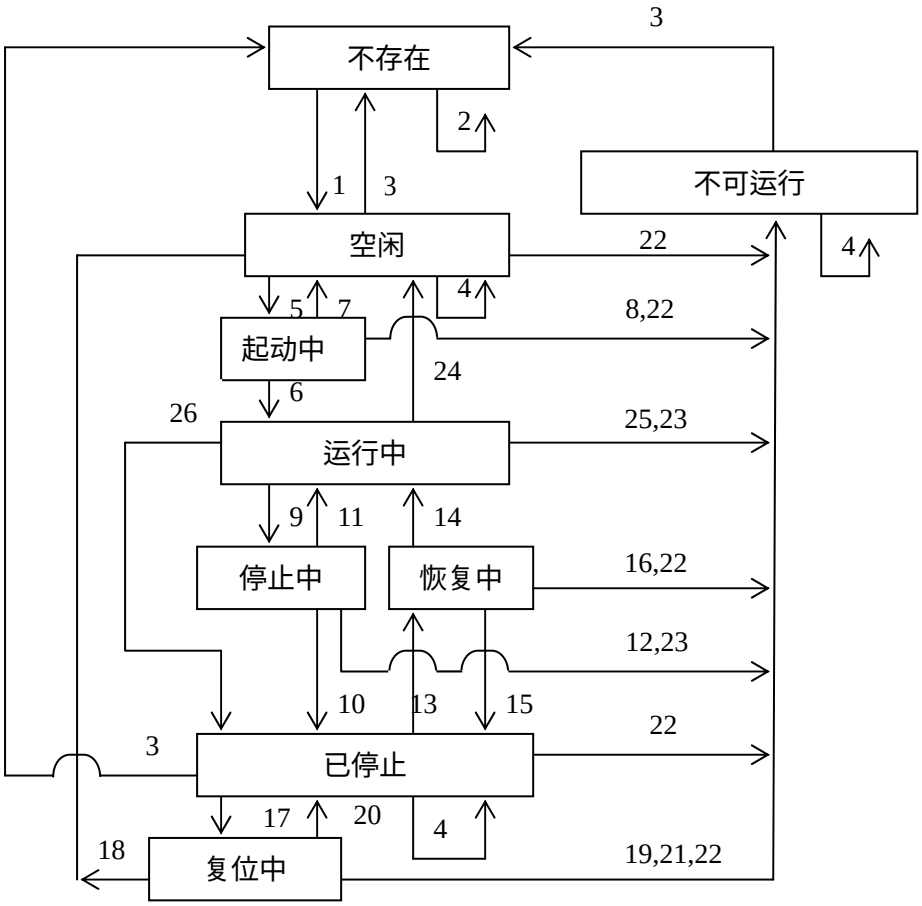

不存在
3
2
1
3
不可运行
空闲
22
4
5
7
4
8,22
起动中
24
6
26
25,23
运行中
9
11
14
停止中
恢复中
16,22
12,23
10
13
15
22
3
已停止
17
20
4
18
复位中
19,21,22

图 56. 状态图

4.6.4.2 状态转换

事件
\退出条件
⇒采取的动作

---

1 Create Program Invocation. ind
\域可用
且允许对域存取
⇒ .res(+)

2 Create Program Invocation ind
\域不可用
或不允许对域存取
⇒ .res(-)

3 Delete Program Invocation. ind
⇒ .res(+)

4 Delete Program Invocation. ind
⇒ .res(-)

5 Start. ind

6 起动成功执行
\域处于 READY 或 IN-USE 状态
⇒计数器加 1
Statr.res(+)

7 起动失败，未破坏
（如：域不处于 READY 或 IN-USE 状态）
⇒Start. res（-）

8 起动失败，破坏
⇒Start. res(-)

9 Stop. ind

10 成功停止执行
⇒计数器减 1
Stop. res(+)

11 停止失败，未破坏
⇒Stop.res(-)

12 停止失败，破坏
⇒计数器减 1
Stop. res(-)

13 Resume. ind

事件

\退出条件

⇒采取的动作

---

14 成功恢复执行

\域处于 READY 或在用 IN-USE 状态

⇒Resume.res(+)

15 恢复失败，未破坏

（如：域不处于 READY 或 IN-USE 状态）

⇒Resume. res(-)

16 恢复失败，破坏

⇒Resume. res(-)

17 Reset. ind

18 成功复位执行，Reusable（可再用的）=真(true)

⇒Reset. res(+)

19 成功复位执行，Reusable（可再用的）=假(false)

⇒Reset. res(+)

20 复位失败，未破坏

⇒Reset. res(-)

21 复位失败，破坏

⇒Reset. res(-)

22 Kill. ind

⇒ .res(+)

23 Kill. ind

⇒计数器减 1

.res(+)

24 程序结束，Reusable（可再用的）=真

⇒计数器减 1

25 程序结束，Reusable(可再用的)=假

⇒计数器减 1

26 程序停止

⇒计数器减 1

计数器加 1，减 1：见 State Machine Domain Download（状态机域下载

4.7 变量存取

4.7.1 模型描述

变量存取模型提供读和写变量的服务，并且动态定义和删除新变量表对象（Variable List Object）。

定义以下服务和对象

表 36. 变量存取服务和对象

服务	需确认/无需确认	对象
Read（读）	需确认	简单变量， 数组， 记录， 变量表
Write(写)	需确认	
Read With Type（带类型的读）	需确认	
Write With Type（带类型的写）	需确认	
Information Report（信息报告）	无需确认	
Information Report With Type（带类型的信息报告）	无需确认	
Phys Read （物理读）	需确认	物理存取对象
Phys Write （物理写）	需确认	
Define Variable List （定义变量表）	需确认	对象描述变量表， 变量表
Delete Variable List （删除变量表）	需确认	
- - -	- - -	数据类型
- - -	- - -	数据类型结构描述

4.7.2 变量存取对象

登入在 S-OD 中的对象为不可删除

4.7.2.1 物理存取对象

物理存取对象描述对一个实际的八位位组串的存取。这种八位位组串既不可能被定义，也可不能予以删除。

没有存储物理存取对象的显性对象描述。对用户来说，其地址和含义是已知的。

存取权通过服务上下关系协议（Service Context Agreement）进行控制，即该上下关系协议规定在此通信关系中是否允许物理存取。

对物理存取对象的服务：

-Phys Read（物理读）

-Phys Write（物理写）

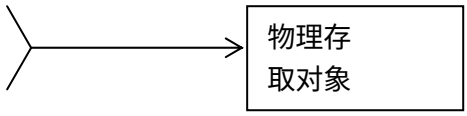


图 57. 物理存取对象服务

属性

对象：物理存取对象

关键属性：本地地址

属性：　　长度

本地地址

在已知其物理地址及其结构的前提下对对象进行存取，而不受有关设备的对象字典的控制，数据类型八位位组串被隐性采用。

长度

在读和写操作期间，对此对象的长度描述（以八位位组为单位）。

4.7.2.2　简单变量对象

简单变量对象表示单一的简单变量，其特征用一个已定义的数据类型来表述。变量存取对象简单变量的对象描述静态地存储在对象字典（S-OD）中。一个 PROFIBUS 简单变量是一个在该应用系统中真实存在的简单变量的映象，它由简单变量对象的对象描述所定义。在简单变量对象的对象描述中，只允许在组态数据类型集内的一个单一数据类型。

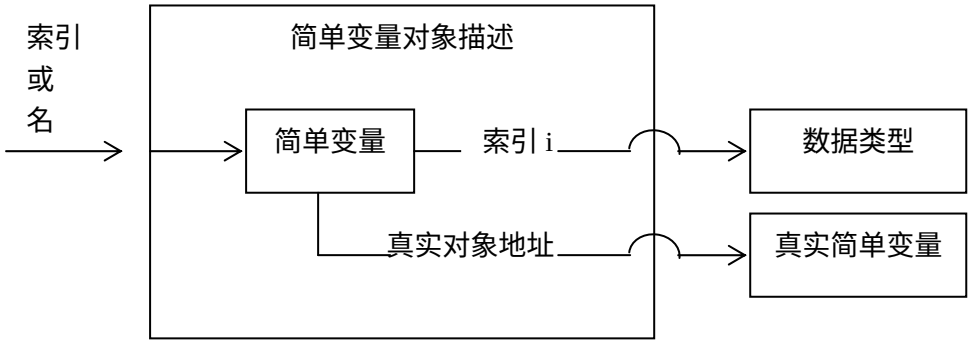


图 58. 简单变量概述

对简单变量对象的服务：

-OD Services（OD 服务）

-Read（读）

-Write（写）

-Read With Type（带类型的读）

-Write With Type（带类型的写）

-Information Report（信息报告）

-Information Report With Type（带类型的信息报告）

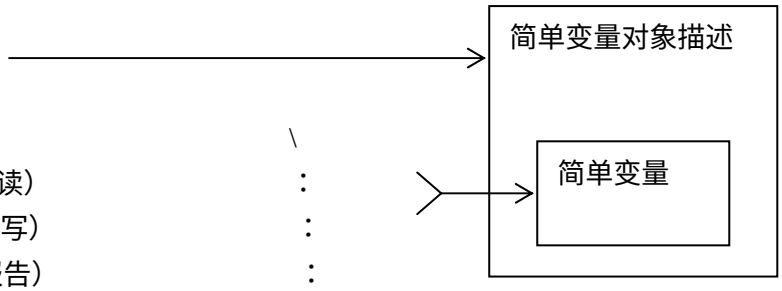


图 59. 简单变量服务

属性

对象：简单变量

关键属性：索引(Index)

关键属性：变量名(Variable Name)

属　　性：数据类型索引(Data Type Index)

属　　性：长度(Length)

属　　性：口令(Password)

属　　性：存取组(Access Group)

属　　性：存取权(Access Rights)

属　　性：本地地址(Local Address)

属　　性：扩展(Extention)

索引

对象的逻辑地址

变量名

对象名，其存在及其长度在 OD 对象描述中定义。

数据类型索引

在静态类型字典（ST-OD）中相应数据类型的逻辑地址

长度

以八位位组为单位的数据长度

口令

此属性包含存取权的密码

存取组

此属性说明特定存取组与对象的关系。当设置相应位时，此对象是一个组元。

表 37. 简单变量的存取组

位	含 义
7	存取组 1
6	存取组 2
5	存取组 3
4	存取组 4
3	存取组 5
2	存取组 6
1	存取组 7
0	存取组 8

存取权

此属性包含有关存取权的信息，在设置相应位时，即指定了相关的存取权。

表 38. 简单变量的存取权

位	名	含 义
7	R	对已登记口令的读取权
6	W	对已登记口令的写入权
3	Rg	对存取组的读取权
2	Wg	对存取组的写入权
15	Ra	对所有通信伙伴的读取权
14	Wa	对所有通信伙伴的写入权

逻辑地址

逻辑地址是真实对象的系统专用地址，它用于对象内部编址。若不执行这种映象，此属性将具有值 FFFFFFFF（十六进制）

扩展

此属性包含行规专用信息

在传送中 OD 的对象描述

//

索引	对象代码	数据类型索引	长度	口令	存取组
110	Var	3	2	17	

//

//

存取权	本地地址	变量名	扩展
Ra Wa	A3E5 hex	D.Krum siek	

//

图 60. 在 S-OD 内简单变量的对象描述（举例）

对象代码

简单变量对象的标记，在 Get OD 服务和 Put OD 服务中，它附加于对象属性被传送。

4.7.2.3 数组对象

数组对象包括相同数据类型的简单变量的一个集合

变量存取对象数组的对象描述静态地被存入对象字典（S-OD）。一个应用系统中存在的一个 PROFIBUS 数组到真实数组的映象由此对象的对象描述说明。对象描述表明该对象与数据类型的关系。对全部数组元素而言，只允许组态的数据类型集合中的一个单一数据类型。

数组对象可完全存取，或按元素存取。若对象将全部存取，对该服务必须用索引。若对象的一个元素将被存取，那么，对该服务在用索引时还必须用子索引。子索引 1 对对象的第 1 个元素进行存取。

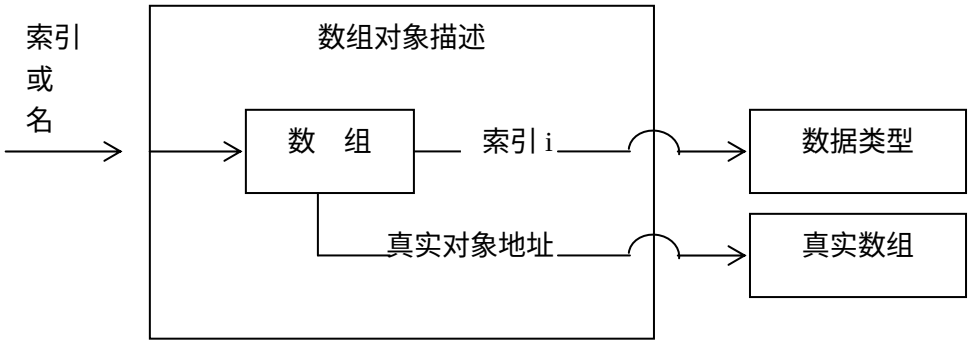


图 61. 数组概貌

对数组对象的服务：

-OD Services (OD 服务)

-Read（读）

-Write（写）

-Read With Type（带类型的读）

-Write With Type（带类型的写）

-Information Report（信息报告）

-Information Report With Type（带类型的信息报告）

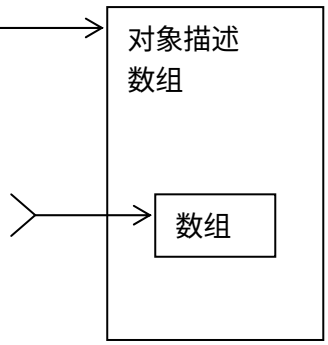


图 62. 数组服务

属性

对象：数组

关键属性：索引

属　　性：变量名

属　　性：数据类型索引

属　　性：长度

属　　性：元素数量

属　　性：口令

属　　性：存取组

属　　性：存取权

属　　性：本地地址

属　　性：扩展

索引

对象的逻辑地址

变量名

对象名，在 OD 对象描述内定义它的存在和长度

数据类型索引

在静态类型字典中相应数据类型的逻辑地址

长度

一个数组元素的长度，以八位位组为单位

口令

此属性包含存取权的密码

存取组

此属性为对象与专用存取组的关系。当设置相应位时，此对象为一个组元。

存取权

此属性包含有关存取权的信息。当设置相应位时，存在专用存取权。

表 39. 数组的存取组

位	含　义
7	存取组 1
6	存取组 2
5	存取组 3
4	存取组 4
3	存取组 5
2	存取组 6
1	存取组 7

0	存取组 8

表 40. 对数组的存取权

位	名	含　义
7	R	对已登记口令的读取权
6	W	对已登记口令的写入权
3	Rg	对存取组的读取权
2	Wg	对存取组的写入权
15	Ra	对所有通信伙伴的读取权
14	Wa	对所有通信伙伴的写入权

本地地址

本地地址是真实对象的系统专用地址，它用于对象的内部编址。若不执行此类映象，此属性将赋以值 FFFFFFFF（十六进制）。

扩展

此属性包含行规专用信息

在传送中 OD 的对象描述

//

索引	对象代码	数据类型索引	长度	元素数量	口令
121	Array	5	1	6	34

//

//

存取组	存取权	本地地址	变量名	扩展
	R W	1234 hex	M.Braun	

//

图 63. 在 S-OD 内数组的对象描述

对象代码

数组对象的标记，在 Get OD 和 Put OD 服务中，它附加于对象属性被传送。

4.7.2.4 记录对象

记录对象包括不同数据类型的简单变量的一个集合。

变量存取对象记录的对象描述被动态地存入对象描述（S-OD）中。

存在于一个应用系统中的一个 PROFIBUS 记录到实际记录的映象由此对象的对象说明来描述。对象说明表述该对象与一个组态的数据类型结构说明的关系。

记录对象可被全部存取或逐个元素存取。若对象要被全部存取，对此服务必须用索引。若要存取对象中的一个元素，对此服务必须在用索引时还用子索引，子索引 1 对对象的第一个元素进行存取。

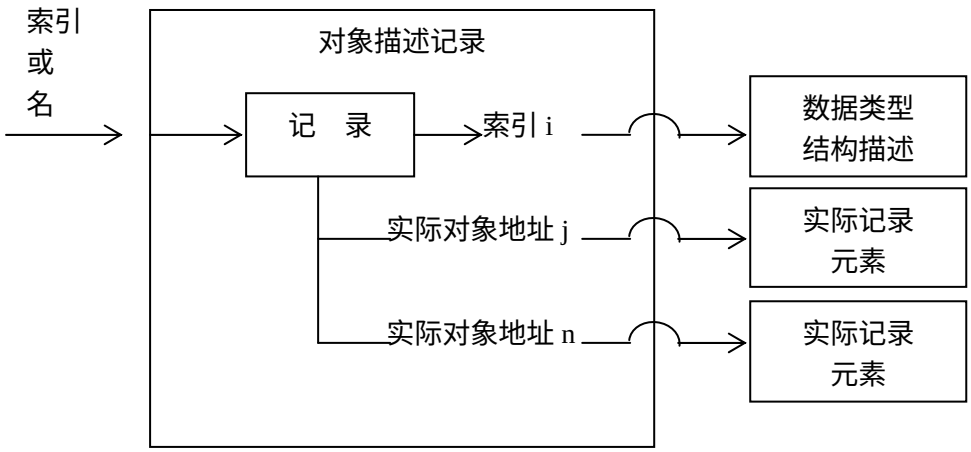


图 64. 记录概貌

对记录对象的服务

-OD Services(OD 服务)

-Read（读）

-Write（写）

-Read With Type（带类型的读）

-Write With Type（带类型的写）

-Information Report（信息报告）

-Information Report With Type（带类型的信息报告）

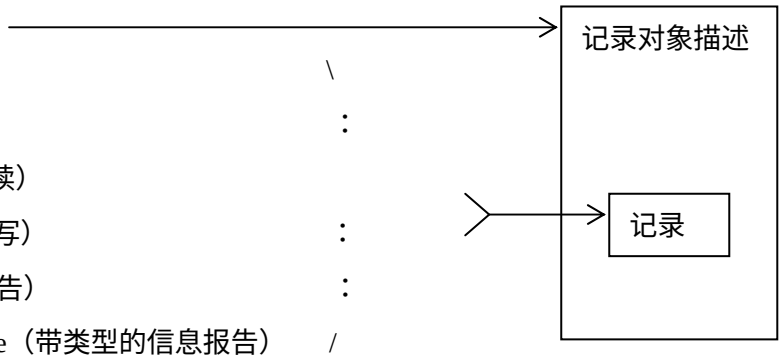


图 65. 记录服务

属性

对象：记录

关键属性：索引

关键属性：变量名

属　　性：数据类型索引

属　　性：口令

属　　性：存取组

属　　性：存取权

属　　性：扩展

属　　性：本地地址表

属　　性：本地地址

索引

对象的逻辑地址

变量名

对象名，在 OD 对象描述内定义它的存在和长度

数据类型索引

在静态类型字典（ST-OD）中有关数据类型结构说明的逻辑地址

口令

此属性包含存取权的密码

存取组

此属性为对象与专用存取组的关系。当设置相应位时，此对象为一个组元。

表 41. 记录的存取组

位	含　义
7	存取组　1
6	存取组　2
5	存取组　3
4	存取组　4
3	存取组　5
2	存取组　6
1	存取组　7
0	存取组　8

存取权

此属性包含有关存取权的信息。当设置相应位时，专用存取权存在。

表 42. 记录的存取权

位	名	含　义
7	R	对已登记口令的读取权
6	W	对已登记口令的写入权
3	Rg	对存取组的读取权
2	Wg	对存取组的写入权
15	Ra	对全部通信伙伴的读取权
14	Wa	对全部通信伙伴的写入权

扩展

此属性包含行规专用信息

本地地址表

此属性包含记录元素的内部地址，这些地址为实际对象的系统专用地址。它们用于对象的内部编址。若实际对象被顺序而无间隔地存储，那么只要给出本地地址（1）作为起始地址。若不执行这类映象变换，此属性应被设置为值 FFFFFFFF（十六进制数）。

在传送中 OD 的对象描述

索引	对象代码	数据类型索引	口令	存取组	存取权
111	Record	37	34		R

变量名	扩展	本地地址（1）	本地地址（2）
R.Breithaupt		43 A7 hex	62 AB hex

本地地址（3）	本地地址（4）	本地地址（5）
3942 hex	2324 hex	AC 56 hex

图 66. 在 S-OD 中记录的对象描述（举例）

对象代码

记录对象的标记，在 Get OD 服务和 Put OD 服务中，它附加于对象属性被传送。

4.7.2.5 变量表对象

变量表对象包括一个变量存取对象的集合。

变量存取对象变量表的对象描述被动态地存入对象字典（DV-OD）中，此对象描述包含一个来自 S-OD 的数组、记录和简单变量对象的索引表。

用 Define Variable List（定义变量表）和 Delete Variable List （删除变量表）服务可创建和删除一个变量表。

存取权应对 Define Variable List 服务予以声明。对单一对象的存取权在创建（Define Variable List）一个变量表时进行检查，如果所请求的存取权未超过单一对象的存取权，则仅创建变量表对象。

若相同的变量表有所请求的存取权存在，则不创建新的对象，但向客户机给出已存在对象的逻辑地址。在其它情形下，用所请求的存取权创建变量表。

仅被授权有适当的存取权的客户机才可删除一个变量表对象（Delete Variable List）。

若对象字典被刷新装载，则所有的变量表对象被删除。

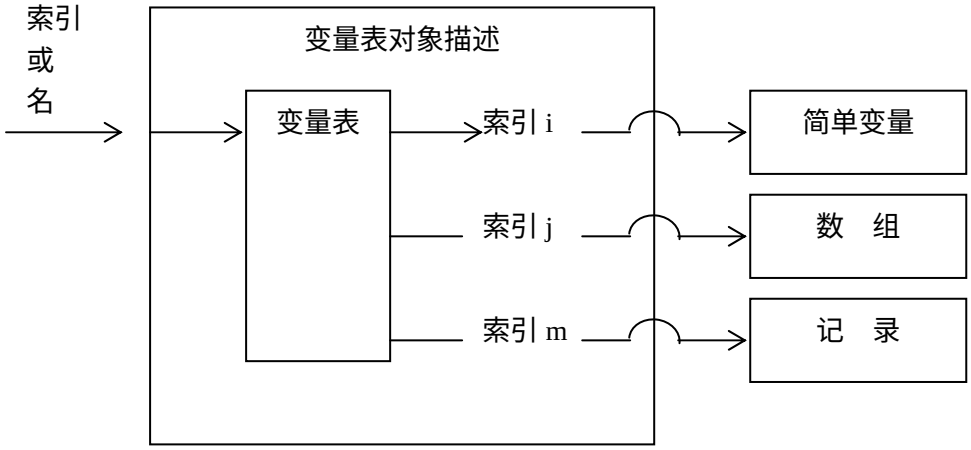


图 67. 变量表概貌

对变量表对象的服务：

-OD Services（OD 服务）

-Define Variable List（定义变量表）

-Delete Variable List（删除变量表）

-Read（读）

-Write（写）

-Read With Type（带类型的读）

-Write With Type（带类型的写）

-Information Report（信息报告）

-Information Report With Type（带类型的信息报告）

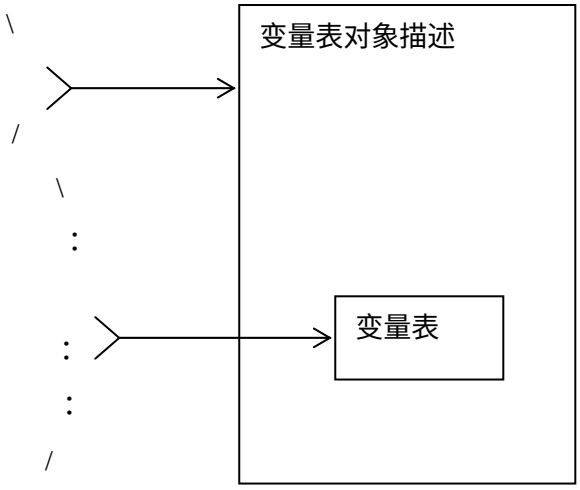


图 68. 变量表服务

属性

对象：变量表

关键属性：索引

关键属性：变量表名

属　　性：元素数量

属　　性：口令

属　　性：存取组

属　　性：存取权

属　　性：可删除

属　　性：元素索引表

属　　性：索引

属　　性：扩展

索引

对象的逻辑地址

变量表名

对象名称，在 OD 对象描述内定义它的存在和长度。

元素数量

此参数表述包括在变量表中的变量存取对象的数量

口令

此参数包含存取权的密码

存取组

此属性为对象与专用存取组的关系。当相应位被设置，此对象为一个组元。

表 43. 变量表的存取组

位	含　义
7	存取组　1
6	存取组　2
5	存取组　3
4	存取组　4
3	存取组　5
2	存取组　6
1	存取组　7
0	存取组　8

存取权

此属性包含有关存取权的信息。当设置相应的位时，专用存取权存在。

表 44. 变量表的存取权

位	名	含　义
7	R	对已登记口令的读取权
6	W	对已登记口令的写入权
5	D	对已登记口令的删除权
3	Rg	对存取组的读取权
2	Wg	对存取组的写入权
1	Dg	对存取组的删除权
15	Ra	对所有通信伙伴的读取权
14	Wa	对所有通信伙伴的写入权
13	Da	对所有通信伙伴的删除权

可删除

此属性指示是否可用 Delete Variable List 服务对变量表对象删除（真）或不删除（假）

元素索引表

此属性包含被组合在此变量表中的变量存取对象的索引

扩展

此属性包含行规专用信息

在传送中 OD 的对象描述

索引	对象代码	元素数量	口令	存取组
220	Var-List	8	34	

存取权	可删除	元素索引（1）	元素索引（2）	元素索引（3）
R WD	真	142	127	158

元素索引（4）	元素索引（5）	元素索引（6）	元素索引（7）	元素索引（8）
172	108	196	134	188

变量表名	扩展

Z.Szabo	

图 69. 在 DV-OD 中变量表的对象描述

对象代码

变量表对象标记，对于 Get OD 服务和 Put OD 服务，它附加于对象属性一起传送。

4.7.2.6 数据类型对象

一个数据类型表述数值的一个集合和适用于加于这些数值的操作的一个集合的特征。该数据类型定义句法、变量范围及其在通信系统内的表达。用对象描述中的一个相应属性来指示一个对象的数据类型。为效率的缘故，类型定义不可远程进行。在静态类型字典（ST-OD）中它有一个固定的配置。

PROFIBUS 规范定义可自由组态的数据类型和标准数据类型。用索引 1 将它们存储在静态类型字典（ST-OD）中的起始地址中。未使用的标准数据类型在 ST-OD 中标记为未组态（见在 OD 中的数据类型对象表达）。

对数据类型对象的服务：

无

属性

对象：数据类型

关键属性：索引

属　　性：描述

属　　性：符号长度

属　　性：符号

索引

对象的逻辑地址（索引的语义是固定的）

描述

此属性包含一个数据类型的文字说明，它包括符号长度和符号

符号长度

此属性包含符号的长度。值 0 表示没有使用符号。

符号

此属性包含一个 0-32 八位位组的可视字符串，作为符号说明

在传送中 OD 的结构

在 ST-OD 内，一个数据类型的登入项具有以下结构

索引	对象代码	描　　述	
		符号长度	符号

01	数据类型	7	Boolean（布尔数）

图 70. 在 ST-OD 中数据类型的对象描述

对象代码

数据类型对象的标记，对于 Get OD 服务和 Put OD 服务，它附加于对象属性被传送。

Null 对象<=>不予组态

例

索引	对象代码	描述	
01	数据类型	布尔数	布尔数
02	Null 对象		8 位整数，未予组态
03	数据类型	16 位整数	16 位整数
- - -			
76	数据类型	xy 专用	应用专用

图 71. 在 ST-OD 中数据类型的对象描述（举例）

4.7.2.7 数据类型结构描述对象

数据类型结构描述对象表述记录的大小和结构（见记录对象）

数据类型结构描述对象的对象描述在静态类型字典中有一固定的组态。它包含一个元素表，该元素表包括相应记录元素的数据类型和长度。作为单一元素的数据类型，只允许在静态类型字典内已组态的数据类型的集合中选用。

对数据类型结构描述对象的服务：无

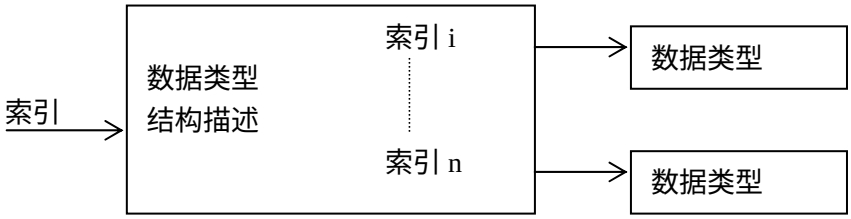


图 72. 数据类型结构描述的概貌

属性

对象：数据类型结构描述

关键属性：索引

属　　性：元素数量

属　　性：元素表

属　　性：数据类型索引

属　　性：长度

索引

对象的逻辑地址

元素数量

结构数据类型的元素数量

元素表

此属性包含一个元素表，它包括相应记录元素的数据类型和长度。

数据类型索引

元素 n 的数据类型的逻辑地址

长度

元素 n 的长度，以八位位组为单位。

举例

在传送中 OD 的结构

索引	对象代码	元素数量	数据类型索引（1）	长度（1）
39	Ds	5	7	4

数据类型索引（2）	长度（2）	数据类型索引（3）	长度（3）
9	12	8	4

数据类型索引（4）	长度（4）	数据类型索引（5）	长度（5）
9	8	8	4

图 73. 在 ST-OD 中数据类型结构描述的对象描述（例）

对象代码

数据类型结构描述的标记，对于 Get OD 和 Put OD 服务，它附加于对象属性被传送。

4.7.3 变量存取服务

4.7.3.1 读（Read）

用此服务，可读取通信伙伴的简单变量对象、数组和记录的值。用子索引可存取数组和记录的单个元素。

表 45. 读

参数名	.req .ind	.res .con
变元 存取说明 索引 变量名 变量表名 子索引	M M S S S U	
结果（+） 数据		S M
结果（-） 出错类型		S M

变元

变元包含服务请求的服务专用参数

存取说明

此参数表述索引或名称是否用于编址

索引

对象的逻辑地址

变量名

此参数为对象的名称

变量表名

此参数为变量表对象的名称

子索引

此参数包含对象的逻辑子地址

结果（+）

结果（+）参数应指示服务已成功完成

数据

已读取的数据。应按照在编码这一章内数据类型的说明将对象的数据应映象为参数数据。

结果（-）

结果（-）应指示服务请求未曾成功完成

出错类型

此参数包含有关服务请求未曾成功完成的原因信息

4.7.3.2 写（Write）

用此服务将数值写入对象简单变量、数组、记录和变量表。用子索引可对数组和记录中的单个元素进行存取。

表 46. 写

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
变量名	S	
变量表名	S	
子索引	U	
数据	M	
结果（+）		S
结果（-）		S
出错类型		M

变元

变元包含服务请求的服务专用参数

存取说明

此参数表述索引或名称是否用于编址

索引

对象的逻辑地址

变量名

此参数为变量表对象名称

变量表名

此参数为变量表对象名称

子索引

此参数包含对象的逻辑子地址

数据

将被写入的数据。应按在编码一章中数据类型的说明将对象的数据应映象为参数数据。

结果（+）

结果（+）参数应指示服务已成功完成

结果（-）

结果（-）应指示服务请求未曾成功完成

出错类型

此参数包含有关服务未曾成功完成的原因

4.7.3.3 物理读（Phys Read）

用此服务读取通信伙伴的物理存取对象的数值。

表 47. 物理读

参数名	.req .ind	.res .con
变元 本地地址 长度	M M M	
结果（+） 数据		S M
结果（-） 出错类型		S M

变元

变元包含服务请求的服务专用参数

本地地址

对象的物理地址

长度

待读的八位位组数

结果（+）

结果（+）参数将指示服务已成功完成

数据

已读取的数据

结果（-）

结果（-）将指示服务请求未曾成功完成

出错类型

此参数包含有关服务未曾成功完成的原因信息

4.7.3.4 物理写（Phys Write）

用此服务将一新值赋给物理存取对象

表 48. 物理写

参数名	.req .ind	.res .con
变元 本地地址 数据	M M M	
结果（+）		S
结果（-） 出错类型		S M

变元

变元包含服务请求的服务专用参数

本地地址

对象的物理地址

数据

待写的数据

结果（+）

结果（+）参数将指示服务已成功完成

结果（-）

结果（-）将指示服务未曾成功完成

出错类型

此参数包含此服务未曾成功完成的原因信息

4.7.3.5 信息报告（Information Report）

用此信息报告服务传送简单变量、数组、记录和变量表对象的值。此服务的执行不由通信伙伴确认。数组和记录中的单一元素可用子索引存取。

此服务可在 LLI 映象至 Layer 2（第 2 层）广播或群播（multicast）。这样，可以将数值传送至网络中的全部站或某些站中。

表 49. 信息报告

参数名	.req .ind
变元	M
优先权	M
存取说明	M
索引	S
变量名	S
变量表名	S
子索引	U
数据	M

变元

变元包含服务请求的服务专用参数

优先权

此参数指示此服务的优先权，它可有以下值。

真：高优先权

假：低优先权

存取说明

此参数表述索引或名称是否用于编址

索引

在源（Source）OD 中对象的逻辑地址

变量名

此参数为变量对象名称

变量表名

此参数为变表对象名称

子索引

此参数为对象的逻辑子地址

数据

此参数包含数据。应按照编码一章中数据类型的说明将对象的数据映象为参数数据。

4.7.3.6 定义变量表（Define Variable List）

用此服务在通信伙伴上建立一个变量表对象。对 Define Variable List 服务，由服务器自动计算元素属性的数目。作为客户机的用户将确保变量表服务的数据可在一个单一的 PDU 内传送。由 Define Variable List 服务所建立的一个变量表，其属性 Deletable（可删除）设置

为“真”值。

表 50. 定义变量表

参数名	.req .ind	.res .con
变元	M	
口令	M	
存取组	M	
存取权	M	
变量表	M	
变量索引	S	
变量名	S	
变量表名	U	
扩展	U	
结果（+）		S
索引		M
结果（-）		S
出错类型		M

变元

变元包含该服务请求的服务专用参数。若不支持存取权（支持存取保护（Access Protection Supported）=假），那么参数口令、存取组和存取权均无效。

口令

此参数表述口令属性将设置的值

存取组

此参数表述存取组属性将设置的值

存取权

此参数表述存取属性将设置的值

变量表

此参数表述被组合在变量表中的变量的地址。每个地址可以是逻辑地址，也可是一个名称。

变量索引

一个变量的索引

变量名

一个变量的名称

变量表名

此参数表述变量表对象的名称属性将设置的值。

扩展

此变量表述变量表对象的扩展属性将设置的值

结果（+）

结果（+）参数将指示该服务已成功完成

索引

对象的逻辑地址

结果（-）

结果（-）参数将指示该服务请求未曾成功完成

出错类型

此参数包含有关该服务未曾成功完成的原因信息

4.7.3.7 删除变量表（Delete Variable List）

如果对此对象具有存取权，用此服务在通信伙伴上删除一个变量表对象。但只有其可删除属性为“真”值的变量表对象，才可以被删除。

表 51. 删除变量表

参数名	.req .ind	.res .con
变元 存取说明 索引 变量表名	M M S S	
结果（+）		S
结果（-） 出错类型		S M

变元

变元包含该服务请求的服务专用参数

存取说明

此参数表述索引或名称是否被用于编址

索引

对象的逻辑地址

变量表名

此参数是变量表对象的名称

结果（+）

结果（+）参数将指示该服务已成功完成

结果（-）

结果（-）参数将指示该服务未曾成功完成

出错类型

此参数包含有关该服务未曾成功完成的原因信息

4.7.3.8 带类型的读（Read with Type）

用此服务可读取通信伙伴的简单变量对象、数组和记录的值和数据类型描述。数组和记录的单个元素可用子索引予以读取。

表 52. 带类型的读

参数名	.req .ind	.res .con
变元 存取说明 索引 变量名 变量表名 子索引	M M S S S U	
结果（+） 类型描述 数据		S M M
结果（-） 出错类型		S M

变元

变元包含服务请求的服务专用参数

存取说明

此参表述索引或名称是否用于编址

索引

对象的逻辑地址

变量名

此参数为变量表对象的名称

变量表名

此参数为变量表对象的名称

子索引

此参数包含对象的逻辑子地址

结果（+）

结果（+）将指示服务已成功完成

数据

待读数据。对象的数据应按编码这一章中数据类型的描述变换为参数数据。

类型描述

此参数包含数据类型描述（见参数类型描述）

结果（-）

结果（-）将指示服务请求未曾成功完成

出错类型

此参数包含有关服务未曾成功完成的原因信息

4.7.3.9 带类型的写（Write With Type）

用此服务可向对象简单变量、数组和记录和变量表写入数值。数据类型描述也附加于被写入的数据。数组和记录的单个元素可用子索引进行存取。

表 53. 带类型的写

参数名	.req .ind	.res .con
变元	M	
存取说明	M	
索引	S	
变量名	S	
变量表名	S	
子索引	U	
类型描述	M	
数据	M	
结果（+）		S
结果（-）		S
出错类型		M

变元

变元包含服务请求的服务专用参数

存取说明

此参数表述索引或名称是否用于编址

索引

对象的逻辑地址

变量名

此参数为变量对象名称

子索引

此参数包含对象的逻辑子地址

类型描述

此参数包含数据类型描述

数据

待写数据。对象的数据要按照在编码章节中的数据类型描述映象为参数数据。

结果（+）

结果（+）参数指示此服务已成功完成

结果（-）

结果（-）指示服务请求未曾成功完成

出错类型

此参数包含有关服务未曾成功完成的原因信息

4.7.3.10 带类型的信息报告（Information Report With Type）

用 Information Report(信息报告)服务传送对象简单变量、数组、记录和变量表的值和数据类型描述。此服务的执行不由通信伙伴确认。数组和记录的单个元素可用子索引存取。

此服务可在 LLI 映象到 Layer 2（第 2 层）广播或群播（multicast）。这样有可能在网络中将数值传送到所有的站或某些站去。

表 54. 带类型的信息报告

参数名	.req .ind
变元	M
优先权	M
存取说明	M
索引	S
变量名	S
变量表名	S
子索引	U
类型描述	M
数据	M

变元

变元包含服务请求的服务专用参数

优先权

此参数指示该服务的优先权，它可能有以下值。

真：高优先权

假：低优先权

存取说明

此参数表述索引或名称是否用于编址

索引

在源 OD 内对象的逻辑地址

变量名

此参数为变量对象名称

变量表名

此参数为变量对象名称

子索引

此参数包含对象的逻辑子地址

类型描述

此参数包含数据类型描述

数据

此参数包含数据。对象的数据应按照在编码章节中的数据类型描述映象为参数数据。

4.8 事件管理

4.8.1 模型描述

事件服务用于从一个设备向另外的设备（或在广播方式中，向所有其它设备）发送一个重要消息。产生一个事件的检测条件由用户负责确定。当条件满足时应用程序调用 Event Notification（事件通告）服务。监视及确认（可选）的检查是用户的责任。

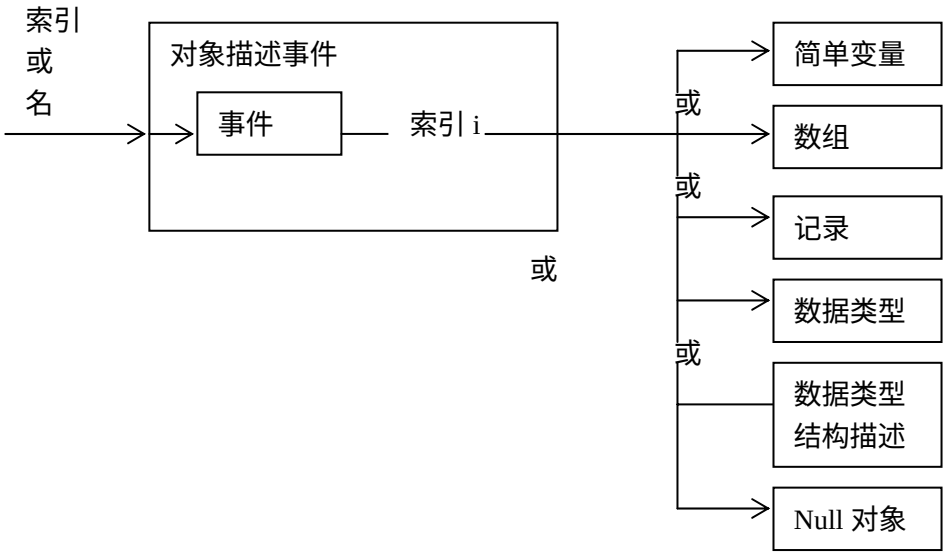


图 74. 事件概貌

对事件对象的服务：

-OD Services(OD 服务)

-Alter Event Condition Monitoring（改变事件条件监视）

-Event Notification（事件通告）

-Event Notification with Type（带类型的事件通告）

-Acknowledge Event Notification（确认事件通告）

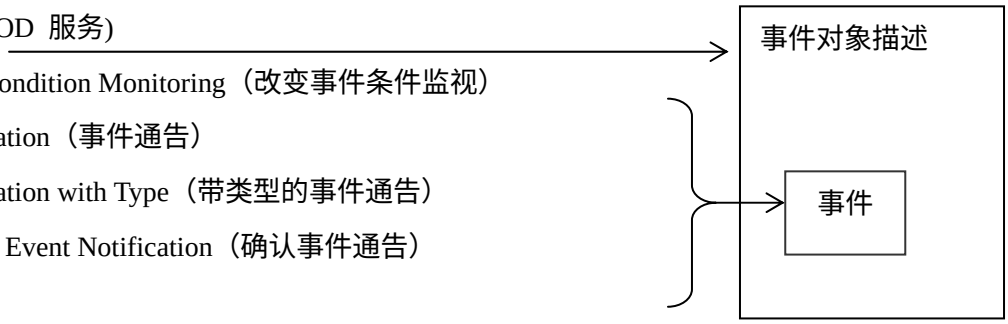


图 75. 事件服务

用户用 Event Notification（事件通告）服务发送一个事件通告。用户用 Event Notification 服务传送一个计数器值（事件数）和可选数据（事件数据），如测量值、状态、单位。

计数器值的管理（增加）是用户的责任。事件通告可以表达一个报警的集合，在该报警中的数据可以包含有关哪个通道其报警集合已被触发的信息。还有在此情形下触发条件的逻辑操作以及数据的逻辑操作是应用程序的部分。事件通告的接收者可以用 Acknowledge Event Notification(确认事件通告)服务确认 Event Notification（事件通告）。

用 Acknowledge Event Notification（确认事件通告）服务传送计数器值。计数值服务于 Event Notification（事件通告）和 Acknowledge Event Notification（确认事件通告）关连的可靠性。监视和确认检查是用户的责任。

Event Notification（事件通告）的接收者可以用 Alter Event Condition Monitoring（改变事件条件监视）服务锁定或不锁定 Event Notification（事件通告）。

4.8.2 事件对象

4.8.2.1 属性

对象：事件

关键属性：索引

关键属性：事件名

属　　性：索引事件数据

属　　性：长度

属　　性：口令

属　　性：存取组

属　　性：存取权

属　　性：启用（Enabled）

属　　性：扩展

索引

事件对象的逻辑地址

事件名

对象名称。其存在与长度在 OD 对象描述中定义

索引事件数据

事件数据索引。此登入项定义读取事件数据的索引，或定义事件数据的数据类型的对象描述或数据类型结构描述的对象描述可读取的索引。在此事件对象无数据存在的情形下，该索引事件数据将包含一个 Null 对象的索引。

长度

若索引事件数据指向一数据类型，那么此属性包含事件数据的长度（以八位位组为单位），否则此属性是无意义的。

口令

此属性包含存取权的口令

存取组

此属性确定对象与特定存取组的关系。在相应位被设置时，此对象为一个组元。

表 55. 事件的存取组

位	含 义
7	存取组 1
6	存取组 2
5	存取组 3
4	存取组 4
3	存取组 5
2	存取组 6
1	存取组 7
0	存取组 8

存取权

此属性包含有关存取权的信息。当相应位被设置时，存在专用存取权。

表 56. 事件的存取权

位	名	含 义
6	W	对已登记口令的确认权
5	D	对已登记口令的启用/停用权
2	Wg	对存取组的确认权
1	Dg	对存取组的启用/停用权
14	Wa	对全部通信伙伴的确认权

13	Da	对全部通信伙伴的启用/停用权

启用

按状态机的事件对象的状态

启用(Enabled)=真<=>未锁定（UNLOCKED）

启用(Enabled) =假<=>锁定（LOCKED）

扩展

此属性包含行规专用信息

4.8.2.2 在传输中 OD 的对象描述

索引	对象代码	索引事件数据	长度	口令	存取组
130	Event	39	2	17	

存取权	启用	事件名	扩展
WD		V. Buding	

图 76. 事件在 S-OD 中的对象描述（举例）

对象代码

事件对象标记，对 Get OD 和 Put OD 服务，它附加在对象属性中一起被传送。

4.8.3 事件管理服务

4.8.3.1 事件通告（Event Notification）

Event Notification（事件通告）服务允许传送一个事件通告

表 57.事件通告

参数名	.req .ind
变元	M
优先权	M
存取说明	M
索引	S
事件名	S
事件号	M
事件数据	D

变元

变元包含服务请求的服务专用参数

优先级

此参数定义此服务的优先级，它可有下列值：

真：高优先权

假：低优先权

存取说明

此参数表述索引或名称是否用于编址

索引

此参数定义该服务与哪个事件对象有关

事件名

此参数为事件对象名称

事件号

此参数包含计数值

事件数据

此参数包含数据。对象的数据应按照编码这一章中数据类型描述映象为参数数据。索引被存入作为索引事件数据事件对象的对象描述中。

4.8.3.2 应答事件通告（Acknowledge Event Notification）

此服务允许对一个事件通告予以确认

表 58. 应答事件通告

参 数 名	.req .ind	.res .con
变元 存取说明 索引 事件名 事件号	M M S S M	
结果（+）		S
结果（-） 出错类型		S M

变元

变元包含服务请求的服务专用参数

存取说明

此参数表述索引或名称是否用于编址

索引

此参数定义此服务与哪个事件对象有关

事件名

此参数为事件对象名称

事件号

此参数包含计数器的值

结果（+）

结果（+）参数将指示服务已成功完成

结果（-）

结果（-）将指示服务请求未曾成功完成

出错类型

此参数包含有关该服务未曾成功完成的原因信息

4.8.3.3 改变事件条件监视（Alter Event Condition Monitoring）

此服务允许事件对象锁定或不锁定

表 59. 改变事件条件监视

参 数 名	.req .ind	.res .con
变元 存取说明 索引 事件名 启用	M M S S M	
结果（+）		S
结果（-） 出错类型		S M

变元

变元包含此服务请求的服务专用参数

存取说明

此参数表述索引或名称是否用于编址

索引

此参数定义此服务与哪个事件对象有关

事件名

此参数为事件对象名称

启用

此布尔类型的参数定义事件对象的启用属性将被设置为什么值

结果（+）

结果（+）参数将指示该服务已成功完成

结果（-）

结果（-）将指示该服务请求未曾成功完成

出错类型

此参数包含有关该服务未曾成功完成的原因信息

4.8.3.4 带类型的事件通告（Event Notification With type）

此服务允许传送一个事件通告。通告包含数据和数据类型描述，此服务是一种无需确认的服务。

表 60. 带类型的事件通告

参 数 名	.req .ind
变元	M
优先权	M
存取说明	M
索引	S
事件名	S
事件号	M
类型描述	O
事件数据	O

变元

变元包含该服务请求的服务专用参数

优先权

此参数定义此服务的优先权，它可有下列值：

真：高优先权

假：低低先权

存取说明

此参数表述索引或名称是否用于编址

索引

此参数定义该服务与哪个事件对象有关

事件名

此参数为事件对象名称

事件号

此参数包含计数器值

类型描述

此参数包含数据类型描述（见下面的参数类型描述）。仅当存在事件数据参数时，它才会有此参数。

事件数据

此参数包含数据。仅当存在类型描述参数时，才会用此参数。对象的数据将按照在编码章节中数据类型描述变换为参数数据。在事件对象的对象描述中，索引作为索引事件数据存入。

4.8.4 状态机

4.8.4.1 状态机描述

LOCKED（锁定）

在 LOCKED 状态（Enabled=假），无事件通告发送。

UNLOCKED（不锁定）

通常在 UNLOCKED 状态（Enabled=真），发送事件通告。

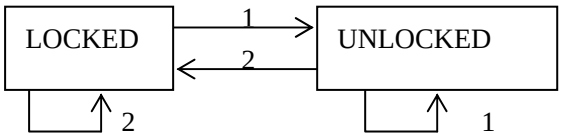
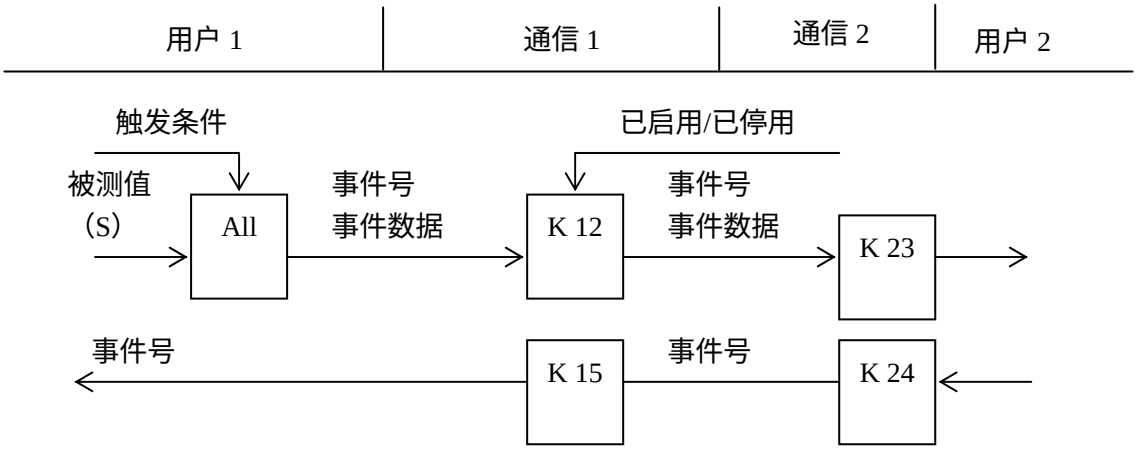


图 77. 事件状态机

4.8.4.2 状态转换

事件 \退出条件 ⇒采取的动作
1 Alter Event Condition Monitoring. ind \Enabled =true（真） ⇒ .res+ 2 Alter Event Condition Monitoring. ind \Enable=false（假） ⇒ . res+

4.8.5 举例：事件管理服务



All 用户 1 以已确定的条件与被测量值进行比较，若一个或多个条件被满足，于是调用一个事件通告服务（K 12）。在此服务中，事件号和事件数据作为参数被传送。

K 12 -未锁定

事件通告服务将事件号和事件数据传送给通信 2（K 23）

-锁定 事件通告服务停用，不会产生进一步的动作。

K 23 通信 2 从通信 1 接收事件通告服务，并将事件号和事件数据传送给用户 2。

K 24 用户 2 调用确认事件通告服务。在此服务中事件号作为参数被传送，该服务将事件号传送给通信 1。

K 15 通信 1 从通信 2 接收事件通告确认服务，并将事件号传送到用户 2。

图 78. 事件服务举例

触发条件：

在应用程序中对可能触发一 个事件通告的任何判别

已启用/已停用：

对 Alter Event Condition Monitoring(改变事件条件监视)服务的顺序在此不作进一步说明

测量值（S）：

在应用程序中任何一个与触发条件组合后可能触发事件的信号。

事件数据：

随 Event Notification (事件通告)服务作为一个参数被传送的数据。数据类型和语义是应用特定的，如

-事件代码（原因，来，去）

-测量值

-通道号（用于收集报警）

-时间

-通告丢失

-确认成功

4.9 FMS 与 LLI 间的接口

FMS 对低层接口（Lower Layer Interface，LLI）的耦合在本节予以规定。LLI 在此接口向 FMS 提供服务，此接口服务于建立和解除连接，以及传送数据。

4.9.1 服务概述

向 FMS 提供以下服务：

-Associate(联接) （建立一个连接）

-Abort（中止） （解除一个连接）

-Data Transfer Confirmed(数据传送需确认) （由用户确认数据传送）

-Data Transfer Acknowledged（数据传送需应答）（由 LLI 应答数据传送）

-Data Transfer Unconfirmed（数据传送无需确认）（数据传送无需确认）

这些服务用其对应的服务原语来描述：

-Associate (ASS. req, ASS.ind，ASS. res, ASS. con)

-Abort (ABT. req, ABT. ind)

-Data Transfer Confirmed (DTC. req, DTC. ind,DTC.res,DTC.con)

-Data Transfer Acknowledged (DTA. req, DTA. ind)

-Data-Transfer-Unconfirmed (DTU. req, DTU. ind)

关于 LLI 服务，其服务原语及其参数的详细描述请参考低层接口（Lower Layer Interface）。

4.9.2 FMS 服务对 LLI 服务的映象

FMS 服务用以下方式映象到 LLI 服务：

-初始化服务映象为连接服务，参数数据包含初始化 PDU。

-中止服务直接映象为 LLI 异常终止服务，无 FMS PDU 产生。

-所有需确认 FMS 服务映象为 LLI 的数据传送需确认服务，参数数据包含对应的 FMS PDU。

-若 FMS CRL 的属性 CREL 类型具有“真”值，所有无需确认 FMS 服务映象为 LLI 的数据传送确认服务，参数数据包含对应的 FMS PDU。

-若 FMS CRL 的属性 CREL 类型具有“假”值，所有无需确认 FMS 服务映象为 LLI 的数据传送无需确认服务。参数数据包含相应的 FMS PDU。

-FMS 服务拒绝映象为服务原语 DTC. Res。参数数据包含拒绝 PDU。

4.10 FMS 与 FMA 7 间的接口

本节阐述 FMS 和现场总线管理层 7（Fieldbus Management Layer 7，FMA7 ）的耦合。FMS 在此接口为 FMS 7 提供组态、标识和复位 FMS 的服务。

4.10.1 本地 FMS 管理服务概述

FMS 向 FMA 7 提供以下服务：

-FMS Disable (FMS 停用)　　-FMS Read CRL　(FMS 读 CRL)

-FMS Load CRL（FMS 装载 CRL）　-FMS Indent（FMS 标识）

-FMS Enable　（FMS 启用）　　-FMS Reset（FMS 复位）

4.10.1.1 FMS 停用（FMS Disable）

用此服务停用 FMS 的数据传送。所有连接均终止，而 FMS 用户得到一个带原因代码 13 的中止。由 FMS 用户或由 LLI 发出的建立连接的请求被拒绝，同时带有原因代码为 13 的中止。此服务无参数，并且总是导致一个肯定的确认。

表 61. FMS 停用

参 数 名	.req	.con
变元	M	
结果（+）		M

变元

对此服务请求，变元不包含参数

结果（+）

结果（+）参数将指示服务已成功完成

4.10.1.2 FMS 装载通信关系表 （FMS Load CRL）

在 FMS CRL 中用此服务登入一个通信关系的 FMS CRL 首部，或送入 FMS CRL Entry（FMS CRL 登入项）的静态部分。

表 62. FMS 装载 CRL

参 数 名	.req	.con
变元 FMS CRL 静态登入项	M M	
结果（+）		S
结果（-） FMA7 出错类型		S M

变元

变元包含服务请求的服务专用参数

FMS CRL 静态登入项

此参数包含一个 FMS CRL 登入项的 FMS CRL 首部或静态部分

结果（+）

结果（+）参数将指示服务已成功完成

结果（-）

结果（-）将指示服务请求未曾成功完成，在此情况下无 FMS CRL 登入项被写入 FMS CRL。

FMA 7 出错类型

此参数包含有关服务未曾成功完成的原因信息

4.10.1.3 FMS 启用（FMS Enable）

用此服务使 FMS 启用。在此后由用户或 LLI 可以建立一个新的连接。

表 63. FMS 启用

参 数 名	.req	.con
变元	M	
结果（+）		S

结果（-） FMA 7 出错类型 出错 CREF		S M C

变元

对此服务请求，变元不包含参数

结果（+）

结果（+）参数将指示服务已成功完成

结果（-）

结果（-）将指示服务请求未曾成功完成

FMA 7 出错类型

此参数包含有关服务未曾成功完成的原因信息

出错 CREF

此参数确定在哪个 CREF 上 FMS Enable 服务被 FMS 中止，此参数的存在与 FMA 7 出错类型有关。

4.10.1.4 FMS 读通信关系表（FMS Read CRL）

用此服务读取 FMS CRL 首部或 FMS CRL 登入项

为读取 FMS CRL 首部将用通信引用 0。为读取一个 FMS CRL 登入项，将指示相应的通信引用。若此 CREF 未用，则此服务以结果（-）予以回答。

表 64. FMS 读 CRL

参 数 名	.req	.con
变元 期望的 CREF	M M	
结果（+） FMS CRL 登入项		S M
结果（-） FMA 7 出错类型		S M

变元

变元包含该服务请求的参数

期望的 CREF

为读取 FMS CRL 首部，应给出“ 0 ”值，否则应给出期望的 FMS CRL 登入项的 CREF。

结果（+）

结果（+）参数将指示服务已成功完成

FMS CRL 登入项

此参数包含 FMS CRL 首部或一个 FMS CRL 登入项

结果（-）

结果（-）将指示服务请求未曾成功完成

FMA 7 出错类型

此参数包含有关服务未曾成功完成的原因信息

4.10.1.5 FMS 标识（FMS Ident）

此服务提供 FMS 标识的信息

表 65. FMS 标识

参 数 名	.req	.con
变元	M	
结果（+）		S
厂商名		M
控制器类型		M
硬件版本		M
软件版本		M
结果（-）		S
FMA 7 出错类型		M

变元

对此服务请求，变元不含参数。

结果（+）

结果（+）参数将指示服务已成功完成

厂商名

此参数标识 FMS 的厂商

控制器类型

此参数标识实现 FMS 所用的硬件类型（如处理器、控制器、专用集成电路）

硬件版本

此参数标识 FMS 的硬件版本

软件版本

此参数标识 FMS 的软件版本

结果（-）

结果（-）将指示服务请求未曾成功完成

FMA 7 出错类型

此参数包含有关服务未曾成功完成的原因信息

4.10.1.6 FMS 复位（FMS Reset）

用此服务执行 FMS 复位。在复位后 FMS 执行一次冷起动，如同在上电后的方式一样。

表 66.FMS 复位

参 数 名	.req	.con
变元	M	
结果（+）		S

变元

对此服务请求变元不含参数

结果（+）

结果（+）参数将指示服务已成功完成

4.10.2 FMA 7 出错类型

此参数包含服务未曾成功完成的原因信息，本参数可以有以下值：

-No Resource（无资源）

在超出可提供的资源时，显示此错误

-No FMS CRL Entry（无 FMS CRL 登入项）

发现无期望的 CREF 的登入项

-FMS CRL Parameter Invalid（FMS CRL 参数无效）

-FMS State Conflict（FMS 状态冲突）

在当前的 FMS 状态不能执行该服务

-Other（其它）

出错原因不是以上的任一个

4.11 FMS 的操作行为

由 FMS 基本状态机表达 FMS 的操作行为

4.11.1 FMS 起动

在上电后或由 FMA 7 复位 FMS 后，FMS 起动。

FMS CRL 的动态部分由 FMS 建立并初始化如下：

If CREL type = true

then OSCC：0

OSCS：0

CREL state: CONNECTION-NOT-ESTABLISHED

If CREL type = false AND Max-PDU-Receiving-High-Prio=0 AND Max-PDU-Receiving-Low-Prio=0

then OSCC:0

OSCS:0

CREL state: CONNNECTIONLESS-CLIENT

If CREL type = false AND Max-PDU-Sending-High-Prio=0 AND Max-PDU-Sending-Low-Prio=0

then OSCC:0

OSCS:0

CREL state: CONNECTIONLESS-SERVER

4.11.2 操作准备就绪的条件

为成功地终止起动阶段（FMS 基本状态机的状态=FMS READY）必须满足以下条件：

-FMS CRL 的静态部分存在

-有足够充分的用于 FMS CRL 动态部分的资源可供使用

4.11.3 FMS 操作准备就绪

在 FMS READY 状态，FMS 是可操作的，而且可以接受连接建立请求或数据传送请求（广播/群播）。

4.11.4 FMS 基本状态机

4.11.4.1 FMS 基本状态机描述

FMS START （FMS 起动）

检查当前 FMS CRL。当存在一个有效 FMS CRL 时，退出此状态。

FMS DISABLE（FMS 停用）

为了数据传输由 FMA 7 停用 FMS

LOADING CRL（装载 CRL）

在此状态，用 FMS Load CRL 服务装载

CHECK CRL（检查 CRL）

在一个装载顺序后对 FMS CRL 进行检查

FMS READY（FMS 准备就续）

FMS 准备就续数据传输

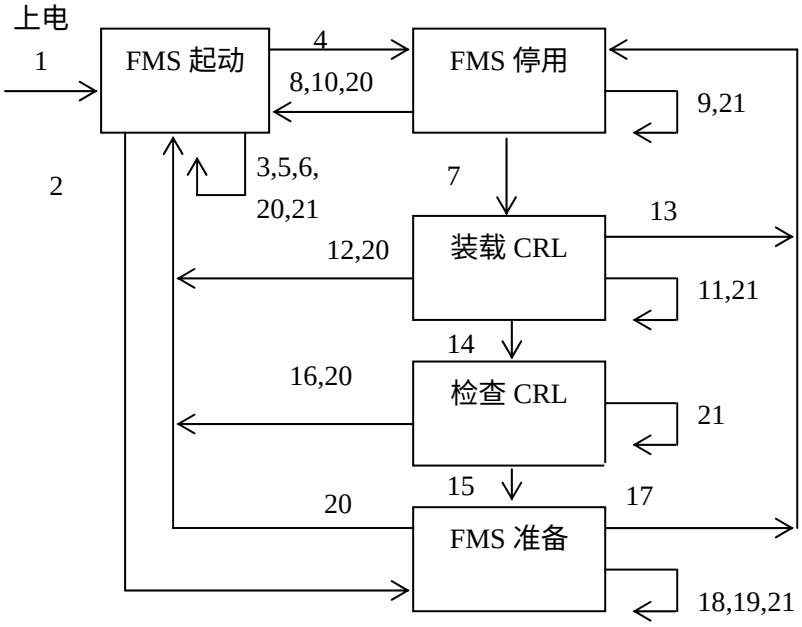


图 79. FMS 基本状态机

4.11.4.2 状态转换

当前状态 事件 \退出条件 ⇒采取的动作	转换	下一状态
上电 ⇒起动 FMS CRL 测试	1	FMS-START
FMS-START FMS CRL 测试完成 \有效 FMS CRL 可供使用 AND（“与”）资源充分 ⇒预置 FMS CRL 的动态部分 起动与 CREF 有关的全部 FMS 状态机	2	FMS-READY
FMS-START \无有效 FMS CRL 可供使用 OR（“或”）资源不充分	3	FMS-START
FMS-START	4	FMS-DISABLE

FMS-Disable .req
⇒FMS-Disable.con（+）

当前状态 事件 \退出条件 ⇒采取的动作	转换	下一状态
FMS-START FMS-Enable. req ⇒FMS-Enable. con(-)	5	FMS-START
FMS-START FMS-LOAD-CRL. Req ⇒FMS-Load-CRL. con(-)	6	FMS-START
FMS-START 来自 FMS 用户的任何服务原语 ⇒Abort. ind (ABT RC 13)	23	FMS-START
FMS-START 来自 LLI 的任何服务原语 ⇒ABT. req（ABT RC 13）	24	FMS-START
FMS-DISABLE FMS-Load-CRL. req \FMS CRL 登入项有效 ⇒FMS-Load-CRL. con（+）	7	LOADING-CRL
FMS-DISABLE FMS_Load_CRL.req \FMS CRC 登入项无效 ⇒FMS-Load-CRL. con（-）	8	FMS-START
FMS-DISABLE FMS-Disable. req ⇒FMS_Disable.con（+）	9	FMS- DISABLE
FMS-DISABLE FMS-Enable.req ⇒FMS-Enable. con（-）	10	FMS- START
FMS-DISABLE 来自 FMS 用户的任何服务原语 ⇒Abort. ind（ABR RC 13）	25	FMS-DISABLE
FMS-DISABLE 来自 LLI 的任何服务原语 ⇒ABT. req（ABT RC 13）	26	FMS-DISABLE
LOADING-CRL FMS-Load-CRL.req \FMS CRL 登入项有效 ⇒FMS-Load-CRL.con（+）	11	LOADING-CRL
LOADING-CRL	12	FMS-START

FMS-Load CRL.req
\FMS CRL 登入项无效
⇒FMS-Load-CRL. con（-）

当前状态 事件 \退出条件 ⇒采取的动作	转换	下一状态
LOADING-CRL FMS-Disable. req ⇒FMS-Disable. con（+）	13	FMS-DISABLE
LOADING-CRL FMS. Enable. req ⇒起动 FMS CRL 测试	14	CHECK-CRL
LOADING-CRL 来自 FMS 用户的任何服务原语 ⇒Abort. ind（ABT RC 13）	27	LOADING-CRL
LOADING-CRL 来自 LLI 的任何服务原语 ⇒ABT.req(ABT RCI3)	28	LOADING-CRL
CHECK-CRL FMS CRL 测试完成 \有效的 FMS CRL 可供使用 “ 与 ” 资源充分 ⇒预置 FMS CRL 动态部分 起动与 CREF 有关的所有 FMS 状态机 FMS-Enable. con（+）	15	FMS-READY
CHECK-CRL FMS CRL 测试完成 \无有效的 FMS CRL 可供使用 “ 或 ” 资源不充分 ⇒FMS-Enable. con（-）	16	FMS-START
CHECK-CRL 来自 FMS 用户的任何服务原语 ⇒Abort. ind（AQT RC 13）	29	CHECK-CRL
CHECK-CRL ⇒ABT. req（ABT RC 13）	30	CHECK-CRL
FMS-READY FMS-Disable. req ⇒解除所有 FMS 连接 ABT. req（ABT RC 13） ABT.ind（ABT RC 13） FMS-Disable. con（+）	17	FMS-DFSABLE
FMS-READY	18	FMS-READY

FMS-Enable. req
⇒FMS-Enable. con（-）

当前状态 事件 \退出条件 ⇒采取的动作	转换	下一状态
FMS-READY FMS-Load-CRL.req ⇒FMS-Load-CRL.con(-)	19	FMS-READY
FMS-READY 接收来自 FMS 用户的任何与 CREF 无关的服务原语 ⇒Abort. ind（ABT RC2）	22	FMS-READY
FMS-READY 接收来自 LLI 的任何与 CREF 无关的服务原语 ⇒ABT.req<ABT-RC5>	31	FMS-READY
（任意状态） FMS-Reset. req ⇒复位所有 CREF 起动 FMS CRL 测试 FMS Reset. con（+）	20	FMS-START
（任意状态） FMS-Ident. req“ 或 ”FMS-Read-CRL. req ⇒执行 FMS 服务 FMS Service. con（+/-）1 个 FMA 7	21	<相同状态>

4.12 结构参数出错类型和类型描述

4.12.1 参数出错类型

包含在结果（-）（Result（-））中的一个出错信息具有以下结构

表 67. 出错类型

出错类型	.res .con
出错类别	M
出错代码	M
附加代码	U
附加描述	U

出错类别

此参数指示出错类别，其编码见 3.16 节句法描述。

出错代码

此参数给出出错的更详细的说明，其编码见句法描述。

附加代码（可选）

此参数可选，用户负责其使用及评估。

附加描述

此参数可选，可用于对出错消息附加一个文本。

4.12.1.1 出错类别和出错代码的含义

出错类别

出错类别的含义

-出错代码

出错代码的含义

VFD 状态

每当 VFD 状态使请求服务不能被执行时，此出错类别被返回。

-其它

此出错代码是由于上列任何一个原因以外而造成的返回

应用引用

此出错类别与在其上执行服务的通信关系有关

-应用不可达到

应用过程不可达到

-其它

此出错代码是由于上列任何一个原因以外而造成的返回

定义

在对象定义（Create Program Invocation, Define Variable List）有问题时，此出错类别返回。

-对象未定义

所要求的对象不存在

-对象属性不一致

所指示的对象以不一致的属性予以定义

-名称已存在

名称已经存在

-其它

此出错代码是由于上列任何一个原因以外而造成的返回

资源

此出错类别在超出可供使用的资源时返回

-存贮区不够使用

没有更多的存贮区供执行此服务之用

-其他

此出错代码是由于上列任何一个原因以外而造成的返回

出错类别

出错类别的含义

-出错代码

出错代码的含义

服务

在服务本身存在问题时此出错类别返回

-对象状态冲突

对象的当前状态不允许执行服务

-对象约束冲突

在此时服务执行是不可能的

-参数不一致

此服务包含了不一致参数

-非法参数

参数有非法值

-PDU-Size

Response-PDU>Max-DPU-Sending-Low-Prio

-其它

此出错代码是由于上列任何一个原因以外的原因而造成的返回

存取

在存取失效时此出错类别返回

-不支持对象存取

对所请求的存取，未曾定义对象

-对象不存在

对象不存在

-对象存取被拒绝

FMS 客户机对对象无足够的存取权

-无效地址

所指示的地址超出合法范围（Phys Read, Phys Write）

-对象已无效

对虽已定义但其引用属性却未定义的对象进行存取，这是一种永久性出错。

-硬件故障

由于硬件有错，对对象进行存取失败

出错类别

出错类别含义

-出错代码

出错代码含义

存取

-类型冲突

因数据类型不正确，存取被拒绝。

-不支持所命名的存取

对带名称的存取不予支持

-对象属性不一致

对象属性不一致

-其它

此出错代码是由于上列任意一个原因以外的原因而造成的返回

OD 出错

每当对 OD 进行了不正确的改变（Put OD, Create Program Invocation, Define Variable List）时此出错类别返回。

-名称长度溢出

超出名称的合法长度

-OD 溢出

超出 OD 的合法长度

-对象描述长度溢出

超出一个单对象描述的合法长度

-OD 有写保护

对象字典有写保护

-扩展长度溢出

超出扩展的合法长度

-运行问题

当前装载的 OD 不正确

-其它

此出错代码是由于上列任何一个原因以外的原因而造成的返回

其它

此出错类别是由于上列任何一个原因之外的原因而造成的返回

-其它

此出错代码是由于上列任何一个原因之外的原因而造成的返回

4.12.1.2 其余参数的含义

附加代码

无进一步说明

附加描述

无进一步说明

4.12.2 参数类型描述

表 68. 类型描述

参 数 名	
类型描述	M
简单	S
数据类型索引	M
长度	M
数组	S
数据类型索引	M
长度	M
元素数目	M
结构	S
元素类型表	M
数据类型索引	M
长度	M

类型说明

此参数包含数据类型描述

简单

此参数包含一个简单变量的数据类型和长度

数组

此参数包含数组的一个元素的数据类型和长度，以及元素数目。

结构

此参数包含记录元素的数据类型描述表

元素类型表

此参数是记录元素的数据类型描述表，给出每个元素的数据类型和长度。

数据类型索引

数据类型的逻辑地址

长度

长度以八位位组为单位。

4.13 表

4.13.1 对象代码表（List of Object Codes）

表 69. 对象代码

对象	对象代码
Null 对象	0
-保留-	1
域	2
程序调用	3
事件	4
数据类型	5
数据类型结构描述	6
简单变量	7
数组	8
记录	9
变量表	10

对象代码应为八位整数类型

4.13.2 标准数据类型表

表 70. 标准数据类型

数据类型	索引	八位位组数
布尔数	1	1
8 位整数	2	1
16 位整数	3	2
32 位整数	4	4
8 位无符号数	5	1
16 位无符号数	6	2
32 位无符号数	7	4
浮点数	8	4
可视字符串	9	1，2，3..
八位位组串	10	1，2，3..
日期	11	7
日时	12	4 或 6
时差	13	4 或 6
位串	14	1,2,3.

4.13.3 对象属性和服务参数表

表 71. 对象属性和服务参数

对象属性/服务参数	数据类型
异常终止细节	八位位组串，最多 16 个八位位组
异常终止标识符	8 位(bit)无符号数
存取组	位串，1 个八位位组
支持存取保护	布尔数
存取权	位串，2 个八位位组
存取说明	--
附加代码	16 位（bit）整数
附加描述	可视字符串
附加信息	可视字符串
所有属性	布尔数
FMS CRL 登入项数	16 位（bit）整数
数组	--
结果	8 位(bit)整数
控制器类型	可视字符串
计数器	8 位(bit)整数
数据	--
数据类型索引	16 位（bit）无符号数
可删除	布尔数
描述	--
在此检测	布尔数
域索引	16 位（bit）无符号数
域名	可视字符串，最多 32 个八位位组
域状态	8 位(bit)无符号数
DP-OD 长度	16 位（bit）无符号数
DV-OD 长度	16 位（bit）无符号数
元素索引	16 位（bit）无符号数
启用	布尔数
出错类别	8 位(bit)整数
出错代码	8 位(bit)整数
出错 CREF	16 位（bit）无符号数
出错类型	--
事件数据	八位位组串
事件名	可视字符串，最多 32 个八位位组
事件数	八位(bit)无符号数
执行变元	八位位组串
扩展	长度+八位位组串
最终结果	布尔数
第一索引 DP-OD	16 位（bit）无符号数
第一索引 DV-OD	16 位（bit）无符号数
第一索引 S-OD	16 位（bit）无符号数
FMS CRL 登入项	八位位组串
FMS CRL 登入项静态	八位位组串
支持 FMS 特征	位串，6 个八位位组
硬件版本	可视字符串
索引	16 位（bit）无符号数
索引事件数据	16 位（bit）无符号数
调用 ID	8 位(bit)整数
CREL 状态	8 位(bit)无符号数
CREL 类型	布尔数
CREF	16 位（bit）无符号数
长度	8 位(bit)无符号数

元素类型表	--
元素索引表	--
域表	--
索引表	--
本地地址表	--
对象描述表	--

表 71（续）

对象属性/服务参数	数据类型
变量表	--
VFD 专用对象表	--
装载数据	八位位组串
本地地址	32 位（bit）无符号数
本地地址 DP-OD	32 位（bit）无符号数
本地地址 DV-OD	32 位（bit）无符号数
本地地址 OD-ODES	32 位（bit）无符号数
本地地址 S-OD	32 位（bit）无符号数
本地细节	位串，3 个八位位组
本地地址 ST-OD	32 位（bit）无符号数
本地产生	布尔数
逻辑状态	8 位（bit）无符号数
最长字节	16 位（bit）无符号数
最多未完成服务客户机	8 位（bit）无符号数
最多未完成服务的服务器	8 位（bit）无符号数
最大 PDU 发送高优先权	8 位（bit）无符号数
最大 PDU 发送低优先权	8 位（bit）无符号数
最大 PDU 接收高优先权	8 位（bit）无符号数
最大 PDU 接收低优先权	8 位（bit）无符号数
型号名	可视字符串
随后还有（More Follows）	布尔数
名长度	8 位（bit）无符号数
域的数目	8 位（bit）无符号数
元素数目	8 位（bit）无符号数
原调用 ID	8 位（bit）整数
未完成服务计数器客户机	8 位（bit）无符号数
未完成服务计数器服务器	8 位（bit）无符号数
OD 对象描述	八位位组串
口令	8 位（bit）无符号数
物理状态	8 位（bit）无符号数
PI 名	可视字符串，最多 32 个八位位组
PI 状态	8 位（bit）无符号数
优先权	布尔数
行规号	八位位组串，2 个八位位组
原因代码	8 位（bit）无符号数
拒绝代码	8 位（bit）整数
拒绝 PDU 类型	8 位（bit）无符号数
可再用	布尔数
修订版	可视字符串
ROM/RAM 标志	布尔数
S-OD 长度	16 位（bit）无符号数
简单	- -
软件版本	可视字符串
ST-OD 长度	16 位（bit）无符号数
起始索引	16 位（bit）无符号数
结构	- -
子索引	8 位（bit）无符号数
符号	可视字符串
符号长度	8 位（bit）无符号数
类型描述	- -
上载状态	8 位（bit）无符号数
变量索引	16 位（bit）无符号数
变量表名	可视字符串，最多 32 个八位位组
变量名	可视字符串，最多 32 个八位位组
供货厂商名	可视字符串
版本 OD	16 位（bit）整数
VFD 索引	32 位(bit)无符号数
支持的 VFD 索引	布尔数

4.13.4 服务表

表 72. 服务含义

服务	含义
启动(Initiate)	建立连接
中止(Abort)	解除连接
拒绝(Reject)	拒绝不适当的服务或 PDU
状态(Status)	读设备/用户状态
未请求状态(Unsolicited status)	报告未请求状态
标识(Identify)	读取供货厂商，型号，版本
获得对象字典(Get OD)	读取对象描述
启动放置对象字典(Initiated put OD)	启动 OD 装载
放置对象字典(Put OD)	装载对象描述
结束放置对象字典(Terminate Put OD)	停止 OD 装载
启动下载序列(Initiate Download Sequence)	启动下载序列
下载段(Download Segment)	传送下载数据块
结束下载序列(Terminate Download Sequence)	停止下载序列
请求域下载(Request Domain Download)	请求下载
启动上载序列(Initiat Upload Sequence)	启动上载序列
上载段(Upload Segment)	传送上载数据块
结束上载序列(Terminate Upload Sequence)	停止上载序列
请求域上载(Request Domain Upload)	请求上载
创建程序调用(Create Program Invocation)	创建程序
删除程序调用(Delete Program Invocation)	删除程序
起动(Start)	在复位后起动程序
停止(Stop)	停止程序
恢复(Resume)	在停止后恢复程序
复位(Reset)	复位程序
削除(Kill)	取消程序
读(Read)	读变量
写(Write)	写变量
带类型的读(Read with Type)	读变量（带类型）
带类型的写(Writ with Type)	写变量（带类型）
物理读(Phys Read)	读存储单元
物理写(Phys Write)	写存储单元
信息报告(Information Report)	发送数据
带类型的信息报告(Information Report with Type)	发送数据（带类型）
定义变量表(Define Variable List)	定义变量表
删除变量表(Delet Variable List)	删除变量表
事件通告(Event Notification)	报告事件
带类型的事件通告(Event Notification with Type)	报告事件（带类型）
应答事件通告(Acknowledge Event Notification)	确认事件
改变事件条件监测(Alter Event Condition Monitoring)	停用/启用事件

4.14 一致性

本节阐述协议实现一致性陈述（Protocol Implementetion Conformance Statements, PICS）。每个设备供应厂商都应完全填写这些表。

PICS 有 4 个部分：

-第 1 部分叙述有关实现和系统的信息

-第 2 部分为所支持的服务表

-第 3 部分为所支持的参数表

-第 4 部分为本地实现值表

4.14.1 实现和系统（PICS 第 1 部分）

表 73. 实现和系统

系统参数	细节
实现的供货厂商名称	
实现的型号名称	
实现的修订版标识符	
FMS 的供货厂商名称	
FMS 的控制器型号称	
FMS 的硬件版本	
FMS 的软件版本	
行规号	
调用的 FMS 用户（键入“ Yes ”或“ No ”）	
被调用的 FMS 用户（键入“ yes ”或“ No ”）	

4.14.2 支持的服务（PICS 的第 2 部分）

每行都应键入支持的服务原语（“ Yes ” 或“ No ”）.如果在一行中不止一个服务原语，则仅在此行列出的所有服务原语均被支持时可键入“ Yes ”。

表 74. 所支持的服务

服务	所支持的原语	
启动	.req,.con	
状态	.req,.con	
标识	.req,.con	
Get OD	.req,.con	
Get OD(长格式)	.req,.con	.ind,.res
未请求状态	.req	.ind
启动 Put OD Put OD 结束 Put OD	.req,.con .req,.con .req,.con	.ind,.res .ind,.res .ind,.res
启动下载序列 下载段 结束下载序列	.req,.con .ind,.res .req,.con	.ind,.res .req,.con .ind,.res
启动上载序列 上载段 结束上载顺序	.req,.con .req,.con .req,.con	.ind,.res .ind,.res .ind,.res
请求域下载	.req,.con	.ind,.res
请求域上载	.req,.con	.ind,.res
创建程序调用 删除程序调用	.req,.con .req,.con	.ind,.res .ind,.res
起动 停止 恢复 复位	.req,.con .req,.con .req,.con .req,.con	.ind,.res .ind,.res .ind,.res .ind,.res
削除	.req,.con	.ind,.red
读	.req,.con	.ind,.red
写	.req,.con	.ind,.red
带类型的读	.req,.con	.ind,.red
带类型的写	.req,.con	.ind,.red
物理读	.req,.con	.ind,.red
物理写	.req,.con	.ind,.red

表 74. 支持的服务（续）

服务	所支持的原语	
信息报告	.req	.ind
带类型的信息报告	.req	.ind
定义变量表 删除变量表	.req,.con .req,.con	.ind,.res .ind.res
事件通告	.req	.ind
带类型的事件通告	.req	.ind
应答事件通告	.req,.con	.ind,.res
改变事件条件监测	.req,.con	.ind,.res

4.14.3 FMS 参数及选项（PICS 第 3 部分）

FMS 实现是否支持该参数或该选项，设备的供应厂商应在每一行都用“ YES ”或“ NO ”来表示。

表 75. FMS 参数和选项

FMS 参数和选项	细节
用名称编址	Yes/no
名称的最长长度	数值
支持存取保护	Yes/no
扩展的最长长度	数值
执行变元的最长长度	数值

4.14.4 本地实现值（PICS 第 4 部分）

表 76. 本地实现值

本地实现值	细节
FMS PDU 最长长度	数值
未完成调用服务的最大数	数值
未完成被调用服务的最大数	数值
执行变元的句法和语义	解释
扩展的句法和语义	解释