1. 目的

本标准的目的是要定义一些数据通信的服务和协议，这些服务和协议用于采暖、通风、空调和制冷控制设备（简称 HVAC&R）的监控计算机设备之中，也可用于其它楼宇自动控制系统的计算机设备之中。此外本标准还要定义一个抽象的和面向对象的方法，用来表示这些设备之间信息通信方式，从而促使在楼宇系统中使用数字控制技术。

2. 范围

2.1 本协议提供了一个完整的报文集合，用来在设备之间传送二进制编码、模拟量、文本数据和数字数据，这些数据是（当然不局限于这些）：

(a) 硬件的二进制输入量和输出量的值，

(b) 硬件的模拟输入量和输出量的值，

(c) 软件的二进制和模拟量的值；

(d) 文本字符串的值；

(e) 时间进度表信息，

(f) 报警和事件信息，

(g) 文件，

(h) 控制逻辑。

2.2 本协议**将每个楼宇自动控制系统计算机建模成为一些数据结构的集合，并且称这些数据结构为对象。**对象的各个属性代表了这个设备硬件、软件以及操作的各个方面。这些对象提供了一种不需了解设备的内部设计或配置细节，而能够识别和访问信息的方法。

3. 术语定义

3.1 所采用的国际标准中的术语

本标准中所使用的以下术语由其它国际标准或开放系统互联的草案标准所定义。在此重新列出这些定义以及相应参考的标准。在第 25 节参考文献中，包括了本节和本标准中其它地方所涉及的所有美国标准和国际标准的名称。斜体书写的词或短语在本节中都有专门的定义条目。

3.1.1 抽象句法结构（abstract syntax）：用来表示应用层数据或*应用层协议控制信息*的符号规则的规定，这些符号规则独立于用来表示它们的编码技术。（ISO 8822）

3.1.2 应用（application）：用户的处理请求信息的集合。（ISO 8649）

3.1.3 应用实体（application-entity）：与 OSI 相关的一个*应用进程*的表示形式。（ISO 7498）

3.1.4 应用进程（application-process）：在一个*实开放系统*中的一个活动元素，其功能是为一个特定*应用*而执行信息处理的任务。（ISO 7498）

3.1.5 应用层协议控制信息（application-protocol-control-information）：在*应用实体*之间交换的信息，这些信息通过使用表示层服务来协调它们之间相连接的操作。（ISO 9545）

3.1.6 应用层协议数据单元（application-protocol-data-unit）：应用协议中规定的数据单元，其中包含有*应用层协议控制信息*和应用层用户数据（application-user-data）。（ISO 9545）

3.1.7 应用层服务元素（application-service-element）：是一个*应用实体*的一部分，其功能是在适当的时候，通过使用下层服务来提供一个 OSI 环境的能力。（ISO 7498）

3.1.8 具体句法结构（concrete syntax）：用于数据的正式规范中的规则的表示形式，具体表示某数据的规范表示形式。（ISO 7498）

3.1.9 对等实体（peer-entities）：在相同的协议层次中的实体。（ISO 7498）

3.1.10 实开放系统（real open system）：是一个*实系统*，在与其它的*实系统*通信时遵守 OSI 标准的要求。（ISO 7498）

3.1.11 实系统（real system）：是一个集合体，其组成部分包括有计算机、相关的软件、外部设备、终端、操作者、操作处理过程、信息传递方法等等，这些部分组成一个自主的整体，完成信息处理和/或信息传递的功能。（ISO 7498）

3.1.12 第 N 层服务访问点（(N)-service-access-point）：协议中第 N 层*实体*向第 N+1 层*实体*所提供的服务的访问点。（ISO 7498）

3.1.13 第 N 层服务数据单元（(N)-service-data-unit）：协议中第 N 层的接口数据，其一致性不受第 N 层通信的影响。（ISO 7498）

3.1.14 服务用户（service-user）：是在某个开放系统中的*实体*，该实体通过*服务访问点*使用一个服务。（ISO TR 8509）

3.1.15 服务提供者（service-provider）：向对等*服务用户*提供服务的所有*实体*的抽象总和。（ISO TR 8509）

3.1.16 传输句法结构（transfer-syntax）：在开放系统之间用于数据传递目的的*具体句法结构*。（ISO 7498）

3.1.17 服务原语；原语（service-primitive；primitive）：用来描述*服务用户*和*服务提供者*之间交互作用的、抽象的、与具体实现无关的表示方式。（ISO TR 8509）

3.1.18 请求（原语）(request (primitive))：一种交互作用的表示方式，用来描述*服务用户*希望得到完成某些操作的服务而调用一些过程。(IISO TR 8509)

3.1.19 指示（原语）(indication (primitive))：一种交互作用的表示方式，在该作用中，*服务提供者*或者指示本身已经主动调用了某些过程，或者指示在对等*服务访问点*的*服务用户*已经调用了一个过程。(ISO TR 8509)

3.1.20 响应（原语）(response (primitive))：一种交互作用的表示方式，在该作用中，*服务用户*表明已经完成了先前一个交互作用所调用的一些过程，其中所指的先前的一个交互作用是由一个*指示*原语所表示的。(ISO TR 8509)

3.1.21 证实（原语）(confirm (primitive))：一种交互作用的表示方式，在该作用中，一个*服务提供者*在一个特定的*服务访问点*指示某些过程已经完成，这些过程是在相同的*服务访问点*由前一个*请求*原语所调用的。(ISO TR 8509)

3.1.22 用户元素 (user element)：是一个*应用进程*的某部分的表示，这个应用进程的部分因为需要实现该*应用进程*的通信目的而要使用*应用服务元素*。(ISO 7498)

3.2 本标准中定义的术语

3.2.1 访问控制 (access control)：一种控制访问*网络资源*的方法。

3.2.2 报警 (alarm)：1.一种听觉的或者视觉的或者两者都有的通报，提醒操作者出现了可能需要采取纠错操作的非正常现象。2.由特定设备或控制器检测出的非正常状态，这种设备或控制器是为监测这种状态而专门设计的。

3.2.3 算法改变报告 (algorithmic change reporting)：利用在一个**事件登记**对象中指定的算法，得出的关于一个报警或者一个事件的检测和报告。参见 3.2.27 的*内部报告*。

3.2.4 BACnet 设备 (BACnet device)：是指任何一种支持用 BACnet 协议进行数字通信的真实的或者虚拟的设备。

3.2.5 BACnet 用户 (BACnet-user)：是一个*应用进程*中由 BACnet *用户元素*所表示的那个部分。

3.2.6 网桥 (bridge)：在物理层和数据链路层上连接两个或多个*网段*的设备。该设备还可以根据 MAC 层地址完成报文的过滤。

3.2.7 广播 (broadcast)：表示一个报文，这个报文作为一个单一单元而发送到多个设备上。

3.2.8 状态改变 (change of state)：表示一个事件，这个事件在测量出或者计算出布尔值或离散量值发生改变时发生。

3.2.9 数值改变 (change of value)：表示一个事件，这个事件在测量出或者计算出模拟值与预定义的值发生改变时发生。

3.2.10 客户（client）：表示一个系统或者设备，它为某些特定目的，通过一个服务请求实例调用其它设备的服务，客户向*服务器*请求服务。

3.2.11 上下文范围（context）：能够完全描述某一个特定时间的、特定通信环境的数据或信息的集合。

3.2.12 控制器（controller）：用于调节或管理一个系统或部件的设备。

3.2.13 数据保密性（data confidentiality）：描述信息不被未经授权的个体、实体或过程所利用或识别的属性。

3.2.14 数据完整性（data integrity）：描述数据不被未经授权的方法所改变或破坏的属性。

3.2.15 数据来源鉴别（data origin authentication）：对接收到的数据的来源是否为其声明的来源的证实行为。

3.2.16 直接连接网络（directly connected network）：通过一个路由器即可访问的、报文无须中间路由器中继的网络。一个 PTP 连接，当 PTP 连接有效时，并且没有使用中间路由器时，就是一个直接连接网络。

3.2.17 下载（download）：一种特定的文件传输行为，专门用于将可执行程序或数据库传送到远程可执行设备上。

3.2.18 实体（entity）：具有独立的、可区分特征的存在事物，并且可以用一组属性来描述它的具有可标识的性质。

3.2.19 检错（error detection）：用来识别在通信中是否存在错误的一个过程。

3.2.20 纠错（error recovery）：由被检测到的错误所激活的过程，该过程将确保信息交换得以继续。

3.2.21 网关（gateway）：用来连接两个或多个不同*网络*的设备，信息可以通过这个设备在这些*网络*之间交换。

3.2.22 全局（global）：关于一个通信*互联网络*上的所有设备或节点。

3.2.23 全局广播（global broadcast）：表示一个报文，这个报文被发送到一个 BACnet *互联网*上的所有*网络*中的设备或 *节点*。

3.2.24 半路由器（half router）：是一个设备或者 *节点*，用在一个 PTP 连接的一端。两个半路由器形成一个*路由器*，并建立起一个有效的 PTP 连接。

3.2.25 初始化（initialization）：表示建立一个已知状态的过程，通常这个过程是从加电开始的。初始化可能需要重新建立一个节点的逻辑或物理地址。

3.2.26 互联网络（internetwork）：由*路由器*互联起来的两个或多个*网络*的集合。在一个 BACnet 的*互联网络*中，任意两个节点之间存在着唯一一条报文传输路径。

3.2.27 内部报告（intrinsic reporting）：表示对一个报警或者事件的探测和报告，其探测基于在*对象类型*规范中定义的算法，不需要调用对**事件登记**对象的外部引用。参见 3.2.3 的*算法改变报告*。

3.2.28 密钥（key）：控制加密和解密操作的一个符号序列。

3.2.29 本地（local）：同一*网络*上作为参考设备的相关设备。

3.2.30 本地广播（local broadcast）：表示一个报文，这个报文被发送到同一*网络*上所有设备或*节点*。

3.2.31 传输介质（medium）：是物理传输*实体*。典型介质有双绞线、光缆，和同轴电缆。

3.2.32 介质访问控制（medium access control）：表示一个处理过程，这个过用于维持对通讯*传输介质*的访问顺序

3.2.33 网络（network）：由*网桥*互联起来具有相同网络地址的一个或多个*网段*的集合。

3.2.34 网络资源（network resource）：通过通信*传输介质*可以访问的任何物理的或逻辑的*实体*。

3.2.35 节点（node）：与通信*传输介质*相连的可寻址的设备。

3.2.36 对象类型（object type）：由*属性集合*所定义的数据结构。

3.2.37 操作者身份鉴别（operator authentication）：确认登录到一个设备上的操作员是否为其声明的身份的行为。

3.2.38 对等实体身份鉴别（peer entity authentication）：确认在一个联合体中某个同等实体是否为其声明的身份的行为。

3.2.39 物理网段（physical segment）：BACnet *节点*直接相连的单一连续*传输介质*。

3.2.40 可打印字符（printable character）：表示可打印的符号的字符。包括大小写字母、标点符号和算术符号。当然不只局限于此，其精确的集合随所使用的字符集而改变。在 ANSI X.34 标准中可打印字符由十六进制表示，其范围从 X ‘20’ 到 X ‘7E’。

3.2.41 属性（property）：一个*对象类型*的详细特征。

3.2.42 私有的（proprietary）：在 BACnet 的范围内，本标准确定的*对象类型*、*属性*、私有传输服务或枚举的任何扩充或附加部分。

3.2.43 接收方 BACnet 用户（receiving BACnet-user）：接收*指示*或*证实*的 BACnet 用户。

3.2.44 远程（remote）：与当前讨论的设备在不同*网络*的设备或*节点*。

3.2.45 远程广播（remote broadcast）：表示一个报文，这个报文被发送到与报文源不同*网络*上的所有设备或*节点*。

3.2.46 中继器（repeater）：在物理层上连接两个或多个*物理网段*的设备。

3.2.47 请求方 BACnet 用户（requesting BACnet-user）：在有证实的服务中作为*客户*的 BACnet 用户。

3.2.48 响应方 BACnet 用户（responding BACnet-user）：在有证实的服务中作为*服务器*的 BACnet 用户。

3.2.49 路由器（router）：在网络层连接两个或多个*网络*的设备。

3.2.50 安全（security）：用来确保信息交换不泄露给未被授权的个体的所有过程。安全措施包括防止机密信息的泄露，甚至包括设置谁有权访问通讯*网络*。虽然某些安全措施可以通过限制通讯介质的物理访问来实现，安全的定义是不同于访问控制的。

3.2.51 网段（segment）：由中继器互联起来的一个或多个*物理网段*。

3.2.52 发送方 BACnet 用户（sending BACnet-user）：*发送请求*或*响应*的 BACnet 用户。

3.2.53 服务器（server）：为某个目的响应一个服务请求实例的设备或系统。它为*客户*提供服务。

3.2.54 同步（synchronization）：表示一种功能，这种功能在信息传输或交换时，允许处理过程定义和识别某个特定位置，以此来将一次通信对话复位到预设状态。

3.2.55 单元时间（unit_time）：传送一个带有一个比特的发送位和一个比特停止位的八位字节所需的时间，即十个比特时间。

3.2.56 上载（upload）：从一个远程设备上传送一个可执行程序或数据库的过程。通过这种方式，允许随后的下载操作。

3.3 本标准中使用的缩写和简称

A	应用层（前缀）
AE	应用实体
ANSI	美国国家标准学院
APCI	应用层协议控制信息
API	应用编程接口
ARCNET	ARCNET 计算机网络
ASE	应用层服务单元
ASN.1	抽象句法结构符号 1（ISO 8824）
B’’	表示在单引号之间使用二进制符号
BAC	楼宇自动控制
C	有条件的
C（=）	条件（参数与服务原语中左边的参数在语义上相等）

CNF	证实原语
COV	值的改变
CRC	循环冗余码校验
D’’	表示在单引号之间使用十进制符号
DA	本地目标 MAC 层地址
DADR	最终目标 MAC 层地址
DER	需要应答的数据
DES	数据加密标准（FIPS 46-1）
DID	ARCNET 目标 MAC 地址
DLEN	最终目标 MAC 层地址的一个字节长度
DNET	两个字节表示的最终目标的网络号码
DSAP	LLC 目标服务访问点（对于 BACnet 为 X’82’）
EXEC	执行一个服务请求的能力
ICI	接口控制信息
IL	ARCNET 信息长度域
IND	指示原语
IEEE	电气和电子工程师协会
INIT	发起一个服务请求的能力
ISO	国际标准化组织
L	数据链路层（前缀）
LAN	局域网
LLC	逻辑链路控制（ISO 8802-2）
LPCI	链路层协议控制信息
LPDU	链路层协议数据单元
LSAP	链路层服务访问点（对于 BACnet 为 X’82’）
LSDU	链路层服务数据单元
M	命令的
M（=）	命令的（参数与服务原语中左边的参数在语义上相等）
MA	介质访问（前缀）
MAC	介质访问控制
MPCI	MAC 层协议控制信息

MPDU	MAC 层协议数据单元
MSDU	MAC 层服务数据单元
MS/TP	主从/令牌传递
N	网络层（前缀）
NP	网络优先权
NPCI	网络层协议控制信息
NPDU	网络层协议数据单元
NRZ	反向不归零制
NSAP	网络层服务访问点
NSDU	网络层服务数据单元
O	表明对某个属性的支持是可选的
OSI	开放系统互联
P	物理层（前缀）
PAC	ARCNET 数据分组的头字节
PCI	协议控制信息
PDU	协议数据单元
PICS	协议实现一致性声明
PK	私有密钥
PPCI	物理层协议控制信息
PPDU	物理层协议数据单元
PSDU	物理层服务数据单元
PTP	点到点
R	表明对某个属性是支持的，并且用 BACnet 设备可读该属性
REQ	请求原语
RSP	响应原语
S	选择
S（=）	选择（参数与服务原语中左边的参数在语义上相等）
SA	本地网络资源的 MAC 层地址
SAP	服务访问点
SC	ARCNET 系统代码（对于 BACnet 为 X’CD’）
SDU	服务数据单元

SID	ARCNET 源的 MAC 层地址
SK	会话密钥
SLEN	源节点的 MAC 层地址的一个字节长度
SNET	两个字节表示的源节点网络号码
SPC	标准工程委员会
SSAP	LLC 源节点服务访问点（对于 BACnet 为 X'82'）
TSM	事务处理状态机
U	用户选项
U（=）	用户选项（参数与服务原语中左边的参数在语义上相等）
UART	通用的异步收发器
VT	虚拟终端
W	表明对某个属性是支持的，并且用 BACnet 设备可读和可写该属性
X ’ ’	表示在单引号之间使用十六进制符号
XID	交换标识（ISO 8802-2）

4. BACnet 协议的体系结构

国际标准化组织在制定计算机网络通讯协议标准时定义了一个模型，称为开放系统互联参考模型（OSI）（ISO 7498）。模型的目的是解决计算机与计算机之间普遍的通信问题。在这个模型中，将计算机通信这样一个复杂的问题分解成 7 个小的、容易解决的子问题，每个子问题只与某些通信功能相关，并且把这些子问题称为协议体协结构的一层，整个模型是一个七层的体系结构。在图 4-1 中给出这七层的体系结构图。

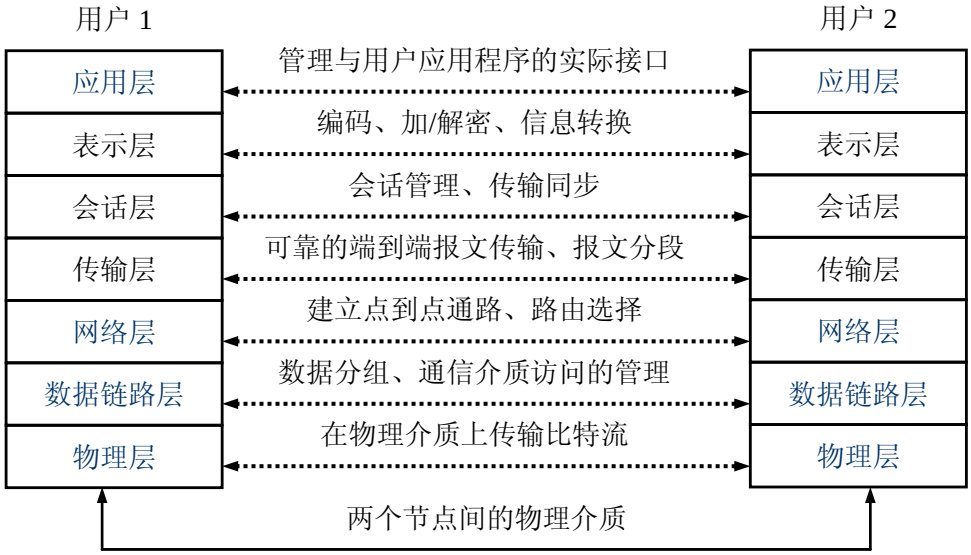


图 4-1. 开放系统互联基本参考模型

对于某个层次来说，它使用下面各层所提供的服务，同时也向它上面的各层提供服务。每一层可以想象成一个黑盒，黑盒的上面和下面都具有经过认真定义的接口。一个应用程序通过与 OSI 应用层的相连，实现与另一个远程应用程序的通信。**对于这种发生在两个应用程序之间的通信，看起来两个程序似乎是通过各自的应用层接口直接相连。**各层之间仅仅需要了解其它层的很少的情况。通过类似的方式，协议的每一层利用下面各层的服务来提供通信服务，与另一个系统的同等层**建立起一个虚的对等层通信**。**而真正的通信只发生在物理层。**

OSI 模型以高度概括的观点来分析计算机与计算机的通信，用来解决在全世界范围内的、巨大而复杂的计算机网络的通信问题。在这种情况下，互相通信的单个计算机之间可能相距很远，因此报文要通过一系列中间点才能到达。而这些中间点相应地可能需要实现路由选择功能、某种解析功能，以及复杂的同步和差错恢复功能。

实现 OSI 模型协议所需的费用较高，在绝大部分楼宇自动控制系统中，并不需要实现 OSI 模型的所有内容。不过只从 OSI 的功能性方面来考虑，经过简化，OSI 模型仍然是楼宇自动控制协议的一个很好的参考。如果只选择 OSI 模型中需要的层次，形成一个简化的模型，作为楼宇自动控制系统的协议体系结构，就可以减少报文的长度，降低通信处理的开销，并

且也满足楼宇自动控制系统的需要。这个简化的体系结构降低了楼宇自动控制工业的生产成本，同时处理器的大批量生产、局域网技术的发展，也为过程控制和办公自动化工业的发展起到了推动作用。另一方面，可以充分利用现有的、易用的、应用广泛的局域网技术，如以太网、ARCNET 和 LonTalk。这样不但可以降低成本，而且也有利于提高性能，为系统集成开辟新的途径。

4.1 BACnet 简化的体系结构

BACnet 建立在包含四个层次的简化分层体系结构上，这四层相当于 OSI 模型中的物理层、数据链路层、网络层和应用层，如图 4-2 所示。BACnet 标准定义了自己的应用层和简单的网络层，对于其数据链路层和物理层，提供了以下五种选择方案。

<table>
<tr><th colspan="4">BACnet 的协议层次</th><th>对应的
OSI 层次</th></tr>
<tr><td colspan="4">BACnet 应用层</td><td>应用层</td></tr>
<tr><td colspan="4">BACnet 网络层</td><td>网络层</td></tr>
<tr><td colspan="2">ISO 8802-2
(IEEE 802.2)类型 1</td><td>MS/TP
(主从/令牌传递)</td><td>PTP
(点到点协议)</td><td rowspan="2">数据
链路层</td></tr>
<tr><td>ISO 8802-3
(IEEE 802.3)</td><td>ARCNET</td><td>EIA-485
(RS485)</td><td>EIA-232
(RS232)</td><td>LonTalk</td></tr>
<tr><td colspan="4">物理层</td></tr>
</table>

图 4-2.BACnet 简化的体系结构层次图

第一种选择是 ISO 8802-2 类型 1 定义的逻辑链路控制（LLC）协议，加上 ISO 8802-3 介质访问控制(MAC)协议和物理层协议。ISO 8802-2 类型 1 提供了无连接(Connectionless)不确认（Unacknowledged）的服务，ISO 8802-3 则是著名的以太网协议的国际标准。

第二种选择是 ISO 8802-2 类型 1 定义的逻辑链路控制协议，加上 ARCNET（ATA/ANSI 878.1)。

第三种选择是主从/令牌传递（MS/TP）协议加上 EIA-485 协议。MS/TP 协议是专门针对楼宇自动控制设备设计的，同 ISO 8802-2 类型 1 一样，它通过控制 EIA-485 的物理层，向网络层提供接口。

第四种选择是点对点（PTP）协议加上 EIA-232 协议，为拨号串行异步通信提供了通信机制。

第五种选择是 LonTalk 协议。

这些选择都支持主/从 MAC、确定性令牌传递 MAC、高速争用 MAC 以及拨号访问。**拓扑结构上，支持星型和总线型拓扑**。物理介质上，支持双绞线、同轴电缆、光缆。这五种选项将在第 7 节到第 11 节中深入论述。

简化的四层 BACnet 体系结构，是在仔细考虑了 BACnet 网络的独特特征和要求，以及尽可能少的协议开销原则后得出的。在下面的分析里，将讨论 BACnet 体系结构只包括物理层、数据链路层、网络层和应用层的原因。

BACnet 完成其固有的操作到底需要哪些层次呢？仔细分析 BACnet 网络的特征后，可以得到以下两点：

首先，BACnet 是一种局域网。即使在某些应用中，楼宇里设备间的远距离通信必不可少时，BACnet 仍然是一种局域网。因为这种远距离的通信功能，由电信网来实现。通信中要完成的路由、中继、可靠的传输等问题都由电信网来处理，电信网可看成是 BACnet 外部的部分。

其次，BACnet 设备是静态的（static），即在空间上，它们不会经常被移来移去。在要完成的功能上，从某种意义上说也是不变的，即不会今天生产的设备的功能是这样，明天就完全不同了。

在充分了解 BACnet 网络的特征后，就可讨论 OSI 模型的各层在 BACnet 网络中的适用性了。

物理层提供了连接设备和传输数据载波信号的方式，显然在 BACnet 协议中，物理层是必不可少的。

数据链路层负责将数据组织成帧（frame）或分组（packet）、管理通讯介质的访问、寻址，以及完成一些差错校正和流量控制的任务，这些都是 BACnet 协议所需要的，因此数据链路层也是必不可少的。

网络层的功能包括：将全局地址解析为局部地址、在一个或多个网络中进行报文的路由、协调不同类型网络的差异（如不同网络所允许的最大报文长度）、序列控制、流量控制、差错控制、以及多路复用。由于 BACnet 网络的拓扑特点，在各个设备之间只存在一条逻辑通路（参见图 4-3），这样便不需要最优路由的算法。其次，BACnet 网络是由中继器或者网桥互联起来的一个或者多个网段所组成的网络，它具有单一的局部地址空间。在这样一种单一网络中，许多 OSI 网络层的功能也变得多余，或者与数据链路层相重复。但是在 BACnet 网络系统中，网络层又是必不可少的。例如，在一个 BACnet 的互联网[注]（internet）中，当两个或者多个网络使用了不同的 MAC 层时，便需要区别局部地址和全局地址，这样才能将报文路由到正确的网络上去。在 BACnet 协议中，通过定义了一个包含必要的寻径和控制信息的网络层头部，来完成这种简化了的网络层功能。

[注]：internet 和 internetwork 这两个词都代表一般的网络互联所形成的网络，在本书中翻译为“互联网”。Internet 这个词代表特定的已经存在于世界范围的计算机互联网，在本书中翻译为“因特网”。——译者注

传输层主要是负责提供可靠的端到端的报文传输、报文分段、序列控制、流量控制，以及差错校正。传输层的许多功能与数据链路层相似，只是在作用范围上有所不同。**传输层提供的是端到端的服务，而数据链路层则提供的是单一网络上点到点的服务**。由于 BACnet 支持多种网络的配置，因此协议必须提供传输层端到端的服务。而可靠的端到端传输和差错校正功能，**在 BACnet 协议中由 BACnet 的应用层利用报文超时重传方式来完成。**其次，考虑到缓冲区和处理器资源的管理，报文分段和端到端的流量控制也是必要的。这是因为即使一个简单的 BACnet 请求，都可能会导致大量的信息回传。同样，这些功能也是由 BACnet 的应用层完成的。最后，为了实现报文的正确重组，序列控制也是必需的。这也是由 BACnet 的应用层中的分段过程实现的。总的来说，由于 BACnet 是建立在无连接的通信模型基础上的，因此所需的服务大大减少，并且可以被高层来实现，这样便省去了一个单独传输层所会增加的通信开销。在 BACnet 协议中，不设单独的传输层，所需的功能由应用层实现。

会话层的功能是在通信双方之间建立和管理长时间对话，其中的一个重要功能是建立同步标志点，用来在发生差错时回复到前一个标志点，以避免对话重新开始。但是在一个 BACnet 网络中，绝大部分的通信都是很简短的，比如读写一个值或者一些值，通知（notify）某个设备有某个警报或事件（event）发生，或者更改某个设定值。当然长时间的信息交换偶然也会发生的，比如上载或下载某个设备参数。由于绝大部分事务处理都是简短的，会话层的服务极少用到，再考虑到设置单独的会话层所带来的开销，所以 BACnet 标准中不包括单独的会话层。

表示层为通信双方提供了屏蔽下层传送语法的服务。这种传送语法是用来将应用层中抽象的用户数据视图，变成适合下层传输的字节序列。但当只存在一种传送语法时，表示层的功能便减少到对应用程序的数据进行编码。由于在 BACnet 应用层中定义了一个固定的编码方案，因此一个独立的表示层也变得不再需要。

协议的应用层为应用程序提供了完成各自功能所需的通信服务。在此基础上，应用程序可以监控 HVAC&R 和其它楼宇自动控制系统。显然应用层是本协议所必需的。

从以上讨论中，可以得到以下几点：

(a) 实现一个完全的 OSI 七层体系结构需要大量的资源和开销，因此它对于目前的楼宇自动控制系统是不适用的。

(b) 按照 OSI 模型的方式构造协议体系结构，并且采用现有的计算机网络技术，可以使得新协议具有实现成本低和便于与其它计算机网络系统集成的特点。

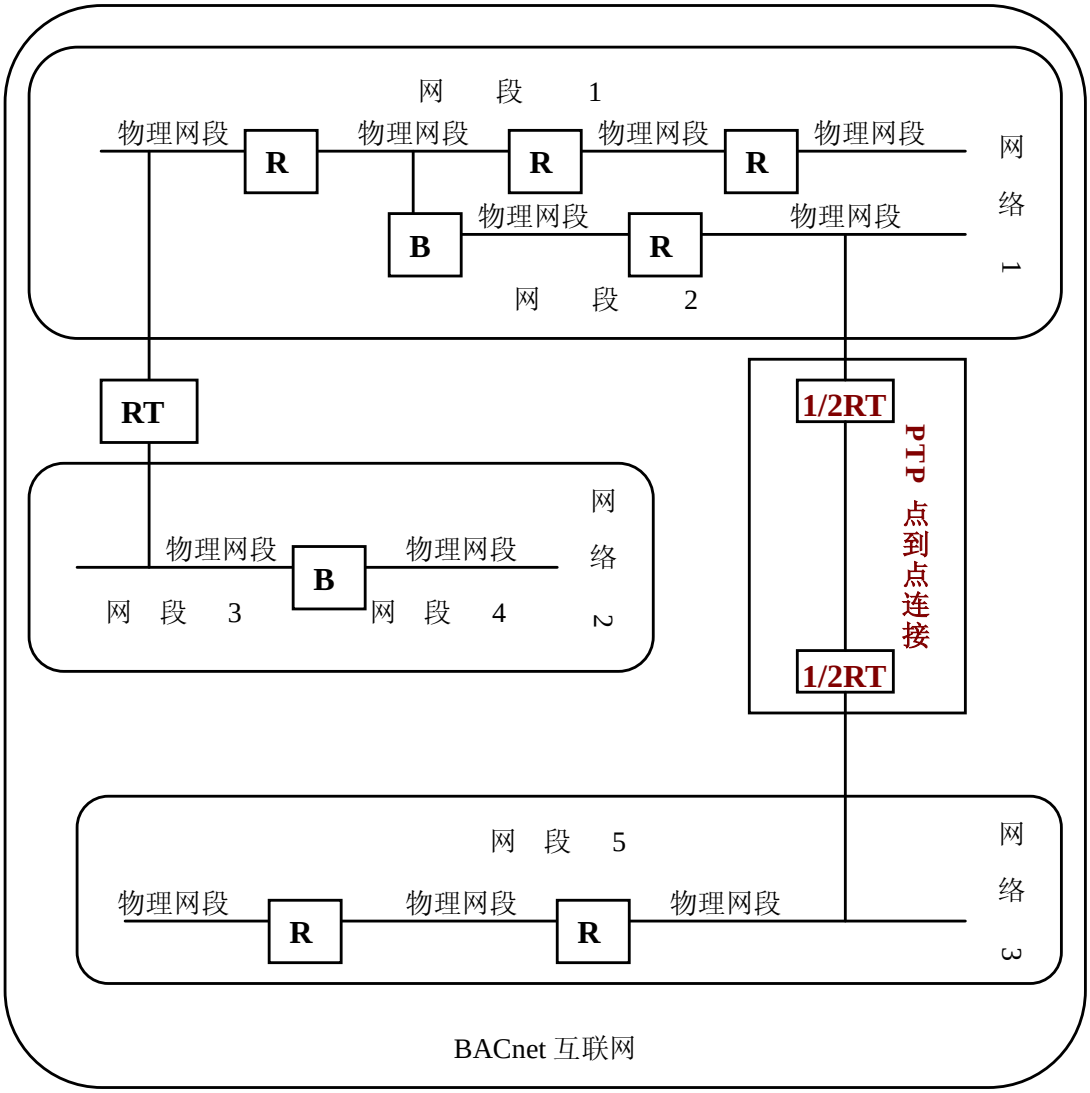
(c) 根据楼宇自动控制系统的环境及要求，可以通过去除 OSI 某些层的功能简化 OSI 模型，来制定新协议的体系结构。

(d) 由物理层、数据链路层、网络层和应用层组成的一个简化体系结构，是当今楼宇自动控制系统的最佳解决方案。

4.2 BACnet 网络的拓扑结构

为了适应各种应用，BACnet 并没有规定严格的网络拓扑结构。BACnet 设备可以直接连接到四种局域网（LANs）中的一种网络上，也可以通过专线或拨号异步串行线连接起来。这几种局域网由可以通过 BACnet 路由器进一步互联（参见第 6 节的详细描述）。

按照局域网拓扑的观点，每个 BACnet 设备与物理介质相连，物理介质称之为**物理网段**。一个或多个物理网段通过**中继器**在物理层连接，便形成了一个 BACnet **网段**。而一个 BACnet **网络**则是由一个或多个 BACnet 网段通过**网桥**互连而成。每个 BACnet 网络都形成一个单一的**介质访问控制 MAC** 地址域，这些在物理层和数据链路层上连接各个网段的设备，可以利用 MAC 地址实现报文的过滤。将使用不同 LAN 技术的多个网络，用 BACnet **路由器**互连起来，便形成了一个 BACnet **互联网（internetwork）**。**如前所述，在一个 BACnet 互联网中，任意两个节点之间恰好存在着一条报文通路。这些概念如图 4-3 所示。**



B = 网桥 RT = 路由器
R = 中继器 1/2RT = 半路由器

图 4-3.BACnet 互联网结构图

4.3 安全

BACnet 系统安全方面的主要隐患，是有人有意或无意地改变了设备的配置参数或控制参数。问题产生的原因经常是由于安全措施没有把某个计算机操作包括在内，从而使得在那台计算机上可以进行非法操作。采取安全措施的一个重要地方是人-机接口处。由于人-机接口不属于通信协议，因此厂家可根据需要自由地在该接口处设置密码保护、跟踪记录或者其它控制措施。另外，在本标准中对任何属性的写访问操作，并没有明确要求是“可写的（writable)”。这一点可以通过限制只能在虚终端节点（virtual terminal node）处才允许修改、或干脆完全禁止来加以改进。这样厂家便可以利用他们认为尽可能合适的、完备的机制，来保护密钥属性（key property)。最后，BACnet 定义了用来提供对等实体、数据来源以及操作员身份鉴别的服务（参见第 24 节)。

5. 应用层

5.1 应用层模型

本节给出 BACnet 应用层的模型，建立这个模型的目的是为了清楚地描述应用层与应用程序之间的交互（interaction)、应用层与协议栈中下面各层次的关系、以及应用层与远程设备中应用层的对等交互。但需要说明的是，该模型并不是应用层的实现规范。

一个**应用进程（Application Process）**是指在一个系统中，针对某个应用而进行信息处理的功能模块。如图 5-1 所示，在**应用进程**中有一部分位于应用层之外，它们与通信功能无关，这些部分都不属于 BACnet 标准的规范范围。我们将**应用进程**中位于应用层内的部分称为**应用实体（Application Entity)**。换句话说，一个**应用实体**是**应用进程**中与 BACnet 通信功能相关的部分。一个应用程序（application program）通过**应用编程接口 API（Application Program Interface）**与**应用实体**进行交互。编程接口不在 BACnet 中定义，但是在具体的实现中它总是一个函数、过程或子程序的调用。在图 5-1 中，阴影部分是应用进程位于 BACnet 应用层中的部分。

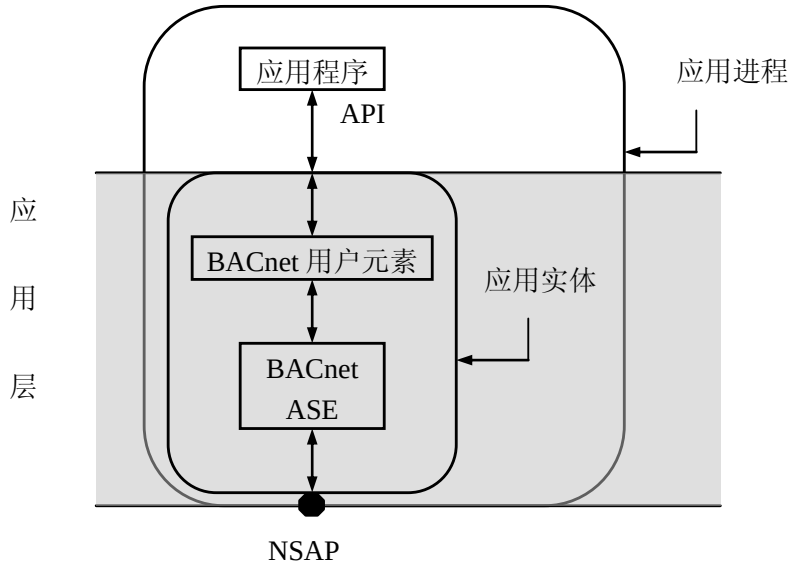


图 5-1. BACnet 应用进程模型

一个**应用实体**由两部分组成，分别是 **BACnet 用户元素（User Element）**和 **BACnet 应用层服务元素 ASE（Application Service Element）**。**BACnet 应用层服务元素**描述了应用服务或功能的集合，在第 13 节到第 17 节以及第 24 节中将进行详细规范。**BACnet 用户元素**执行多种功能，并且支持本地的 API，它描述每个应用服务中“服务过程（service procedure)”的实现。**BACnet 用户元素**负责三个方面的事务，第一，保存事务处理的上下文信息，包括产生请求标识符（ID)，记录哪个标识符是与哪个设备发出的应用服务响应（或对哪个设备的应用服务请求）相对应的；第二，保存超时重传机制所需的超时计时器；第三，将一个设备的行为映射成为 BACnet 对象。

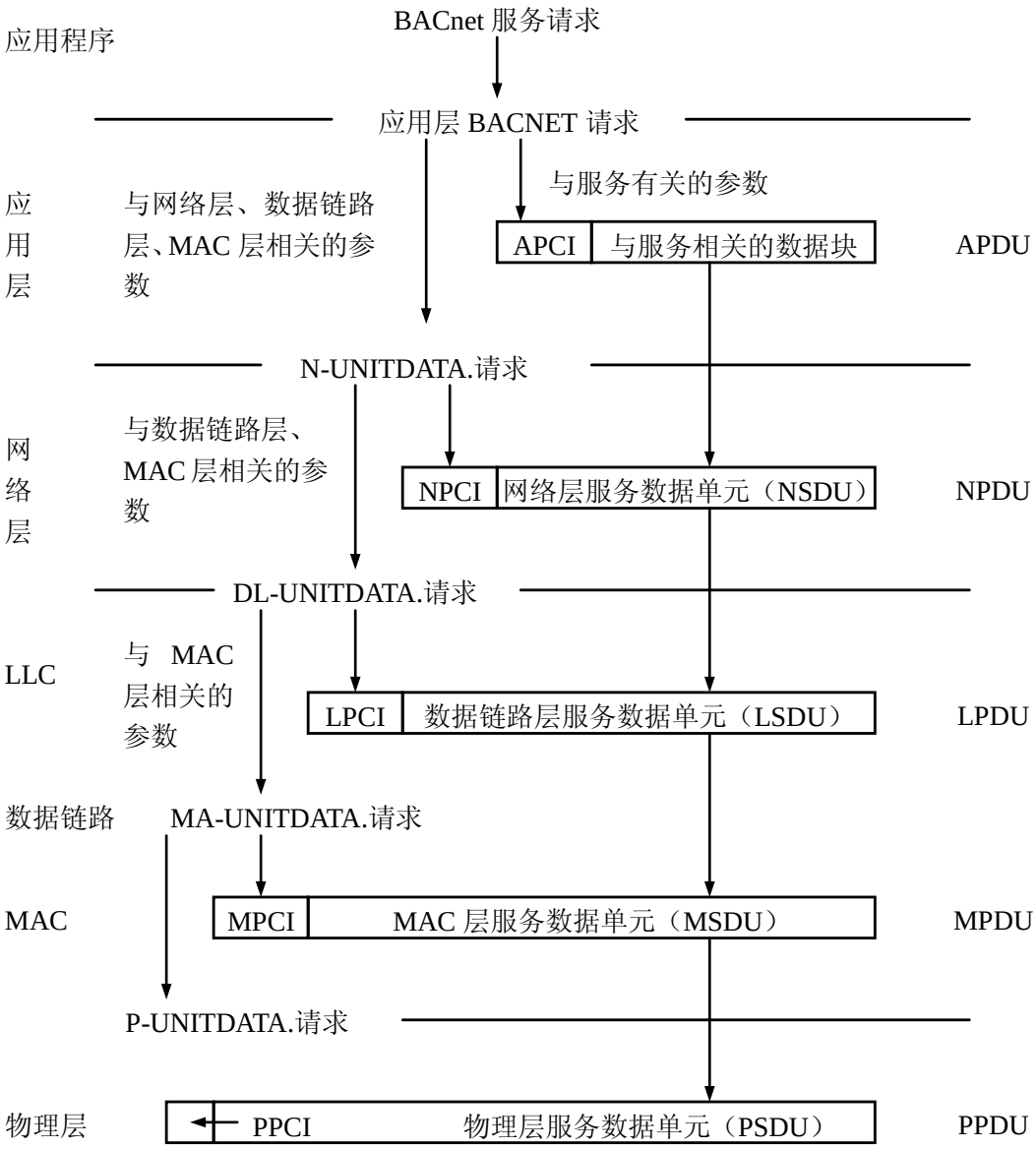
根据 OSI 技术报告中关于 ISO 服务的约定用法（ISO TR 8509)，BACnet 中两个对等应用进程间的信息交换，被表示成抽象服务原语的交换。这些服务原语用来传递一些特定的服务参数，这些参数将在第 13 节到第 17 节以及第 24 节中具体定义。本协议定义四种服务原语：请求（request)、指示（indication)、响应（response）和证实（confirm)。包含在这些原语中的信息，由本标准中定义的各种协议数据单元（PDU：Protocol Data Unit）传递。为了清楚地表示所使用的 BACnet PDU，下面给出一些符号标记：

CONF_SERV.request	CONF_SERV.indication	CONF_SERV.response	CONF_SERV.confirm
UNCONF_SERV.request	UNCONF_SERV.indication		
SEGMENT_ACK.request	SEGMENT_ACK.indication		
ERROR.request	ERROR.indication		
REJECT.request	REJECT.indication		
ABORT.request	ABORT.indication		

有证实(confirmed)服务的符号标记是 CONF_SERV,指明使用 BACnet 的有证实服务 PDU。无证实(unconfirmed)服务的符号标记是 UNCONF_SERV,指明使用 BACnet 的无证实服务 PDU。分段确认（segment acknowledge）服务的符号标记是 SEGMENT_ACK，指明使用 BACnet 的分段确认 PDU。差错（error）服务的符号标记是 ERROR，指明使用 BACnet 的差错 PDU。拒绝（reject）服务的符号标记是 REJECT，指明使用 BACnet 的拒绝 PDU。中止（abort）服务的符号标记是 ABORT，指明使用 BACnet 的中止 PDU。

当应用程序需要同远程的应用进程通讯时，它所要进行的操作是通过 API 访问本地的 **BACnet 用户元素**。应用程序调用 API 接口，并且将诸如服务请求接收设备的标识符（或地址）和协议控制信息等作为参数传递给 API，而将通信内容作为数据传递给 API。API 将参数直接下传到网络层或数据链路层，而将数据组成一个应用层服务原语，通过 **BACnet 用户元素**传递给 **BACnet 应用层服务元素**。从概念上来讲，由应用层服务原语产生的应用层协议数据单元 APDU（application protocol data unit)，构成了网络层服务原语的数据部分，并通过**网络层服务访问点 NSAP（Network Service Access Point）**下传到网络层。按照这

样的方式，这个请求进一步下传到本地设备协议栈的以下各层。整个过程如图 5-2 所示。于是，报文就这样被传送到远程的设备，并在远程设备协议栈中逐级上传，最后指示原语看起来似乎是直接从远程的 **BACnet 应用层服务元素**上传到远程的 **BACnet 用户元素**。同样，任何从远程设备发回的响应，也是以这样的方式回传给请求设备的。



PCI： 协议控制信息（Protocol Control Information）
PDU：协议数据单元（Protocol Data Unit）

图 5-2. BACnet 协议栈及数据流

应用实体通过 API 与应用程序除了交换服务原语和服务参数之外，还交换接口控制信息 ICI（interface control information）参数。ICI 的具体内容取决于服务原语的类型。应用实体将接收到的 ICI 参数下传至下面各层，从而使得各层可以构建自己的 PDU。而由应用实体回传给应用程序的 ICI 参数，则包含了下面各层从各自 PDU 中得到的信息。

通过 API 与各种服务原语交换信息的 ICI 参数包括：

“目的地址 DA（destination_address）”：将要接收服务原语设备的地址。其格式（如设备名称、网络地址等）只与本地有关。这个地址也可以是多目地址、本地广播地址或全局广播地址类型。

“源地址 SA（source_address）”：发送服务原语的设备的地址。其格式只与本地有关。

“网络优先级 NP（network_priority）”：在 6.2.2 节中所描述的一个四级网络优先级参数。

“期待回复数据 DER（data_expecting_reply）”：一个逻辑值参数，用来指明某个服务是否需要一个回复的服务原语。

表 5-1 说明了各种 ICI 参数对各种服务原语的适用性。

表 5-1. ICI 参数在各种服务原语中的适用性

服务原语	DA	SA	NP	DER
CONF_SERV.request	Yes	No	Yes	Yes
CONF_SERV.indication	Yes	Yes	Yes	Yes
CONF_SERV.response	Yes	No	Yes	Yes
CONF_SERV.confirm	Yes	Yes	Yes	No
UNCONF_SERV.request	Yes	No	Yes	No
UNCONF_SERV.indication	Yes	Yes	Yes	No
REJECT.request	Yes	No	Yes	No
REJECT.indication	Yes	Yes	Yes	No
SEGMENT_ACK.request	Yes	No	Yes	No
SEGMENT_ACK.indication	Yes	Yes	Yes	No
ABORT.request	Yes	No	Yes	No
ABORT. indication	Yes	Yes	Yes	No

BACnet 设备（BACnetDevice）是指任何一种支持用 BACnet 协议进行数字通信的真实的或者虚拟的设备。由 12.9 节中的定义可知，每一个 **BACnet 设备**必须且只能包含一个**设备（Device）**对象。每一个 **BACnet 设备**，都由一个 NSAP 唯一定位。在 NASP 中，包含了一个网络编号和一个 MAC 地址。

在多数情况下，一个 **BACnet 设备**就是一个物理设备。然而在某些情况下，一个单一的物理设备也可以形成多个“虚拟的” **BACnet 设备**。(参见附件 H)。

5.1.1 有证实的应用层服务

BACnet 基于客户/服务器通信模型定义了有证实的应用层服务。客户方通过具体的服务请求实例向服务器方请求服务，服务器方通过响应请求来为客户方提供服务，这种关系如图 5-3 所示。在交互过程中，担当客户角色的 BACnet 用户，称为请求方 BACnet 用户；担当服务器角色的 BACnet 用户，称为响应方 BACnet 用户。

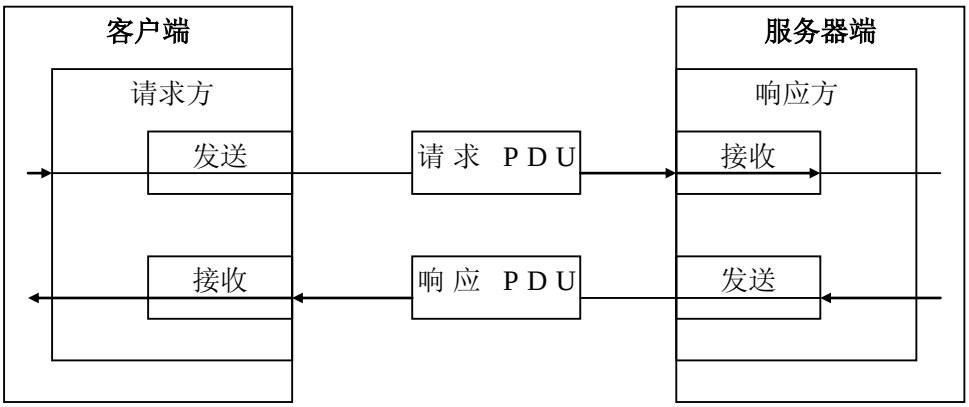


图 5-3.客户与服务器的关系

有证实应用层服务的具体过程如下：由请求方 BACnet 用户发出一个有证实服务请求原语（CONF_SERV.request），形成请求 PDU，发送给响应方 BACnet 用户。当这个请求 PDU 到达响应方 BACnet 用户时，响应方 BACnet 用户则收到一个有证实服务指示原语（CONF_SERV.indication)。同样，由响应方 BACnet 用户发出的一个有证实服务响应原语（CONF_SERV.response)，形成响应 PDU 回传给请求方 BACnet 用户。当响应 PDU 到达请求方 BACnet 用户时，请求方 BACnet 用户则收到一个有证实服务证实原语(CONF_SERV.confirm)。无论是请求方 BACnet 用户还是响应方 BACnet 用户，在该过程中都进行了 PDU 的发送和接收。因此，所谓“发送方 BACnet 用户”指的是发起一个 PDU 发送的 BACnet 用户；而“接收方 BACnet 用户”指的是接收到 PDU 到达指示的 BACnet 用户。

5.1.2 无证实的应用层服务

在无证实应用层服务中，不存在上述客户/服务器模型、“请求方 BACnet 用户”和“响应方 BACnet 用户”等概念，只有“发送方 BACnet 用户”和“接收方 BACnet 用户”，BACnet 标准用它们来定义无证实的应用层服务的服务过程。

5.2 BACnet 报文的分段

为了实现长报文（长度大于通信网络、收/发设备所支持的长度）的传输，BACnet 采取了应用层报文分段的机制来对报文进行分段。在 BACnet 中只有有证实请求（Confirmed-Request）和复杂确认（Complex-ACK）报文可能需要分段，因此分段还是 BACnet 的一个可选特性。

5.2.1 报文分段规则

本节规范对报文进行分段的规则。

5.2.1.1 APDU 数据流的分段规则

每个 BACnet 报文是一个由标记符和数值所组成的序列，这个序列是按照第 21 节中的 ASN.1 定义和第 20 节中的编码规则编码而成的。对于这样的数据流进行分段，依照下列的规则进行：

(a) 一个完整的报文尽可能作为一个 APDU 发送。

(b) 当一个完整的报文不可能作为一个 APDU 发送时，则应分段成最少个数的多个 APDU 发送。

(c) 对报文进行分段时，必须以字节（8 个比特）作为最小的分割单位。

5.2.1.2 APDU 最大长度的确定

在 BACnet 中，APDU 的最大长度不是固定的，其具体值是下列各长度值中的最小值：

(a) 设备所能发送的 APDU 的最大长度。这个长度一般与本地设备的缓冲区大小等因素有关。

(b) BACnet 互联网所能传输到远程设备的 APDU 的最大长度。这个长度一般由本地、远程、以及中间传输网络的数据链路所允许的网络层协议数据单元（NPDU）的最大长度所决定，在第 6 节中将对此详细说明。

(c) 远程设备所能接收的 APDU 的最大长度，其值不能小于 50 个字节。

如果报文的发送设备是请求方 BACnet 用户，也就是说要发送的 APDU 是 **BACnet 有证实请求 PDU（BACnet-Confirmed-Request-PDU）或者是 BACnet 无证实请求 PDU（BACnet-Unconfirmed-Request-PDU）**，这时远程设备所能接受 APDU 的最大长度，由远程**设备**对象的**最大 APDU 长度支持（Max_APDU_Length_Accepted）**属性所确定。可以通过第 15 节中描述的**读属性（ReadProperty）**服务来读取该属性值；也可以按 16.9 节中所述的方法，向远程设备发送一个 Who-Is 服务请求，使其回复一个 I-Am 服务响应，通过读取其中的‘**最大 APDU 长度支持（Max APDU Length Accepted）**’参数来获得该属性值。

如果报文的发送设备不是请求方 BACnet 用户，也就是说要发送的 APDU 是 **BACnet 复杂确认 PDU（BACnet-Complex-ACK-PDU）**，这时远程设备所能接受的 APDU 的最大长度，由本报文所响应的那个 **BACnet 有证实请求 PDU** 中的‘**最大 APDU 长度支持**’参数所确定。

上述的各种约束值最终决定了最大可传输长度的大小。需要注意的是，除非各个约束值都取最小值，否则最大可传输长度（maximum-transmittable-length）不会是一个常数。

5.2.2 分段协议控制信息（PCI）

为了支持报文分段，BACnet 规定 **BACnet 有证实请求** PDU 和 **BACnet 复杂确认** PDU 的头部，包含两个布尔型参数，分别是‘**报文分段（Segmented Message）**’和‘**后继（More Follows）**’。

如果报文经编码处理后生成的 APDU 的长度，小于或等于 5.2.1 节所确定的最大可传输长度的值，则‘**报文分段**’和‘**后继**’两个参数都应设置为 FALSE。

反之，如果报文经编码处理后生成的 APDU 的长度，大于 5.2.1 节所确定的最大可传输长度的值，则所有的分段中‘**报文分段**’参数都应设置为 TRUE；并且除了最后一个分段，其它所有分段中的‘**后继**’参数也应设置为 TRUE。

此外，由**有证实请求**报文或**复杂确认**报文的每个分段所生成 APDU 的头部，还有两个条件参数。第一个条件参数是‘**序号（Sequence Number）**’，这个八位二进制无符号整数被报文分段的发送方用来指定当前分段在整个报文分段序列中的位置。另一个条件参数是‘**预设窗口尺寸（Proposed Window Size）**’，它同样是一个八位二进制无符号整数。报文分段的发送方用该参数来指明，在收到一个**分段确认（SegmentAck）**报文之前，它预备发送的最大报文分段数。这两个参数在分段报文发送中的具体用法将在5.3节和5.4节中进一步说明。

起始分段中的‘**序号**’参数应设置为 0。报文分段接收方在收到一个或一组报文后，会向发送方发送一个包含有‘**序号**’参数的**分段确认** PDU。序号的大小，等于最近一次成功接收到的分段的‘**序号**’参数值。这样，这个**分段确认** PDU 不但用来请求发送方继续发送后一个或一组分段，还用来对先前一个分段或者是先前所有未被确认的分段（当 Window Size 值大于 1 时）的确认。

当然，在分段传输交互过程中的任何一方如果想中止交互过程，只要发送一个**中止** PDU（Abort-PDU）即可。

5.3 BACnet APDU 的传输

在 5.4 节中，将对 BACnet 的 APDU 收发协议进行正式的表述，称之为**事务处理状态机（Transaction State Machine）**。本节对整个收发协议作个整体概述。

5.3.1 有证实请求报文的传输

客户端设备在下列两种情况下将启动一个计时器，用来计量对所关注的报文确认的时间。这两种情况是：(1)传输了一个完整未分段**有证实请求**报文后；(2)在等待接收分段**有证实请求**报文最后一个分段所对应的**分段确认**报文的确认信息时。一旦收到了一个关于那个还没有确认的**有证实请求**报文所发的**差错** APDU、**拒绝** APDU、**中止** APDU、**简单确认（SimpleACK）** APDU 或**复杂确认（ComplexACK）** APDU，则停止计时，同时通知客户应用程序。如果计时器

的时间值超出了客户**设备**对象的 APDU **超时**（APDU_TimeOut）属性值，则整个**有证实请求**报文重新发送，计时器重新计时。所有**有证实请求**报文的超时重发过程都按照上述过程进行，直至重发次数超出了客户端**设备**对象的 APDU **重发次数**（Number_Of_APDU_Retries）属性所规定的次数。若超出规定值仍未收到响应，该报文将被丢弃，同时通知客户应用程序。

5.3.2 分段的有证实请求报文的传输

客户端设备在发送分段的**有证实请求** PDU 的第一个分段之前，首先选择一个**预设窗口尺寸**值，以表示它在收到一个**分段确认**报文之前，一次准备发送的报文分段最大个数。**预设窗口尺寸**值的选取由开发者自己决定，这只与客户端设备有关，但是其取值范围限制在从 1 到 127。这个值由**有证实请求** PDU 分段的‘**预设窗口尺寸**’参数表示，并且同一报文的每个分段该参数值都应该相同。

在发送了**有证实请求**报文的第一个分段后，客户端设备将启动一个计时器，用来计量对该分段确认的时间。一旦收到了一个关于这个还没有确认的**有证实请求**报文分段所发的**差错** APDU、**拒绝** APDU、**中止** APDU、或**分段确认** APDU，则停止计时。如果计时器的时间值超出了客户**设备**对象的 APDU **分段超时**（APDU_Segment_Timeout）属性值，则重发这个分段，并将计时器置零，重新计时。所有分段的超时重发过程都按照上述过程进行，直至重发次数超出了客户端**设备**对象中 APDU **重发次数**属性所规定的次数。若超出规定值仍未收到响应，整个报文将被丢弃，同时通知客户应用程序。

当服务器端设备收到分段的**有证实请求** PDU 报文的第一个分段后，将会选择一个**实际窗口尺寸**（Actual Window Size）值，以表示它在发送一个**分段确认**报文之前，一次准备接收的报文分段最大个数。**实际窗口尺寸**值的选取由开发者自己决定，这只与服务器设备有关，但是它还不能大于**有证实请求** PDU 中的‘**预设窗口尺寸**’参数值，而且其取值范围限制在从 1 到 127。对同一**有证实请求**报文所回应的所有**分段确认**报文，其**实际窗口尺寸**值都应该相同。需要特别说明的是，不管**实际窗口尺寸**值是多少，服务器设备在收到**有证实请求**报文的第一个分段后，应立即响应一个**分段确认**报文。

在收到第一个**分段确认** APDU 后，客户端设备得到该报文中携带的‘实际窗口尺寸（actual-window-size）’参数值，并将该值设置成客户设备自己的**实际窗口尺寸**。这样，客户端设备和服务器端设备的**实际窗口尺寸**值一致。客户端便可以在收到一个**分段确认** APDU 之前，一次发送**实际窗口尺寸**值所规定的多个分段。客户端设备从收到一个**分段确认** APDU 到发送一个分段，中间的时间间隔不能超过参数 T_{seg} 的值。同样，在连续发送的每个分段之间，时间间隔也不能超过 T_{seg} 值。客户端发送完整个报文或发送了窗口值个数的多个分段后，将启动一个计时器，用来计量对这些报文分段确认的时间。一旦收到了一个关于某些或者全部还没有确认的**有证实请求**报文分段所发的**拒绝** APDU、**中止** APDU 或**分段确认** APDU，则停止计时。如果计时器的时间值超出了客户**设备**对象的 APDU **分段超时**属性值，则重发这些分段，并且将计时器置零，重新计时。所有报文分段的超时重发过程都按照上述过程进行，直至重发次数超出了客户端**设备**对象中 APDU **重发次数**属性所规定的次数。如果超出规定值

仍未收到响应，该报文将被丢弃，同时通知客户应用程序。

当客户端设备正在发送一个**有证实请求**报文分段序列的过程中，即使需要确认的分段数量小于**实际窗口尺寸**值，都可能会收到**拒绝** APDU、**中止** APDU 或**分段确认** APDU。在这些情况发生时，如果收到**拒绝** APDU 或**中止** APDU，则停止**有证实请求**报文的传输，如果收到**分段确认** APDU，则将此作为序号等于或小于该 APDU 中‘**序号**’参数值的所有分段的确认，而那些没有被确认的分段将按上述过程进行重发。

需要注意的是还存在这样的情况，当客户端设备收到**拒绝** APDU、**中止** APDU 或**分段确认** APDU 时，可能已经发送或者已经向发送队列中传送了一个或多个（数量少于**实际窗口尺寸**值）额外的**有证实请求** PDU 分段。

5.3.3 分段的复杂确认报文的传输

服务端设备在发送分段**复杂确认** PDU 的第一个分段之前，首先选择一个**预设窗口尺寸**值，以表示它在收到一个**分段确认**报文之前，一次准备发送的报文分段最大个数。**预设窗口尺寸**值的选取由开发者自己决定，这只与服务器端设备有关，但是其取值范围限制在从 1 到 127。**复杂确认** PDU 的每一个分段中都有一个‘预设窗口尺寸（proposed-window-size）’参数，携带服务器端设备的**预设窗口尺寸**值，并且同一报文的每个分段的这个参数值都应该相同。

在传输完第一个分段后，服务端设备将启动一个计时器，用来计量对该报文分段确认的时间。一旦收到了一个关于这个还没有确认的**复杂确认**报文分段所发送的**中止** APDU 或**分段确认** APDU，则停止计时。如果计时器的时间值超出了服务器**设备**对象的 APDU **分段超时**属性值，则重发这个分段，并将计时器置零，重新计时。所有分段的超时重发过程都按照上述过程进行，直至重发次数超出了服务器端**设备**对象中 APDU **重发次数**属性所规定的次数。若超出规定值仍未收到响应，整个报文将被丢弃。

当客户端设备收到分段的**复杂确认** APDU 报文的第一个分段后，将会选择一个**实际窗口尺寸**值，以表示它在发送一个**分段确认**报文以前，一次准备接收的报文分段最大个数。**实际窗口尺寸**值的选取由开发者自己确定，这只与客户设备有关，但是它不能大于**复杂确认** PDU 中的‘预设窗口尺寸’参数值，而且其取值范围限制在从 1 到 127。对同一**复杂确认**的所有**分段确认**，其**实际窗口尺寸**参数值都应该相同。需要特别说明的是，不管**实际窗口尺寸**是多少，客户端设备在收到**复杂确认**报文的第一个分段后，就应立即发送一个**分段确认**报文。

在收到一个**分段确认** APDU 后，服务端设备得到该报文中携带的‘实际窗口尺寸’参数值，并将该值设置成服务器设备自己的**实际窗口尺寸**。这样，服务器端设备便可以在收到一个**分段确认** APDU 之前，一次发送‘实际窗口尺寸’参数值所规定的多个分段。服务端设备从收到一个**分段确认** APDU 到发送一个分段，中间的时间间隔不能超过参数 T_{seg} 的值。同样，在连续发送的每个分段之间，时间间隔也不能超过 T_{seg} 值。服务器端设备发送完整个报文或发送了窗口值个数的多个分段后，将启动一个计时器，用来计量对这些报文分段确认的时间。

一旦收到了一个关于某些或者全部还没有确认的**复杂确认**报文分段所发的**中止 APDU** 或**分段确认 APDU**，则停止计时。如果计时器的时间值超出了服务器**设备**对象的 **APDU 分段超时**属性值，则重发这些分段，并且将计时器置零，重新计时。所有报文分段的超时重发过程都按照上述过程进行，直至重发次数超出了服务器端**设备**对象中 **APDU 重发次数**属性所规定的次数。如果超出规定值仍未收到响应，该报文将被丢弃。

服务器端在发送**复杂确认**报文分段的过程中，即使需要确认的分段数量小于**实际窗口尺寸**值，都可能会收到**中止 APDU** 或**分段确认 APDU**。在这些情况发生时，如果收到**拒绝 APDU** 或**中止 APDU**，则停止**复杂确认**报文的传输，如果收到**分段确认 APDU**，则将此作为序号等于或小于该 APDU 中‘**序号**’参数值的所有分段的确认，而那些没有被确认的分段将按上述过程进行重发。

需要注意的是还存在这样的情况，当服务器端设备收到一个**中止 APDU** 或**分段确认 APDU** 时，可能已经发送或者已经向发送队列中传送了一个或多个（数量少于**实际窗口尺寸**值）额外的**有证实请求 PDU** 分段。

5.3.4 分段确认 APDU 的传输

分段确认 APDU 是分段接收方设备用来确认发送方的信息。在以下四种情况下，设备应传送一个**分段确认**报文。

(a) 设备收到分段的报文的第一个分段。在这种情况下，用于确认的**分段确认**报文中参数‘否定确认（negative-ACK）’应设置为 FALSE，以表明这是个肯定的确认；参数‘序号’应设置为 0，以表明第一个分段已经被确认，分段的发送方可以继续发送后续的分段。

(b) 设备收到未确认的、有序的、数量为**实际窗口尺寸**的多个报文分段构成的序列。在这种情况下，用于确认的分段确认报文中参数‘否定确认’应设为 FALSE，以表明这是个肯定的确认;，参数‘序号’应设置为此次所收的最后一个分段的‘序号’值，以表明包括该‘序号’在内的分段都被确认了，分段的发送方可以继续发送后续的分段。

(c) 设备收到一个乱序分段报文（可能表明丢失了某个分段）。在这种情况下，分段接收方将丢弃这个乱序分段。此处“乱序（out of order）”这个词的意思是指所收的分段的‘序号’不等于所预期的‘序号’。用于确认的分段确认报文中参数‘否定确认’应设为 TRUE，以表明这是个否定的确认；参数‘序号’应设置为最后一次正确收到的分段的‘序号’值，以表明到此‘序号’值为止的所有分段都被确认了，分段的发送方可以继续发送后续的分段。

(d) 设备收到报文的最后一个分段。在这种情况下，用于确认的分段确认报文中参数‘否定确认’应设为 FALSE，以表明这是个肯定的确认；参数‘序号’应设置为最后报文分段的‘序号’值，以表明所有分段都被确认了。

5.3.5 重复的 APDU 和报文分段

5.3.5.1 客户端事务处理状态机的中止

当使用 BACnet 标准定义的差错重传机制时，在报文传输的交互处理中，必然存在收到重复报文或分段的可能。在客户端，当发送了**有证实请求** APDU 报文（或报文的第一个分段）之后，将创建**事务处理状态机**，进行事务处理。当出现以下四种情况之一时，客户端的事务处理中止，同时结束该**事务处理状态机**。

(a) 当收到服务器端设备发来的、包含有这个事务处理的**调用标识符（invokeID）**的五种报文时。这五种报文分别是：**简单确认** APDU、不分段的**复杂确认** APDU、**差错** APDU、**拒绝** APDU 和**中止** APDU。

(b) 收到服务器发来的分段的**复杂确认** APDU 的最后一个分段，并发送了相应的**分段确认** APDU 后。

(c) 当超时重传次数用尽后。

(d) 在向服务器发送了包含有该事务处理的调用标识符的**中止** APDU 后（例如：客户端中止这个事务处理）。

5.3.5.2 服务端事务处理状态机的中止

在服务端，当收到了**证实请求** APDU 报文（或报文的第一个分段）之后，将创建**事务处理状态机**，进行事务处理。当出现以下四种情况之一时，服务器端的事务处理中止，同时结束该**事务处理状态机**。

(a) 当向客户端设备发送完成包含有这个事务处理的**调用标识符**的五种报文时。这五种报文分别是：**简单确认** APDU、不分段的**复杂确认** APDU、**差错** APDU、**拒绝** APDU 和**中止** APDU。

(b) 当收到客户端设备发来的关于分段的**复杂确认** APDU 最后一个分段的**分段确认** APDU 之后。

(c) 当接收到客户端发来的包含有该事务处理调用标识符的**中止** APDU 后。

(d) 在传输一个分段的**复杂确认** APDU 的过程中，超时重传次数达到规定值仍未成功时。

5.3.5.3 重复报文的处理

重复报文或重复报文分段的处理过程如下：

(a) 当服务器接收到一个重复的**有证实请求**报文时，如果服务器具有识别重复的**有证实请求**报文的能力，则该重复的报文将被服务器丢弃。否则，服务器仍然响应这个重复的**有证实请求**报文，在这种情况下，客户将根据这个服务器响应报文中的**调用标识符**不与任何一个当前的**事务处理状态机**绑定这个情况，而丢弃这个响应报文。

(b) 当服务器接收到一个重复的**有证实请求**报文分段，即已经收到并针对该分段发送了**分段确认**报文时，服务器将丢弃这个重复的分段，并回传一个合适的**分段确认 APDU**。任何分段都可由对等方地址（peer address）、**调用标识符**和分段的**序号**唯一确定。

(c) 当客户接收到一个重复的**复杂确认**报文分段，即已经收到并针对该分段发送了**分段确认**报文时，客户将丢弃这个重复的分段，并回传一个合适的**分段确认 APDU**。任何分段都可由对等方地址、**调用标识符**和分段的**序号**唯一确定。

(d) 当一个设备接收到一个重复的**分段确认 APDU** 时，该设备将丢弃这个重复的**分段确认 APDU**。其它的一些措施（包括可能的报文分段重发），应符合 5.4 节中的规定。

5.3.6 失效资源（Stale Resource）的处理

上述 BACnet 的差错重传过程，其具体实现需要客户和服务器两端提供一定的资源。这些资源通常是事务处理的各个细节，包括**事务处理状态机**、计时器、以及 APDU 或 APDU 分段缓冲区等。当差错重传过程失败时，这些相关的资源也变得失效，应被释放掉。资源释放的具体细节取决于系统的具体设计。作为设计建议，本协议给出了资源失效而应释放的依据：

(a) 在客户端，当收到对一个**有证实请求 APDU** 的完整响应后。

(b) 在客户端，当一个**有证实请求 APDU** 报文被重发了 **APDU 重发次数**属性所规定的次数，但仍未成功时。

(c) 在客户端，当一个**有证实请求 APDU** 分段被重发了 **APDU 重发次数**属性所规定的次数，但仍未成功时。

(d) 在服务器端，当发送了对某个**有证实请求 APDU** 的响应并收到相应的**分段确认**后。

5.4 应用层协议状态机

BACnet 的 APDU 可以分为两种：一种是由请求方 BACnet 用户（客户）发送的，另一种是由响应方 BACnet 用户（服务器）发送的。所有的 BACnet 设备都应能工作在响应方 BACnet 用户状态，从而能够接收请求方 BACnet 用户发来的 APDU。另一方面，许多设备还可以工作在请求方 BACnet 用户状态，接收响应方 BACnet 用户发来的 APDU。

请求方 BACnet 用户（客户）发送的 APDU：

BACnet 无证实请求 PDU（BACnet-Unconfirmed-Request-PDU）

BACnet 有证实请求 PDU（BACnet-Confirmed-Request-PDU）

BACnet 分段确认 PDU（BACnet-SegmentACK-PDU）（参数‘服务器（server）’=FALSE）

BACnet 中止 PDU（ BACnet-Abort-PDU）（参数‘服务器’=FALSE）

响应方 BACnet 用户（服务器）发送的 APDU：

BACnet 简单确认 PDU（BACnet-SimpleACK-PDU）

BACnet 复杂确认 PDU（BACnet-ComplexACK-PDU）

BACnet 差错 PDU（BACnet-Error-PDU）

BACnet 拒绝 PDU（BACnet-Reject-PDU）

BACnet 分段确认 PDU（BACnet-SegmentACK-PDU）（参数‘服务器’=TRUE）

BACnet 中止 PDU（BACnet-Abort-PDU）（参数‘服务器’=TRUE）

请求方和响应方 BACnet 用户为每一个事务处理都要创建并且维护一个**事务处理状态机（简称 TSM）**。该 TSM 在事务开始时创建，在事务结束时中止。其过程如下：由**空闲（IDLE）**状态转变进入到 TSM 状态，建立 TSM；由 TSM 状态转变到**空闲**状态，中止 TSM。一个事务处理由客户 BACnet **地址（BACnetAddress）**、服务器 BACnet **地址**和**调用标识符**（如果存在的话）唯一确定。

当一个 PDU 从网络层传递给应用层时，应用层软件检查 PDU 的类型、源 BACnet **地址**、目的 BACnet **地址**、**调用标识符**（如果存在的话），确定 PDU 的类型（是请求方还是响应方 BACnet 用户）以及应送给哪个 TSM 处理。如果不存在对应的 TSM，必须新建一个 TSM。

当一个请求从应用程序传递给应用层时，应用层软件检查请求的类型、源 BACnet **地址**、目的 BACnet **地址**、**调用标识符**（如果存在的话），确定请求的类型（是请求方还是响应方 BACnet 用户）以及应送给哪个 TSM 处理。如果不存在对应的 TSM，必须新建一个 TSM。

为了简化状态机的描述，这里仅给出**应用实体**所进行分段的情况，**应用程序**所进行的分段也是存在的。在这种情况下，当前的 TSM 无论是接收一个分段还是一组分段或是发送**分段确认**，分段都将通过更新后的 TSM 传递给**应用程序**，而**分段确认**则只沿着从**应用程序**通过**分段确认请求（SEGMENT_ACK.request）**原语这个方向传递。当更新的状态机接收到**分段确认** PDU，就向**应用程序**传递一个**分段确认指示（SEGMENT_ACK.indication）**原语。

5.4.1 变量与参数

在每个**事务处理状态机**的实例中，都使用了一些变量，定义如下：

重发计数（RetryCount）：用于计数 APDU 重发的次数

分段重发计数（SegmentRetryCount）：用于计数分段重发的次数

分段发送完毕（SentAllSegments）：用于控制 APDU 的重发及接收服务器的响应

最近序号（LastSeqenceNumber）：用于保存顺序接收的最近一个分段的序号

初始序号（InitialSequenceNumber）：用于保存填充一个窗口的分段序列的第一个分段的序号

实际窗口尺寸（ActualWindowSize）：用于保存当前窗口尺寸值

预设窗口尺寸（ProposedWindowSize）：用于保存分段发送方的预设的窗口尺寸值

分段计时器（SegmentTimer）： 用于记录 PDU 分段的超时

请求计时器（RequestTimer）： 用于记录**有证实请求**的超时

以下是所用的参数的说明：

T_{seg} ： 这个参数是一个节点在发送完一个分段序列的最后一个分段后，等待**分段确认** PDU 的时间长度。这个时间长度值等于节点的**设备**对象中 APDU **分段超时**属性值。

$T_{wait\_for\_seg}$ ：这个参数是一个节点在发送完**分段确认** PDU 后，等待后续报文分段的时间长度。这个时间长度值是节点的**设备**对象中 APDU **分段超时**属性值的 4 倍。

T_{out} ： 这个参数表示节点的**设备**对象中 APDU **超时**属性值。

N_{retry}：这个参数表示节点的**设备对象**中 APDU **重发次数**属性值。

5.4.2 InWindow 函数

函数“InWindow”用于完成对两个序号（由无符号 8 位整数构成）的模 256 运算。所有的计算和比较都是在 8 位无符号整数基础上模 256 运算。

函数 InWindow(seqA, seqB)

(a) 如果参数 seqA 减去参数 seqB 的差再模 256，其结果少于**实际窗口尺寸**，则返回 TRUE

(b) 否则返回 FALSE。

例程（参考资料）：假定**实际窗口尺寸**为 4，则

InWindow(0,0)返回 TRUE

InWindow(1,0)返回 TRUE

InWindow(3,0)返回 TRUE

InWindow(4,0)返回 FALSE

InWindow(4,5)返回 FALSE（因为(4-5) mod 256 = 255）

InWindow(0,255)返回 TRUE（因为(0-255) mod 256 = 1）

5.4.3 FillWindow 函数

函数“FillWindow”用于发送一组 PDU 分段，该 PDU 分段序列的数目或者正好填充窗口，或者是一个报文的最后部分分段。在收到一个**分段确认** APDU 到发送下一个分段序列，时间间隔不能大于 T_{seg} 值。分段序列中两个相邻分段发送的时间间隔，也不能大于 T_{seg} 值。

函数 FillWindow(sequenceNumber)

(a) 将局部变量 ix 值设置为 0。

(b) 如果下一个要发送的分段（分段的序号是**序号**加上 ix）是报文的最后一个分段，则跳至步骤（g）。

(c) 产生一个参数‘期待回复数据’=TRUE 的 N-UNITDATA.request，用来发送后续的 BACnet APDU 分段。此分段中参数‘报文分段’＝TRUE，‘后继’＝TURE，‘预设窗口尺寸’＝**预设窗口尺寸**值，‘序号’＝(**序号**+ix)模 256。

(d) 将 ix 自加 1。

(e) 如果 ix 小于**实际窗口尺寸**值，跳至步骤（b）。

(f) 跳至步骤（i）。

(g) 产生参数‘期待回复数据’＝TRUE 的 N-UNITDATA.request，用来发送最后一个 BACnet APDU 分段。此分段中参数‘报文分段’＝TRUE，‘后继’＝FALSE，‘预设窗口尺寸’＝**预设窗口尺寸**值，‘序号’＝(**序号**+ix)模 256。

(h) 将变量**分段发送完毕（SentAllSegments）**设置为 TRUE，用来表示此次已经将所有分段发送完毕。

(i) 返回到调用者。

5.4.4 请求方 BACnet 用户（客户）状态机

5.4.4.1 空闲状态

在**空闲**状态，设备等待着本地应用程序请求服务。

发送无证实请求（SendUnconfirmed）

如果设备接收到本地应用程序发出的 UNCONF_SERV.request，

则产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个 **BACnet 无证实请求 PDU**，然后进入**空闲**状态。

发送有证实不分段请求（SendConfirmedUnsegmented）

如果设备接收到本地应用程序发出的 CONF_SERV.request、并且 APDU 的长度不大于 5.2.1 节所确定的“最大可传输长度”，

则设备对该事务过程指定一个**调用标识符**，设置变量**分段发送完毕**＝TRUE，**重发计数**＝0，启动**请求计时器**，产生一个参数‘期待回复数据’＝TRUE 的 N-UNITDATA.request 报文，用来传送一个参数‘报文分段’＝FALSE 的 **BACnet 有证实请求 PDU**，然后进入**等待证实状态（AWAIT_CONFIRMATION）**，等待确认。

发送有证实分段请求（SendConfirmedSegmented）

如果设备接收到本地应用程序发出的 CONF_SERV.request、并且 APDU 的长度大于 5.2.1 节所确定的“最大可传输长度”，

则设备对该事务过程指定一个**调用标识符**，设置变量**分段发送完毕**＝FALSE，**重发计数**＝0，**分段重发计数**＝0，**初始序号**＝0，**预设窗口尺寸**＝所希望的大小，**实际窗口尺寸**＝1，启动**请求计时器**，产生一个参数‘期待回复数据’＝TRUE 的 N-UNITDATA.request 报文。该请求报文用来传送参数‘报文分段’＝TRUE、‘后继’＝TURE、‘序号’＝0、‘预设窗口尺寸’＝**预设窗口尺寸**的一个 **BACnet 有证实请求 PDU**,这个 PDU 包含了这个报文的第一个分段。然后进入**分段请求(SEGMENTED_REQUEST)**状态，等待确认。

（注：**预设窗口尺寸**值的选取由开发者自己决定，这只与本地设备有关，但是其取值范围限制在从 1 到 127。）

接收到意外分段信息（UnexpectedSegmentInfoReceived）

如果设备从网络层接收到一个意外的 PDU,例如一个参数‘报文分段’＝TRUE 的 **BACnet 复杂确认 PDU**，或者是一个参数‘服务器’＝TRUE 的 **BACnet 分段确认 PDU** 时，这表明存在一个活动的服务器 TSM，

则产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送参数‘服务器’＝FALSE 的 **BACnet 中止 PDU**，然后进入**空闲**状态。

接收到意外 PDU（UnexpectedPDU_Received）

如果设备从网络层接收到一个意外的 PDU,例如一个参数‘报文分段’＝FALSE 的 **BACnet 简单确认 PDU**、**BACnet 复杂确认 PDU**，或者一个参数‘服务器’＝TRUE 的 **BACnet 差错 PDU**、**BACnet 拒绝 PDU**、**BACnet 中止 PDU** 时，这表明不存在一个活动的服务器 **TSM**，则进入**空闲**状态。（这里，客户端并不了解所讨论的事务处理，因而没有必要产生拒绝指示原语（REJECT.indication）、中止指示原语（ABORT.indication）等服务原语）

5.4.4.2 分段请求状态

在**分段请求**状态，设备等待接收针对 **BACnet 有证实请求 PDU** 所发出的一个或几个分段的 **BACnet 分段确认 PDU** 的确认报文。

接收到重复确认 PDU（DuplicateACK_Received）

如果设备从网络层接收到一个 **BACnet 分段确认 PDU** 报文，其中参数‘服务器’＝TRUE，而函数 InWindow(**BACnet 分段确认 PDU** 的‘序号’参数值，**初始序号**)返回值为 FALSE，则重启分段计时器，进入**分段请求**状态，继续等待确认。

接收到新确认 PDU（NewACK_Received）

如果设备从网络层接收到一个 BACnet 分段确认 PDU 报文，其中参数‘服务器’＝TRUE，而函数 InWindow(BACnet 分段确认 PDU 的‘序号’参数值，初始序号)返回值为 TRUE，同时设备至少还有一个报文分段等待发送，

则设置变量初始序号＝(BACnet 分段确认 PDU 的‘序号’+1)模 256，实际窗口尺寸=BACnet 分段确认 PDU 的‘实际窗口尺寸’参数值，分段重发计数＝0，调用函数 FillWindow(初始序号)来发送一个或多个 BACnet 有证实请求 PDU，这些 PDU 正好包含了后续的实际窗口尺寸个报文分段，然后进入分段请求状态，等待新的确认。

接收到最后一个确认 PDU（FinalACK_Received）

如果设备从网络层接收到一个 BACnet 分段确认 PDU 报文，其中参数‘服务器’＝TRUE，而函数 InWindow(BACnet 分段确认 PDU 的‘序号’，初始序号)的返回值为 TRUE，同时已没有分段需要发送，

则停止分段计时器，启动请求计时器，进入等待证实状态，等待服务响应。

分段超时（Timeout）

如果分段计时器的值超过了 T_{seg}、但分段重发计数的值又小于 N_{retry}，

则分段重发计数自加 1，重启分段计时器，调用函数 FillWindow(初始序号)以重新发送一个或多个 BACnet 有证实请求 PDU，这些 PDU 正好包含了后续的实际窗口尺寸个报文分段，然后进入分段请求状态，重新等待确认。

最终超时（FinalTimeout）

如果分段计时器的值超过了 T_{seg}、同时分段重发计数的值也超过 N_{retry}，

则停止分段计时器，向本地应用程序发送 CONF_SERV.confirm(-)服务原语，然后进入空闲状态。

接收到中止 PDU（AbortPDU_Received）

如果设备从网络层接收到一个 BACnet 中止 PDU 报文，其中参数‘服务器’＝TRUE，

则停止分段计时器，向本地应用程序发送 ABORT.indication 原语，然后进入空闲状态。

接收到简单确认 PDU（SimpleACK_Received）

如果设备从网络层接收到一个 BACnet 简单确认 PDU 报文、并且变量分段发送完毕＝

TRUE，

则停止**请求计时器**，向本地应用程序发送 CONF_SERV.confirm(+)原语，然后进入**空闲**状态。

接收到不分段复杂确认 PDU（UnsegmentedComplexACK_Received）

如果设备从网络层接收到一个 **BACnet 复杂确认 PDU** 报文，其中参数‘报文分段’＝FALSE，并且变量**分段发送完毕**＝TRUE，

则停止**请求计时器**，向本地应用程序发送 CONF_SERV.confirm(+)原语，然后进入**空闲**状态。

接收到分段复杂确认 PDU（SegmentedComplexACK_Received）

如果设备从网络层接收到一个 **BACnet 复杂确认 PDU** 报文，其中参数‘报文分段’＝TRUE、‘序号’＝0，同时该设备支持报文分段，并且变量**分段发送完毕**＝TRUE，

则停止**请求计时器**，然后根据接收到的 PDU 中的‘预设窗口尺寸’参数值以及设备自身情况，计算变量**实际窗口尺寸**的实际取值，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送参数‘否定确认（negative-ACK）’＝FALSE、‘服务器’＝FALSE、‘实际窗口尺寸’＝**实际窗口尺寸**值的一个 **BACnet 分段确认 PDU**，然后启动**分段计时器**，设置变量**最近序号**＝0，**初始序号**＝0，最后进入**分段证实（SEGMENTED_CONF）**状态，接收后续分段。

（注：**实际窗口尺寸**的值由开发者自行确定，只与本地设备有关，但是这个值的取值限制在 1 到 127 范围内，并且实际取值不能大于接收到的 **BACnet 复杂确认 PDU** 报文中的‘预设窗口尺寸’参数值。）

接收到差错 PDU（ErrorPDU_Received）

如果设备从网络层接收到一个 **BACnet 差错 PDU** 报文、并且变量**分段发送完毕**＝TRUE，

则停止**请求计时器**，向本地应用程序发送 CONF_SERV.confirm(-)服务原语，然后进入**空闲**状态。

接收到拒绝 PDU（RejectPDU_Received）

如果设备从网络层接收到一个 **BACnet 拒绝 PDU** 报文、并且变量**分段发送完毕**＝TRUE，

则停止**请求计时器**，向本地应用程序发送 REJECT.indication 原语，然后进入**空闲**状态。

接收到意外 PDU

如果设备从网络层接收到一个 **BACnet 简单确认 PDU**、**BACnet 复杂确认 PDU**、**BACnet 差错 PDU** 或者 **BACnet 中止 PDU** 报文、而变量**分段发送完毕**＝FALSE；或者从网络层接收到的 **BACnet 复杂确认 PDU** 报文且其参数‘报文分段’＝TRUE、而本设备又不支持报文分段；或者是从网络层接收到的 **BACnet 复杂确认 PDU** 报文其参数‘报文分段’”＝TRUE、参数‘序号’≠0，

在以上这些情况下，停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request，传送参数‘服务器’＝FALSE 的一个 **BACnet 中止 PDU**，同时向本地应用程序发送 CONF_SERV.confirm(-)原语，最后进入**空闲**状态。

发送中止 PDU（SendAbort）

如果设备从本地应用程序接收到 ABORT.request 服务原语，

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request，传送一个参数‘服务器’＝FALSE 的 **BACnet 中止 PDU**，然后进入**空闲**状态。

5.4.4.3 等待证实状态

在**等待证实**状态，设备等待接收针对 **BACnet 有证实请求 PDU** 所发出的响应。

接收到简单确认 PDU

如果设备从网络层接收到一个 **BACnet 简单确认 PDU**，

则停止**请求计时器**，向本地应用程序发送 CONF_SERV.confirm(+)服务原语，然后进入**空闲**状态。

接收到不分段复杂确认 PDU

如果设备从网络层接收到一个 **BACnet 复杂确认 PDU** 报文，其中参数‘报文分段’＝FALSE，

则停止**请求计时器**，向本地应用程序发送 CONF_SERV.confirm(+)服务原语，然后进入**空闲**状态。

接收到分段复杂确认 PDU

如果设备从网络层接收到一个 **BACnet 复杂确认 PDU** 报文，其中参数‘报文分段’＝TRUE、‘序号’＝0，同时该设备支持报文分段，

则停止**请求计时器**，根据接收到的 PDU 中的‘预设窗口尺寸’参数值以及设备自身情况，计算变量**实际窗口尺寸**的取值，并产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request，用来传送参数‘否定确认’＝FALSE、‘服务器’＝FALSE、‘实际窗口尺寸’＝变量**实际窗口尺寸**值的一个 **BACnet 分段确认 PDU**，然后启动**分段计时器**，设置变量**最近序号**＝0，变量**初始序号**＝0，最后进入**分段证实**状态，接收后续分段。

（注：**实际窗口尺寸**的值由开发者自行确定，只与本地设备有关，但是这个值的取值限制在 1 到 127 范围内，并且实际取值不能大于接收到的 **BACnet 复杂确认 PDU** 报文中的‘预设窗口尺寸’参数值。）

接收到差错 PDU

如果设备从网络层接收到一个 **BACnet 差错 PDU** 报文，

则停止**请求计时器**，向本地应用程序发送 CONF_SERV.confirm(-)原语，然后进入**空闲**状态。

接收到拒绝 PDU

如果设备从网络层接收到一个 **BACnet 拒绝 PDU** 报文，

则停止**请求计时器**，向本地应用程序发送 REJECT.indication 原语，然后进入**空闲**状态。

接收到放弃 PDU

如果设备从网络层接收到一个 **BACnet 放弃 PDU** 报文，其中参数‘服务器’＝TRUE，

则停止**分段计时器**，向本地应用程序发送 ABORT.indication 原语，然后进入**空闲**状态。

接收到分段确认 PDU

如果设备从网络层接收到一个**BACnet 分段确认 PDU**＝报文,其中参数‘服务器’＝TRUE，

则将此视为重复 PDU 而丢弃掉，然后返回当前状态。

接收到意外 PDU

如果设备从网络层接收到一个意外的 PDU，例如一个参数‘报文分段’＝TRUE，参数‘序号’≠0 的 **BACnet 复杂确认 PDU**，或者一个参数‘报文分段’＝TRUE，而此设备

又不支持报文分段的情况，

则停止**请求计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用以传送参数‘服务器’＝FALSE 的一个 **BACnet 放弃 PDU** 报文，同时向本地应用程序发送 CONF_SERV.confirm(-)服务原语，最后进入**空闲**状态。

请求超时不分段（TimeoutUnsegmented）

如果设备的**请求计时器**的值超过了 T_{out}，但**重发计数**的值又小于 **APDU 重发次数**参数值，并且**有证实请求** APDU 的长度不大于 5.2.1 节所确定的最大可传输长度值，

则停止**请求计时器**，将**重发计数**自加 1，产生一个参数‘期待回复数据’＝TRUE 的 N-UNITDATA.request 报文，用以传送参数‘报文分段’＝FALSE 的一个 **BACnet 有证实请求 PDU**，最后进入**等待证实**状态，重新等待确认。

请求超时分段（TimeoutSegmented）

如果设备的**请求计时器**的值超过了 T_{out}，但**重发计数**的值又小于 **APDU 重发次数**参数值，而**有证实请求 APDU** 的长度大于 5.2.1 节所确定的最大可传输长度，

则停止**请求计时器**，将**重发计数**自加 1，设置变量**分段重发计数**＝0，启动**分段计时器**，设置变量**初始序号**＝0，变量**实际窗口尺寸**＝1，产生一个参数‘期待回复数据’＝TRUE 的 N-UNITDATA.request 报文，用来传送参数‘报文分段’＝TRUE、‘后继’＝TURE、‘序号’＝0 的一个 **BACnet 有证实请求 PDU**，该 PDU 包含了这个报文的第一个分段，最后设备进入**分段请求**状态，等待确认。

最终超时

如果设备的**请求计时器**的值超过了 T_{out}，同时**重发计数**的值也超过了 **APDU 重发次数**参数值，

则停止**请求计时器**，向本地应用程序发送 CONF_SERV.confirm(-)原语，然后进入**空闲**状态。

发送放弃 PDU

如果设备从本地应用程序接收到 ABORT.request 原语，

则停止**请求计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝FALSE 的 **BACnet 放弃 PDU**，然后进入**空闲**状态。

5.4.4.4 分段证实状态

在**分段证实**状态，设备等待接收一个或者几个针对 **BACnet 分段确认 PDU** 的响应分段。

接收到新分段但无空间保存（NewSegmentReceived_NoSpace）

如果设备从网络层接收到一个 **BACnet 复杂确认 PDU**，其中‘报文分段’＝TRUE、‘序号’＝(**最近序号**+1)模 256，但设备自身已没有空间来存储新的分段，

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝FALSE 的 **BACnet 中止 PDU**，然后向本地应用程序发送 CONF_SERV.confirm(-)原语，最后进入**空闲**状态。

接收到新的分段

如果设备从网络层接收到一个 **BACnet 复杂确认 PDU** 报文，其中‘报文分段’＝TRUE、‘后继’＝TRUE、‘序号’＝(**最近序号**+1)模 256、且‘序号’≠(**初始序号+实际窗口尺寸**)模 256，

则保存此 **BACnet 复杂确认 PDU** 分段，将变量**最近序号**自加 1、模 256，重启**分段计时器**，然后进入**分段证实**状态，等待后续的分段。

接收到一组分段中的最后分段（LastSegmentOfGroupReceived）

如果设备从网络层接收到一个 **BACnet 复杂确认 PDU** 报文，其中‘报文分段’＝TRUE、‘后继’＝TRUE、‘序号’＝(**最近序号**+1)模 256、且‘序号’＝(**初始序号+实际窗口尺寸**)模 256，

则保存此 **BACnet 复杂确认 PDU** 分段，将变量**最近序号**自加 1、模 256，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送参数‘否定确认’＝FALSE、‘服务器’＝FALSE、‘实际窗口尺寸’＝**实际窗口尺寸**值的 **BACnet 分段确认 PDU**，重启**分段计时器**，最后进入**分段证实**状态，等待接收后续分段。

接收到复杂确认的最后分段（LastSegmentOfComplexACK_Received）

如果设备从网络层接收到一个 **BACnet 复杂确认 PDU**，其中‘报文分段’＝TRUE、‘序号’＝(**最近序号**+1)模 256、‘后继’＝FALSE（即最后一个分段），

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送参数‘否定确认’＝FALSE、‘服务器’”＝FALSE、‘实际窗口尺寸’＝**实际窗口尺寸**值的一个 BACnet **分段确认 PDU**，然后向本地应用程序发送一个 CONF_SERV.confirm(+)原语，该原语中包含了接收到的所有分段，最后进入**空闲**状态。

接收到乱序分段（SegmentReceivedOutOfOrder）

如果设备从网络层接收到一个 **BACnet 复杂确认 PDU**，其中‘报文分段’＝TRUE、‘序号’”≠(**最近序号**+1)模 256，

则丢弃此 PDU 分段，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘否定确认’”＝TRUE、‘服务器’＝FALSE、‘序号’＝**最近序号**、‘实际窗口尺寸’＝**实际窗口尺寸**的 **BACnet 分段确认 PDU**，然后重启**分段计时器**，进入**分段证实**状态，等待接收后续分组。

接收到中止 PDU

如果设备从网络层接收到一个 **BACnet 中止 PDU** 报文，其中参数‘服务器’＝TRUE，

则停止**分段计时器**，向本地应用程序发送 ABORT.indication 原语，然后进入**空闲**状态。

接收到意外 PDU

如果设备从网络层接收到一个意外的 PDU，例如参数‘报文分段’＝FALSE 的 **BACnet 简单确认 PDU**、**BACnet 复杂确认 PDU**，或者是参数‘服务器’＝TRUE 的 **BACnet 差错 PDU**、**BACnet 拒绝 PDU**、**BACnet 分段确认 PDU**，

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝FALSE 的 **BACnet 中止 PDU**，同时向本地应用程序发送 CONF_SERV.confirm(-)原语，最后进入**空闲**状态。

分段超时

如果设备的**分段计时器**的值超过了 T_{seg} 的 4 倍，

则停止**分段计时器**，向本地应用程序发送 CONF_SERV.confirm(-)原语，然后进入**空闲**状态。

发送中止 PDU

如果设备从本地应用程序接收到 ABORT.request，

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝FALSE 的 **BACnet 中止 PDU**，然后进入**空闲**状态。

5.4.5 响应方 BACnet 用户（服务器）状态机

5.4.5.1 空闲状态

在**空闲**状态，设备等待从网络层传递来一个 PDU。

接收到无证实请求 PDU

如果设备从网络层接收到一个 **BACnet 无证实请求 PDU**，

则向本地应用程序发送一个 UNCONF_SERV.indication 原语，然后进入**空闲**状态。

接收到不分段有证实请求 PDU

如果设备从网络层接收到一个 **BACnet 有证实请求 PDU**，其中参数‘报文分段’＝FALSE，

则向本地应用程序发送一个 CONF_SERV.indication 原语，然后进入**等待响应**状态。

接收到分段证实请求 PDU 且不支持

如果设备从网络层接收到一个 **BACnet 有证实请求 PDU**，其中参数‘报文分段’＝TRUE，但是该设备不支持分段报文的接收，

则产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个 **BACnet 中止 PDU**，其中参数‘服务器’＝TRUE、‘放弃原因（abort-reason）’＝**不支持分段（SEGMENTATION_NOT_SUPPORTED）**，然后进入**空闲**状态。

接收到分段证实请求 PDU

如果设备从网络层接收到一个 **BACnet 有证实请求 PDU**，其中参数‘报文分段’＝TRUE、‘序号’＝0，且该设备支持分段报文的接收，

则根据 **BACnet 有证实请求 PDU** 中的‘预设窗口尺寸’参数值以及设备自身情况，计算变量**实际窗口尺寸**的取值，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个 **BACnet 分段确认 PDU**，其中参数‘否定确认’＝FALSE、‘服务器’＝TRUE、‘实际窗口尺寸’＝**实际窗口尺寸**值，然后启动**分段计时器**，设置**最近序号**＝0、**初始序号**＝0，最后进入**分段请求**状态，准备接收后续分段。

（注：**实际窗口尺寸**的值由开发者自行确定，只与本地设备有关，但是这个值的取值限制在 1 到 127 范围内，并且实际取值不能大于接收到的 **BACnet 有证实请求 PDU** 报文中的‘预设窗口尺寸’参数值。）

接收到中止 PDU

如果设备从网络层接收到一个 **BACnet 中止 PDU** 报文，其中参数‘服务器’＝FALSE，

则进入**空闲**状态。

接收到意外 PDU

如果设备从网络层接收到一个意外的 PDU，例如一个参数‘报文分段’＝TRUE、‘序号’＝0 的 **BACnet 有证实请求 PDU**，或者一个参数‘服务器’＝FALSE 的 **BACnet 分段确认 PDU**，

则产生参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝TRUE 的 **BACnet 中止 PDU** 报文，然后进入**空闲**状态。

5.4.5.2 分段请求状态

在**分段请求**状态，设备等待接收 **BACnet 有证实请求 PDU** 的分段。

接收到新的分段

如果设备从网络层接收到一个 **BACnet 有证实请求 PDU**，其中‘报文分段’＝TRUE、‘后继’＝TRUE、‘序号’＝(**最近序号**+1)模 256、‘序号’≠(**初始序号**+**实际窗口尺寸**)模 256，

则保存此 **BACnet 有证实请求 PDU** 分段，将变量**最近序号**自加 1，模 256，重启**分段计时器**，然后进入**分段请求**状态，等待接收后续的分段。

接收到一组分段中的最后分段

如果设备从网络层接收到一个 **BACnet 有证实请求 PDU** 报文，其中‘报文分段’＝TRUE、‘后继’＝TRUE、‘序号’＝(**最近序号**+1)模 256、‘序号’＝(**初始序号+实际窗口尺寸**)模 256，

则保存此 **BACnet 有证实请求 PDU** 分段，将变量**最近序号**自加 1，模 256，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个 **BACnet 分段确认 PDU**，其中参数‘否定确认’＝FALSE、‘服务器’＝TRUE、‘序号’＝**最近序号**值、‘实际窗口尺寸’＝**实际窗口尺寸**值，然后重启**分段计时器**，设置变量**初始序号**＝变量**最近序号**，最后进入**分段请求**状态，等待接收后续的分段。

接收到报文最后分组

如果设备从网络层接收到一个 **BACnet 有证实请求 PDU** 报文，其中参数‘报文分段’＝TRUE、‘序号’＝(**最近序号**+1)模 256、‘后继’＝FALSE（即最后一个分段），

则保存此 **BACnet 有证实请求 PDU** 分段，将变量**最近序号**自加 1，模 256，停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个 **BACnet 分段确认 PDU**，其中参数‘否定确认’＝FALSE、‘服务器’＝TRUE、‘序号’＝变量**最近序号**值、‘实际窗口尺寸’＝变量**实际窗口尺寸**值，然后设置变量**初**

始序号＝变量**最近序号**值，向本地应用程序发送一个 CONF_SERV.indication(+)原语，在该原语中，包含了接收到的所有分段，最后进入**等待响应**状态。

接收到乱序分段

如果设备从网络层接收到一个 **BACnet 有证实请求 PDU**，其中‘报文分段’＝TRUE、‘序号’≠(**最近序号**+1)模 256，

则丢弃此 PDU，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一**个 BACnet 分段确认 PDU**，其中‘否定确认’＝TRUE、‘服务器’＝TRUE、‘序号’＝变量**最近序号**的值、‘实际窗口尺寸＝变量**实际窗口尺寸**的值，然后重启**分段计时器**，设置变量**初始序号**＝变量**最近序号**，最后进入**分段请求**状态，等待接收后续分组。

接收到中止 PDU

如果设备从网络层接收到一个 **BACnet 中止 PDU** 报文，其中参数‘服务器’＝FALSE，则停止**分段计时器**，然后进入**空闲**状态。

接收到意外 PDU

如果设备从网络层接收到一个意外的 PDU,例如一个参数‘报文分段’＝FALSE 的 **BACnet 有证实请求 PDU**、或者一个参数‘服务器’”＝FALSE 的 **BACnet 分段确认 PDU**，

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝TRUE 的 **BACnet 中止 PDU**，最后进入**空闲**状态。

分段超时

如果设备的**分段计时器**的值超过了 T_{seg} 的 4 倍，

则停止**分段计时器**，进入**空闲**状态。

发送中止 PDU

如果设备从本地应用程序接收到 ABORT.request 原语，

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝TRUE 的 **BACnet 中止 PDU**，然后进入**空闲**状态。

5.4.5.3 等待响应状态

在**等待响应**状态，设备等待本地应用程序针对 **BACnet 有证实请求 PDU** 所做出的响应。参见 9.8 节关于在 MS/TP 网络中的特殊考虑。

发送简单确认 PDU

如果设备从本地应用程序接收到一个 CONF_SERV.response(+)原语，且这个原语要通过一个 **BACnet 简单确认 PDU** 进行传递，

则产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个包含了这个 CONF_SERV.confirm(+)信息的 **BACnet 简单确认 PDU**，然后进入**空闲**状态。

发送不分段复杂确认 PDU

如果设备从本地应用程序接收到一个 CONF_SERV.response(+)原语，且这个原语要通过一个 **BACnet 复杂确认 PDU** 进行传递，而且这个 APDU 的长度不大于 5.2.1 节所确定的最大可传输长度值，

则产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘报文分段’＝FALSE 的 **BACnet 复杂确认 PDU**，然后进入**空闲**状态。

无法发送分段复杂确认 PDU（CannotSendSegmentedComplexACK）

如果设备从本地应用程序接收到一个 CONF_SERV.response(+)原语，且这个原语要通过一个 **BACnet 复杂确认 PDU** 进行传递，而且这个 APDU 的长度大于 5.2.1 节所确定的最大可传输长度值，如果同时还存在以下任何一种情况：

(a) 此设备不支持分段报文的发送，或者

(b) 客户端设备不能接收分段的响应（当客户端设备发来的 **BACnet 有证实请求 PDU** 中参数‘可接收分段响应（segmented-response-accepted）’＝FALSE 时），

则产生参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个 **BACnet 中止 PDU**，其中参数‘服务器’＝TRUE、‘放弃原因’＝**不支持分段**，最后进入**空闲**状态。

发送分段复杂确认 PDU

如果设备从本地应用程序接收到一个 CONF_SERV.response(+)原语，且这个原语要通过一**个 BACnet 复杂确认 PDU** 进行传递，而且这个 APDU 的长度大于 5.2.1 节所确定的最大可传输长度值，同时该设备支持分段报文的发送，而且客户端设备也能够接收分段的响应（当客户端设备发来的 **BACnet 有证实请求 PDU** 中参数‘可接收分段响应’＝TRUE 时），

则设置参数**分段重发计数**＝0、**初始序号**＝0、**预设窗口尺寸**＝所希望的值、**实际窗口尺寸**＝1，启动**分段计时器**，产生一个参数‘期待回复数据’＝TRUE 的 N-UNITDATA.request 报文，用来传送一个包含了这个报文第一个分段的 **BACnet 复杂确认 PDU**，其中参数‘报文分段’＝TRUE、‘后继’＝TURE、‘序号’＝0、‘预设窗口尺寸’＝变量**预设窗口尺寸**值，最后进入**分段请求**状态，等待接收确认。

发送差错 PDU

如果设备从本地应用程序接收到一个 CONF_SERV.response(-)原语，

则产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个 **BACnet 差错 PDU**，然后进入**空闲**状态。

发送中止 PDU

如果设备从本地应用程序接收到 ABORT.request 原语，

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝TRUE 的 **BACnet 中止 PDU**，然后进入**空闲**状态。

发送拒绝 PDU

如果设备从本地应用程序接收到 REJECT.request 原语，

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个 **BACnet 拒绝 PDU**，然后进入**空闲**状态。

接收到中止 PDU

如果设备从网络层接收到一个 **BACnet 中止 PDU**，其中参数‘服务器’＝FALSE，

则向本地应用程序发送 ABORT.indication 原语，然后进入**空闲**状态。

接收到重复的请求 PDU

如果设备从网络层接收到一个 **BACnet 有证实请求 PDU**，其中参数‘报文分段’＝FALSE，

则将这个 PDU 视为重复请求而丢弃掉，然后重新返回当前状态。

接收到重复分段 PDU

如果设备从网络层接收到一个 **BACnet 有证实请求 PDU**，其中参数‘报文分段’＝TRUE，

则将这个 PDU 视为重复分段而丢弃掉，并且产生一个参数‘期待回复数据’＝FALSE

的 N-UNITDATA.request 报文，用来传送一个 **BACnet 分段确认 PDU**，其中参数‘否定确认’＝FALSE、‘服务器’＝TRUE、‘序号’＝变量**最近序号**值、‘实际窗口尺寸’＝变量**实际窗口尺寸**的值，然后重新返回当前状态。

接收到意外 PDU

如果设备从网络层接收到一个意外的 PDU，例如一个参数‘服务器’＝FALSE 的 **BACnet 分段确认 PDU**，

则产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝TRUE 的 **BACnet 中止 PDU**，然后向本地应用程序发送 ABORT.indication 原语，最后进入**空闲**状态。

分段超时

如果设备的**请求计时器**的值超过了 T_{out}，

则产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝TRUE 的 **BACnet 中止 PDU**，然后向本地应用程序发送 ABORT.indication 原语，最后进入**空闲**状态。

5.4.5.4 分段响应状态

在**分段响应**状态，设备等待接收一个 **BACnet 分段确认 PDU**，此 PDU 用于对 **BACnet 复杂确认 PDU** 中的一个或几个分段进行确认。

接收到重复确认 PDU

如果设备从网络层接收到一个 **BACnet 分段确认 PDU** 报文，其中参数‘服务器’＝FALSE，而函数 InWindow(**BACnet 分段确认 PDU** 中的‘序号’参数值，**初始序号**)的返回值为 FALSE，

则重启**分段计时器**，进入**分段响应**状态，继续等待接收确认，直至超时。

接收到新确认 PDU

如果设备从网络层接收到一个 **BACnet 分段确认 PDU** 报文，其中参数‘服务器’＝FALSE，而函数 InWindow(**BACnet 分段确认 PDU** 中的‘序号’参数值，**初始序号**)的返回值为 TRUE，同时设备至少还有一个报文分段等待发送，

则设置变量**初始序号**＝(**BACnet 分段确认 PDU** 的‘序号’+1)模 256、变量**实际窗口尺寸**＝**BACnet 分段确认 PDU** 的‘实际窗口尺寸’参数值，然后重启**分段计时器**，设置**分段重发计数**＝0，调用函数 FillWindow(**初始序号**)，产生一个参数‘期待回复数据’

＝TRUE 的 N-UNITDATA.request 报文，用来传送一个或多个 **BACnet 证实请求 PDU**，在这些 PDU 中，恰好包含了后续的**实际窗口尺寸**个报文分段，最后进入**分段响应**状态，等待新的确认。

接收到最后一个确认 PDU（FinalACK_Received）

如果设备从网络层接收到一个 **BACnet 分段确认 PDU** 报文,其中参数‘服务器’＝FALSE，同时函数 InWindow(**BACnet 分段确认 PDU** 中的‘序号’参数值，**初始序号**)的返回值为 TRUE，并且已没有分段需要发送，

则停止**分段计时器**，进入**空闲**状态。

分段超时

如果设备的**分段计时器**的值超过了 T_{seg}，但**分段重发计数**又小于 N_{retry}，

则将**分段重发计数**自加 1，重启**分段计时器**，调用函数 FillWindow(**初始序号**)，产生一个参数‘期待回复数据’＝TRUE 的 N-UNITDATA.request 报文，用来传送一个或多个 **BACnet 有证实请求 PDU**，在这些 PDU 中，恰好包含了后续的**实际窗口尺寸**个报文分段，最后进入**分段响应**状态，重新等待确认。

最终超时

如果设备的**分段计时器**的值超过**了** T_{seg}，**分段重发计数**的值超过 **APDU 重发次数**，

则停止**分段计时器**，然后进入**空闲**状态。

接收到中止 PDU

如果设备从网络层接收到一个 **BACnet 中止 PDU** 报文，其中参数‘服务器’＝FALSE，

则停止**分段计时器**，向本地应用程序发送 ABORT.indication 原语，然后进入**空闲**状态。

接收到意外 PDU

如果设备从网络层接收到一个意外的 PDU，例如一个 **BACnet 证实请求 PDU**，

则停止**分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝TRUE 的 **BACnet 中止 PDU**，然后进入**空闲**状态。

发送放弃 PDU

如果设备从本地应用程序接收到 ABORT.request 原语，

则**停止分段计时器**，产生一个参数‘期待回复数据’＝FALSE 的 N-UNITDATA.request 报文，用来传送一个参数‘服务器’＝TRUE 的 **BACnet 中止 PDU**，然后进入**空闲**状态。

5.5 应用层协议时序图

各种服务原语具体传递的过程，可以通过时序图（参见图 5-4 至 5-13）来说明。每个时序图可以分为三个或者四个区域。标注为“提供方”的区域代表了服务提供者，标注为“用户”的区域代表了服务使用者。如果还存在第四个区域，则代表的是应用程序。其中位于“提供者”与“用户”之间的竖线，对于应用层而言，指的是 **BACnet 用户元素**与 **BACnet 应用层服务元素**之间的接口。对于下面各层而言，这些竖线表示的则是处于服务提供者与服务使用者之间的服务访问点（service-access-point）。在图中，由上到下是时间流逝的方向。处于服务使用者区域中的箭头，指明了在事务处理过程中由服务原语所表示的信息的流动方向（即从服务使用者流入或流出）。图 5-4 到图 5-13 示出了 BACnet 中定义的各种应用层服务原语传递过程。

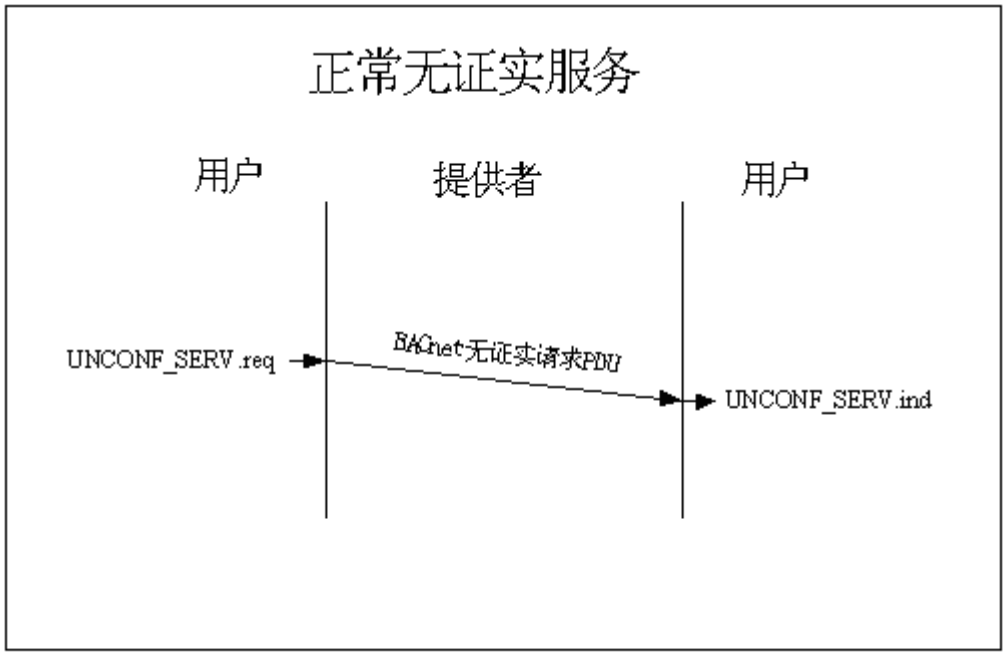


图 5-4. 正常无证实服务时序图

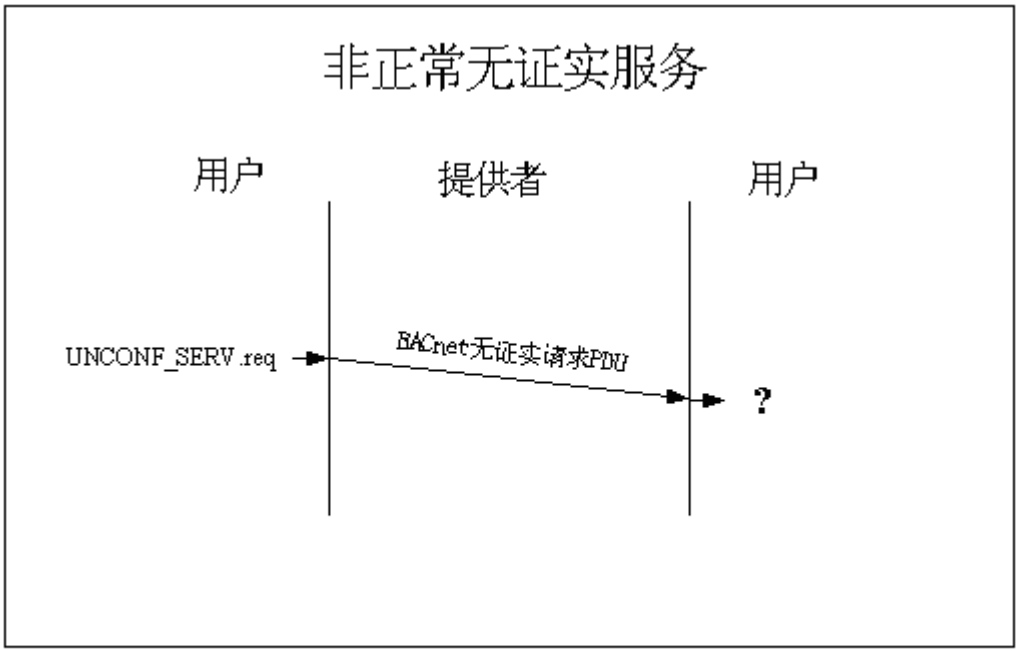


图 5-5. 非正常无证实服务时序图。无证实服务请求报文出现某些差错，被接收用户丢弃，如图中“？”所示。

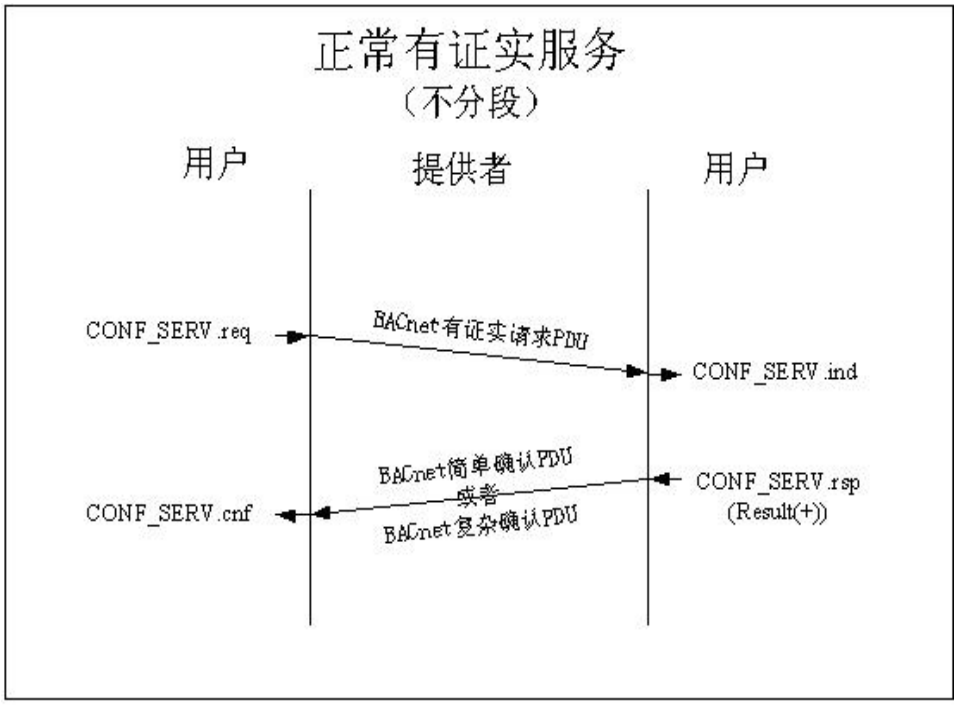


图 5-6. 正常有证实服务（不分段）时序图

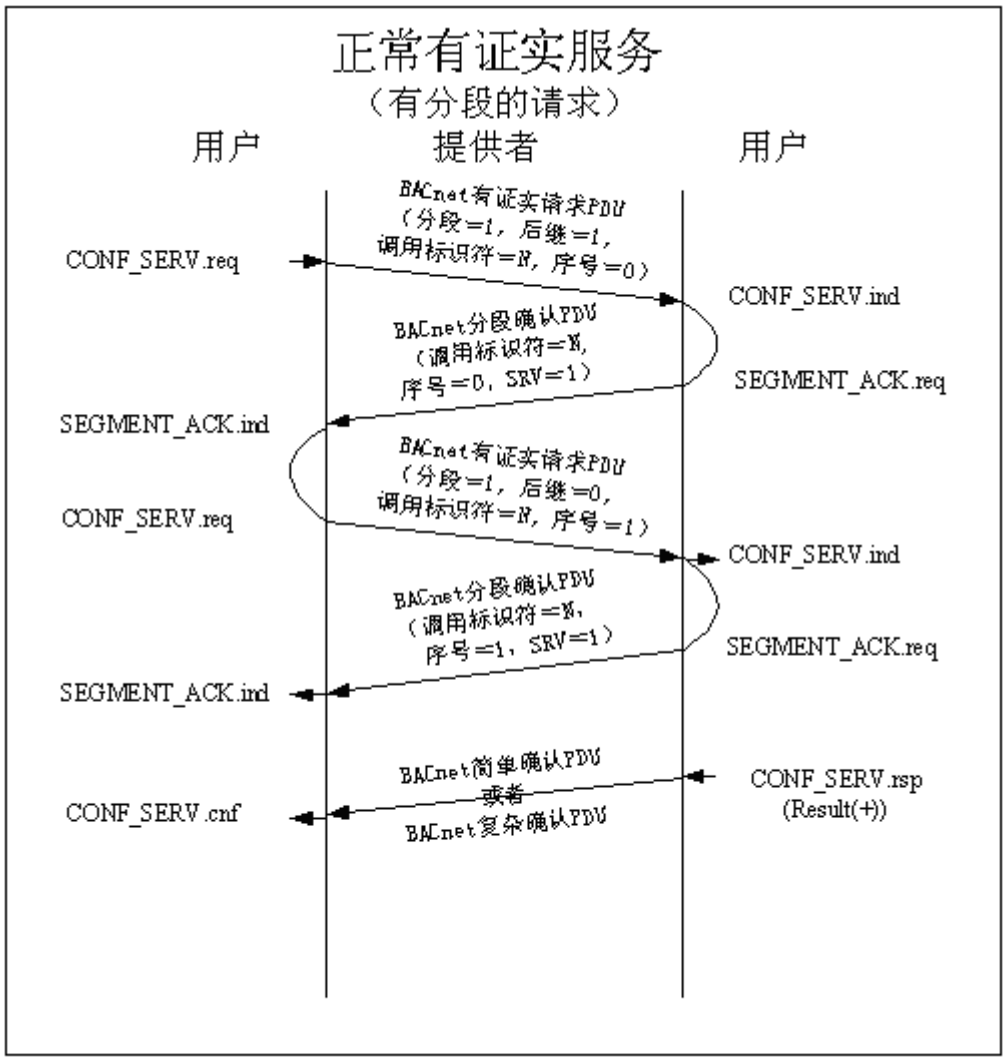


图 5-7. 具有分段请求的正常有证实服务时序图

图 5-7 举例说明了两个分开的、交错的服务原语的交换。一个交换是使用有证实的服务请求、指示、响应以及证实。有些请求需要分段，因此需要使用几个 CONF_SERV.request 原语来传递整个请求。分段确认服务原语是另一个独立的交换，它用来通知客户，服务器已准备接收下一分段了。

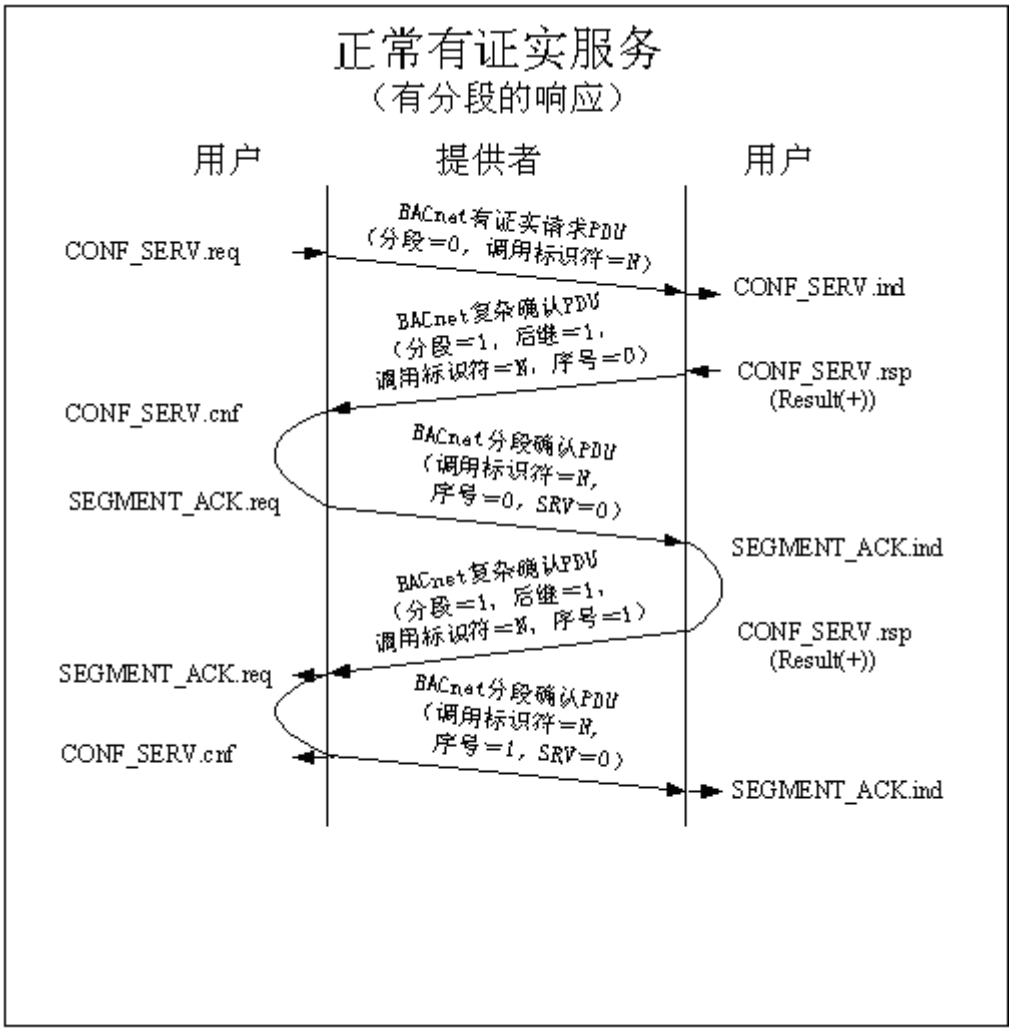


图 5-8. 具有分段响应的正常有证实服务时序图

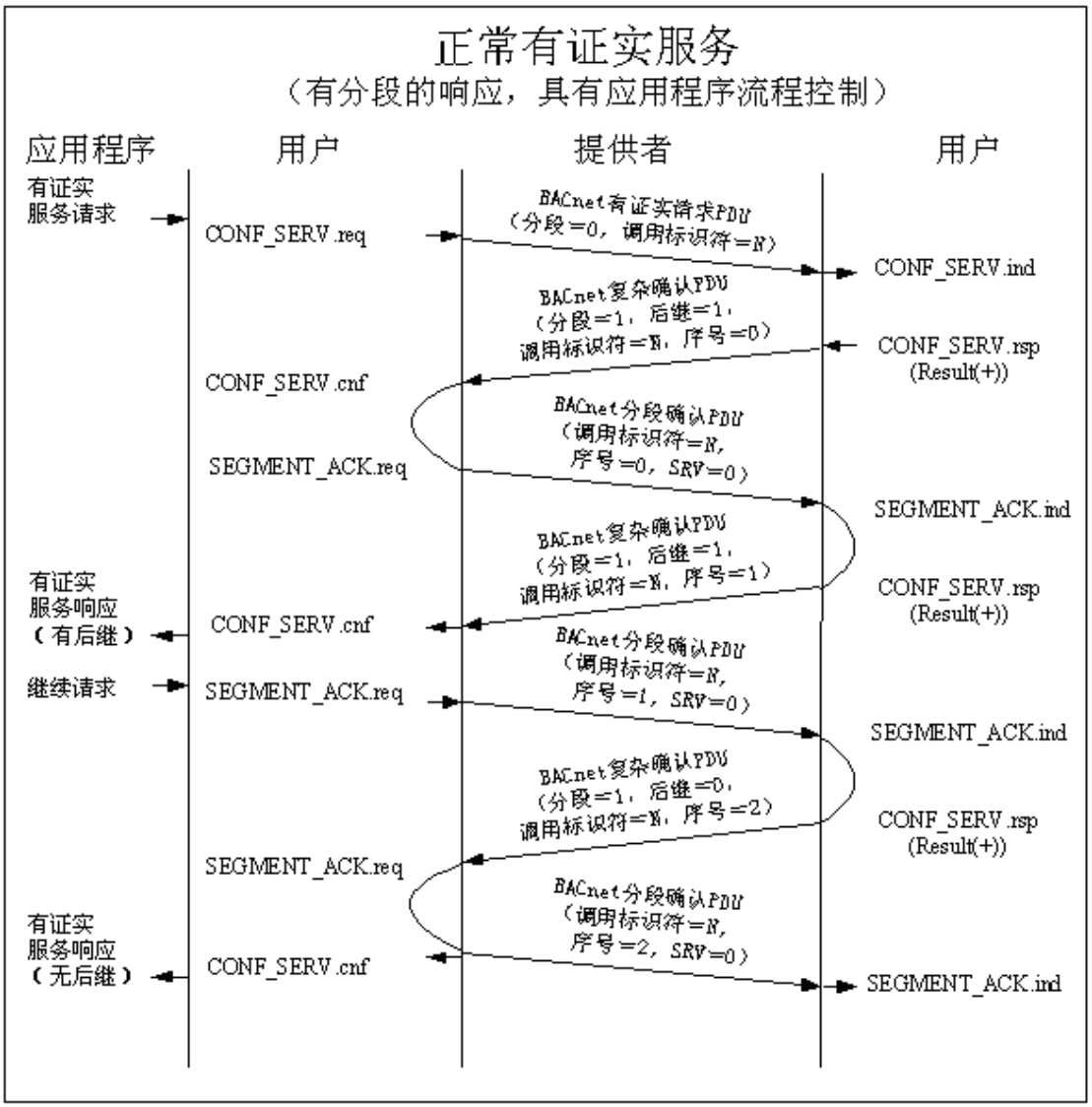


图 5-9. 具有应用程序流程控制的正常有证实服务时序图

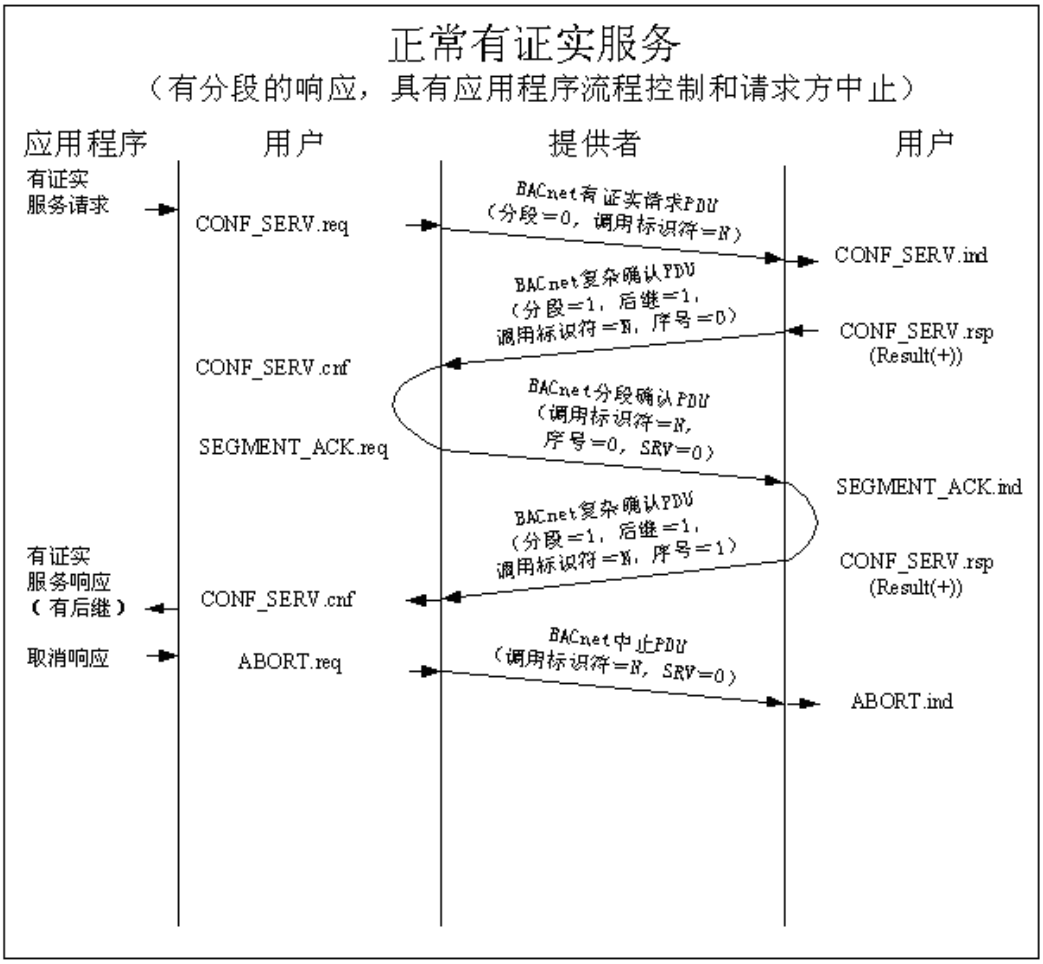


图 5-10. 具有分段请求、应用程序流程控制和取消响应的正常有证实服务时序图

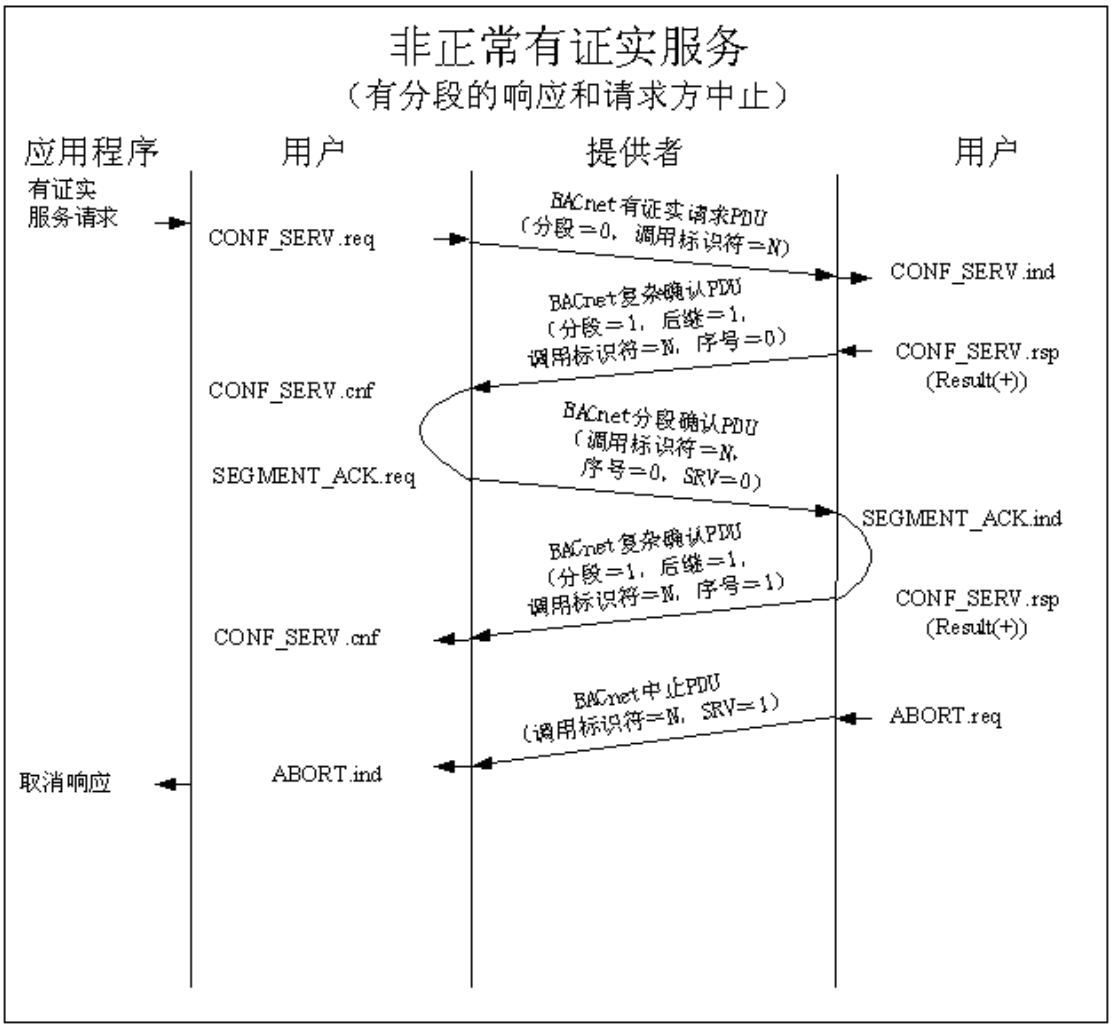


图 5-11．非正常有证实服务时序图

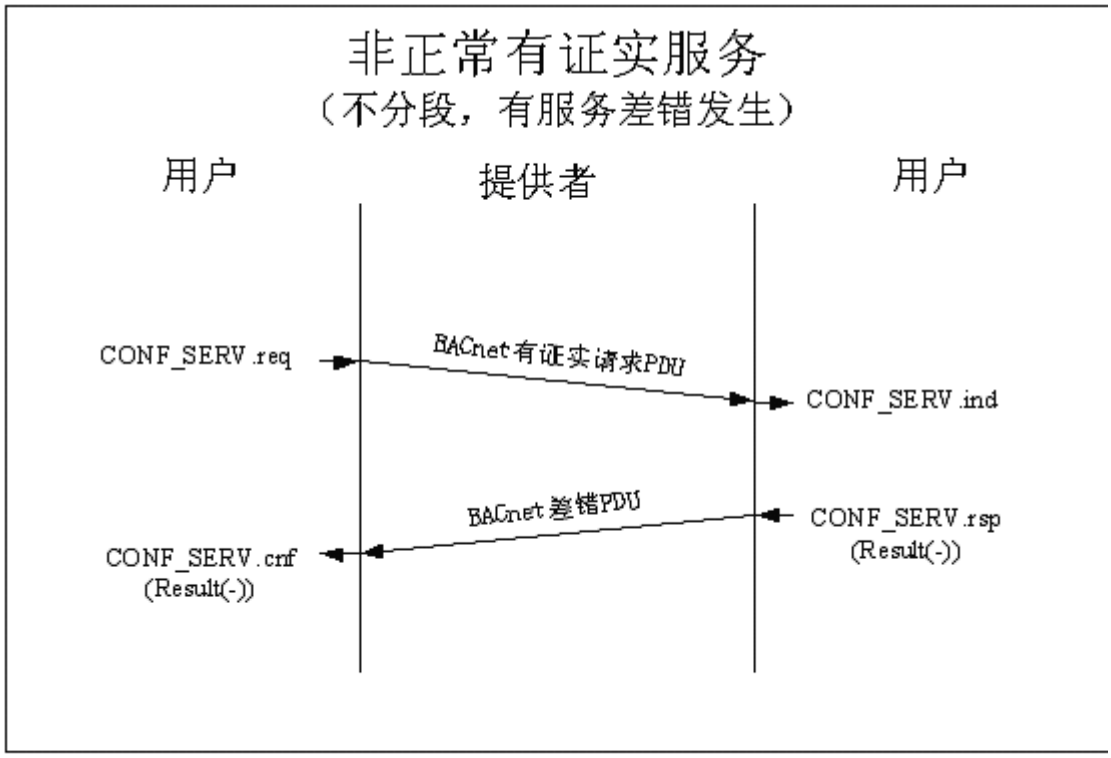


图 5-12. 非正常有证实服务（不分段，有差错发生）时序图

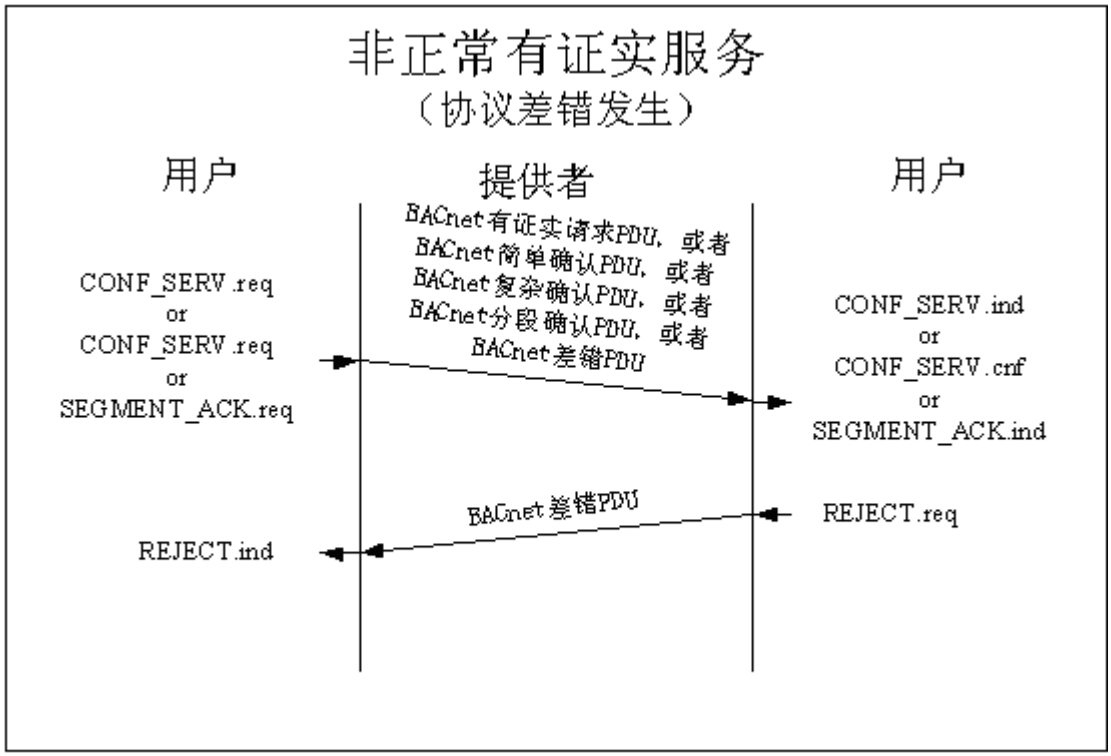


图 5-13. 具有协议错误的非正常有证实服务请求或响应时序图

5.6 应用层服务约定用法

本标准中所使用的描述术语和符号用法遵循 ISO TR 8509,这是关于服务约定用法的 OSI 技术报告。在此 OSI 约定用法中，定义了协议服务提供者与协议服务用户之间的交互。两者之间的信息交流，也被抽象成一种“服务原语”的交换。服务原语是功能规范与用户层交互的抽象表示，这些抽象定义并不包括服务用户/服务提供者交互的具体实现细节。每个原语一般包含有一组参数，这些参数代表要传递的数据元素，它们被传递给由原语所调用的可用函数。参数表示服务用户/服务提供者交互中可用的信息，在任何特定的界面上，某些参数或者被明确地定义（尽管可能没有明确地在原语中定义），或者隐含地与服务访问点相关联。同样，在任何特定的协议规范中，与一个服务原语相对应的函数也或者被明确地定义，或者隐含地说明为可用的。

本协议的第 13 节到第 17 节以及第 24 节用列表的形式,明确地描述了 BACnet 服务原语中的参数。每个表由 5 列组成，一列是服务参数名称，其它四列分别对应请求“Req”、指示“Ind”、响应“Rsp”和证实“Cnf”原语。对于非证实服务，“Req”与“Cnf”所对应的列是空缺的。表的每一行包含了一个参数或子参数，同时在每种服务原语所在的列，都用一个符号指出该参数在这种服务原语中的使用类型。这些符号按 ISO 关于约定用法方面的技术报告（ISO TR 8509）的建议，命名如下：

M—对于此原语，该参数是必需的。

U—此参数是用户选择的，可以不提供。

C—此参数是有条件的（Conditional），它取决于其它一些参数。

S—此参数是个选择参数，即从多个可能的选择参数集合中选取一个。组成选择参数集合的参数在表中按如下方式表示：

(a) 选择参数集合中的每个参数，用符号“S”表示；

(b) 表中属于同一选择参数集合的每个参数名称,位于参数列中相同的缩格位置；

(c) 取下列两选择之一

1. 每个参数名称位于表中的最左端；
2. 每个参数分别属于不同的参数组，每个参数组中的成员参数名称又具有相同的缩格位置。所谓参数组，指的是所有具有同一父参数的参数所构成的集合。在表中，父参数位于该参数组所有成员参数之上，并且在水平位置上没有成员参数的缩格。在下面的例子中，参数 A 与参数 B 组成了一个参数组：

参数 X

参数 A

参数 B

参数 Y

参数 C

对于同一选择参数集合所包含的参数，在表中用相同的缩格来表示其属于该参数组。也就是说，位于同一“高层”参数且具有相同水平缩格的参数，都属于同一选择集合。

在符号 M、U、C 或 S 后跟着有符号“(=)”，这表示此列的服务原语中的参数与在(=)左边的字母单独所在列的服务原语中的参数，在语义上是相等的。例如，在指示原语列中符号为“M(=)”，而“M”单独在请求原语列中，这就表示指示原语中的参数与请求原语中对应的参数，语义上是相等的。

某些参数可能还包含子参数，这些子参数也同样用缩格来标注。子参数是否出现，总是取决于父参数是否出现。比如在上面的例子中，参数 A 与参数 B 都是参数 X 的子参数，而参数 C 是参数 Y 的子参数。如果参数 X 是可选的，在某个服务原语中不提供时，那么其子参数（参数 A 与参数 B）也将不提供。

某些服务参数命名采用了“List of ...”的约定用法。若没有特别说明，这些参数指定了由跟在“List of …”关键词后的若干个项目（item）所构成的列表。

12. 控制设备的对象模型

不同的设备具有不同的存储信息的数据结构。为了应用本协议实现设备间的信息交换，必须定义一种标准的、“网络可见” 的信息描述方式。为实现这种网络可见的描述方式，采用的方法是面向对象的方法。本节定义一组标准的对象类型，这组标准对象类型给出了一种抽象的数据结构，为建立应用层服务提供一个框架。大部分应用层服务设计成为对这些标准对象类型的属性进行访问和操作。在不同的设备中，把应用层服务功能映射为对物理设备实际数据结构的作用有不同的方式，并且设备所能支持的某种对象类型的实例数也是可以不同的，具体的实现方式由生产商自行确定。

所有对象均由**对象标识符**属性所引用。每个 BACnet 设备的对象均有一个唯一的**对象标识符**属性值。将一个对象的**对象标识符**与具有系统全局唯一性质的 **BACnet 设备对象标识符**结合使用，就提供了一种在整个控制网络中引用每个对象的机制。

本标准并不要求设备支持在本标准中定义的所有对象类型才算符合标准。根据第 2332 节有关设备一致性类别的定义，属于不同类别的设备所必须支持的对象类型是不同的。另外，对象类型的属性是可选的。在每个对象类型规范的开始，是其属性的列表。列表项包括属性标识符、属性数据类型和 O、W、R 之一的属性一致性代码。

其中，O： 表示属性是可选的，

R： 表示属性是必需的，并且是 BACnet 服务可读的，

W：表示属性是必需的，并且是 BACnet 服务可读、可写的。

3

当属性是必需的，即一致性代码为 R 时，表示在所有的 BACnet 标准对象中这个属性都是必需的。当属性是可选的，即一致性代码为 O 时，表示这个属性不是必需的。R 或 O 类型的属性值可以通过本标准定义的一个或多个**读属性（ReadProperty）**服务读取。除非本标准明文禁止，否则 R 或 O 类型的属性在不同的实现方式中都可以是可写的。当属性为可写的，即一致性代码为 W 时，表示这个属性在所有的 BACnet 标准对象中都是必需的，并可以通过本标准定义的一个或多个**写属性（WriteProperty）**服务来写入。W 类型的属性值也可以使用一个或多个**读属性**服务读取。对于 O 类型，如果存在于某个对象中，除非本标准明文规定是可写的，否则不必是可写的。

在某些设备中，属性值在内部以不同的数据类型存贮，而不是按照属性的数据类型存贮。例如，实数可能在内部被存为整型。这种情况可能导致的结果是，属性值被**写属性**服务修改后，紧接着读出返回的值会略有不同。但只要在存储这个修改的数据时作了“最大努力（best effort）”，这种情况是可以接受的。

在本标准中定义的对象类型及其属性已经非常全面，但在实现标准时仍然允许自由定义附加的非标准对象类型或标准对象类型的非标准属性。随着控制技术的发展，这种方法是扩充标准的主要方法。创新的修改可以立即被采纳，而不必等待本标准的正式修改版本。这种

易于扩充的能力可以把本标准应用到其它的楼宇设备领域。参见第 23.3 节和第 23.4 节。

非标准对象类型必需支持下列属性：

对象标识符（Object_Identifier） BACnetObjectIdentifier

对象名称（Object_Name） CharacterString

对象类型（Object_Type） BACnetObjectType

当这些属性实现时，其行为应与在标准 BACnet 对象中的行为一样。也就是说，**对象标识符**和**对象名称**属性在拥有它们的 BACnet 设备应是唯一的。**对象名称**字符串至少为一个字符，并且应仅由可打印字符组成。

BACnet 标准对象应支持标准中规定的所有必需属性。除此之外，还可以支持标准中规定的可选属性或标准中未定义的属性。对象类型中的必需属性在这种类型的所有对象中的功能都必须是本标准所规范的功能。如果实现的对象支持本标准中的可选属性，则可选属性的功能也必须是本标准所规范的功能。必需属性必须在该类型的所有对象中出现。如果某一对象中有可选属性，则该类型的所有对象不必均有可选属性。一个属性，不论是必需的，还是可选的，都应返回本标准中规定的数据类型，除了明文规定之外，并不要求属性的返回值能够返回数据类型的所有数值。在对象中，对于所支持的属性，不论是必需的，还是可选的，如果读操作或者写操作限制了属性值的取值范围，使得不能返回数据类型的所有数值，则要在协议实现一致性声明（PICS）中对每个有这种限制的属性进行明确规定。

某些 BACnet 对象的某些属性必须用于表示同一类型数据元素的集合，而不是一个简单数据值或者由其它数据类型构造而成的复合数据类型。在某些对象实例中，这种数据元素集合的个数是固定的，而在另一些实例中，数据元素集合的个数是变化的。在有些情况下，数据元素集合中的元素是可以单个访问的，因此，数据元素的顺序是很重要的。BACnet 为表示同类型数据元素集合，提供两种形式的属性数据类型：“**BACnet 数组（BACnetARRAY）**”和“**列表（List of）**”。

“**BACnet 数组**”数据类型是一个结构化的数据类型，由一组有序的数据元素序列构成，这些数据元素都具有相同的数据类型。属性数组中的元素可以通过“数组索引”（无符号整型值）单个访问（读取或写入）。索引号为 0 的元素将返回数组元素的个数。忽略索引号则访问数组的所有元素。索引号 N（大于 0）表示该数组序列中的第 N 个元素。当数组属性用在 BACnet 对象中时，“**BACnet 数组[N]数据类型（BACnetARRAY [N] of datetype）**”这个标记表示 N 个数据元素组成的有序序列，其中的每个元素都有相同的数据类型。

“**列表**”数据类型是结构化的数据类型，由一组（0 个或多个）数据元素序列构成，每个元素都有相同的数据类型。每个“**列表**”类型的长度是可变的，除了明确规定为某一特殊用途而有确定的长度要求之外，一般“**列表**”类型不对最大元素数目进行限制。“**列表类型（List of datatype）**”这个标记表示 0 个或多个数据元素组成的序列，其中的每个元素都有相同的数据类型。

“**BACnet 数组**”和“**列表**”的区别在于，数组中的元素可以通过索引唯一地访问，而

“**列表**”中的元素不能通过索引访问。并且“**BACnet 数组**”中元素数目可以通过使用索引为 0 而确定，而“**列表**”中元素数目只能通过读取所有属性并通过计算确定。

在本节中定义的几个对象类型：**命令（Command）**、**事件登记（Event Enrollment）**、**组（Group）**、**环（Loop）**、和**时间表（Schedule）**都有一个或多个类型为 **BACnet 对象属性引用（BACnetObjectPropertyReference）**的属性。这些引用的属性标识符成员可以不是**全部的（ALL）**、**必需的（REQUIRED）**或**可选的（OPTIONAL）**这些特别属性标识。这些属性标识保留在**条件读属性（ReadPropertyConditional）**和**读多个属性（ReadPropertyMultiple）**服务中或者标准中未定义的服务中。

本节中定义的对象类型中，有几个对象类型有被称为“**可靠性（Reliability）**”的属性。这个属性是枚举数据类型，对于不同的对象类型有不同的属性值。下面列举的数值是所有对象类型的**可靠性**属性的超集。可能的取值范围在相应的对象类型定义中有描述。

未发现故障（NO_FAULT_DETECTED） 当前值是可靠的，即未发现如下列举的故障。

无传感器（NO_SENSOR） 没有与输入对象相对应的传感器。

超出范围（OVER_RANGE） 映射为输入对象的传感器读取值超出正常值。如果是二进制输入对象，当其状态由模拟传感器或装有电子循环监测管理电路的二进制输入设备派生时，可能会发生这种情况。

低于范围（UNDER_RANGE） 映射为输入对象的传感器读取值小于正常值。如果是二进制输入对象，当其状态只是从模拟传感器计算得到时，可能会发生这种情况。

开路（OPEN_LOOP） 映射为定义对象的物理设备产生表示开路的数值。

短路（SHORTED_LOOP） 映射为定义对象的物理设备产生表示短路的数值。

无输出（NO_OUTPUT） 没有与输出对象相对应的物理设备。

其它不可靠（UNRELIABLE_OTHER） 控制器探测到当前值是不可靠的，而又不能确定问题的原因。此时，除上述的原因外，所有其它错误原因均返回此值，如：二进制输入对象没有按预期循环操作。

12.1 模拟输入对象类型（Analog Input Object Type）

模拟输入对象类型定义为一个标准对象，其属性表示一个模拟输入的外部可见一致性代码。这个对象及其属性见表 12-1，本节进行详细规范。

表 12-1 模拟输入对象的属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R

对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	实数 REAL	R[1]
描述 Description	字符串 CharacterString	O
设备类型 Device_Type	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R
事件状态 Event_State	BACnet 事件状态 BACnet EventState	R
可靠性 Reliability	BACnet 可靠性 BACnetReliability	O
脱离服务 Out_Of_Service	布尔 BOOLEAN	R
更新间隔 Update_Interval	无符号整型 Unsigned	O
单位 Units	BACnet 工程单位 BACnetEngineeringUnits	R
最小值 Min_Pres_Value	实数 REAL	O
最大值 Max_Pres_Value	实数 REAL	O
分辨率 Resolution	实数 REAL	O
COV 增量 COV_Increment	实数 REAL	O[2]
时间延迟 Time_Delay	无符号整型 Unsigned	O[3]
通告类 Notification_Class	无符号整型 Unsigned	O[3]
高阈值 High_Limit	实数 REAL	O[3]
低阈值 Low_Limit	实数 REAL	O[3]
阈值宽度 Deadband	实数 REAL	O[3]
阈值使能 Limit_Enable	BACnet 阈值使能 BACnetLimitEnable	O[3]
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	O[3]
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	O[3]
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	O[3]

[1] 当**脱离服务**为 TRUE 时该属性必需是可写的。

[2] 如果对象支持 COV 报告则该属性是必需的。

[3] 如果对象支持内部报告则该属性是必需的。

12.1.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.1.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 BACnet **设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.1.3 对象类型属性

这个属性的类型是 BACnet **对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**模拟输入（ANALOG_INPUT）**。

12.1.4 当前值属性

这个属性的类型是**实数**类型，以工程单位表示输入测量的当前值（见 12.1.12 节）。当**脱离服务**为 TRUE 时，**当前值**属性是可写的。

12.1.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.1.6 设备类型属性

这个属性的类型是**字符串**类型，是一段对映射为这个模拟输入对象的物理设备的文本描述。通常用于描述与模拟输入对象对应的传感器型号。

12.1.7 状态标志属性

这个属性的类型是 BACnet **状态标志**类型，表示四种布尔标志，这些标志表示模拟输入设备运行时刻的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中，故障，管制，脱离服务**}

其中，

报警中（IN_ALARM） 如果**事件状态**（见 12.1.8 节）属性值为**正常（NORMAL）**则为 FALSE(0)，否则为 TRUE(1) 。

故障（FAULT） 如果**可靠性**（见 12.1.9 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制（OVERRIDDEN） 如果某值被与 BACnet **设备**本身的有关机制所管制，则为 TRUE(1)。在此情况下，“管制”表示**当前值**和**可靠性**属性值不再随物理设备的输入变化而变化。

脱离服务（OUT_OF_SERVICE） 如果**脱离服务（Out_Of_Service）**属性值（见 12.1.10 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.1.8 事件状态属性

这个属性的类型是 **BACnet 事件状态**类型，用于检测对象是否处于事件活动状态。如果对象支持内部报告，这个属性表示对象的事件状态。如果对象不支持内部报告，这个属性的值为**正常**。

12.1.9 可靠性属性

这个属性的类型是 **BACnet 可靠性**类型，表示**当前值**或物理输入设备的运行是否“可靠”，如果不可靠，则指明原因。这个属性有下列值：

{**未发现故障，无传感器，超出范围，低于范围，开路，短路，其它不可靠**}

12.1.10 脱离服务属性

这个属性的类型是**布尔**类型，表示该对象代表的物理输入是（TRUE）否（FALSE）不与设备相关。当**脱离服务**为 TRUE 时，**当前值**属性与物理输入设备分离，并不会随着物理输入设备的改变而变化。当**脱离服务**为 TRUE 时，**可靠性**属性和对应的**状态标志**属性中的**故障**标志的状态也与物理输入设备分离。当**脱离服务**为 TRUE 时，当用于模拟专门的固定条件或用于测试时，**当前值**与**可靠性**属性可以为任何值。而其它依靠**当前值**或**可靠性**属性的功能将响应这些变化，就像这些变化发生在输入中一样。

12.1.11 更新间隔属性

这个属性的类型是**无符号整型**类型，表示在正常运行时两次正常更新**当前值**之间的最大时间间隔（以百分之一秒为单位）。

12.1.12 单位属性

这个属性的类型是 **BACnet 工程单位**类型，表示这个对象的测量单位。详见第 21 节中 BACnetEngineeringUnits ASN.1 部分，其中定义了一系列标准工程单位。

12.1.13 最小值属性

这个属性的类型是**实数**类型，表示**当前值**属性的最小可靠数值（用工程单位表示）。

12.1.14 最大值属性

这个属性的类型是**实数**类型，表示**当前值**属性的最大可靠数值（用工程单位表示）。

12.1.15 分辨率属性

这个属性的类型是**实数**类型，表示**当前值**属性中以工程单位可分辨的最小变化（只读）。

12.1.16 COV 增量属性

这个属性的类型是**实数**类型，定义**当前值**属性的最小改变值，这个最小改变值将导致向 COV 客户发布 **COV 通告（COVNotification）**。如果对象支持 COV 报告，则必须具备这个属性。

12.1.17 时间延迟属性

这个属性的类型是**无符号整型**类型，定义从**当前值**属性处于由**高阈值**和**低阈值**确定的范围之外开始，到生成一个**进入异常（TO_OFFNORMAL）**事件之间的最小时间间隔（以秒为单位），或者从**当前值**属性进入由**高阈值**和**低阈值**确定的范围（包括**阈值宽度**属性值确定的范围）之内时，到生成一个**进入正常（TO_NORMAL）**事件的最小时间间隔（以秒为单位）。如果对象支持内部报告则必须具备这个属性。

12.1.18 通告类属性

这个属性的类型是**无符号整型**类型，用于定义这个对象处理和生成事件通告时的通告类别。这个属性缺省引用有相同**通告类**属性值的**通告类**对象。如果对象支持内部报告则必须具备这个属性。

12.1.19 高阈值属性

这个属性的类型是**实数**类型，定义生成一个事件的**当前值**属性的上限值。如果对象支持内部报告则必须具备这个属性。

12.1.19.1 生成进入异常事件条件

进入异常事件在达到下列三个条件的情况下生成：

(a) **当前值**超过**高阈值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**高阈值使能（HighLimitEnable）**标志，

(c) **事件使能**属性设置为**进入异常**标志。

12.1.19.2 生成进入正常事件条件

一旦**当前值**超过**高阈值**之后，只有在**当前值**下降到低于(**高阈值 - 阈值宽度**)之后，才能生成**进入正常**事件。**进入正常**事件在达到下列三个条件的情况下生成：

(a) **当前值**下降到低于（**高阈值 - 阈值宽度**）之后一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**高阈值使能**标志，

(c) **事件使能**属性设置为**进入正常**标志。

12.1.20 低阈值属性

这个属性的类型是**实数**类型，定义生成一个事件的**当前值**属性的下限值。如果对象支持内部报告则必须具备这个属性。

12.1.20.1 生成进入异常事件条件

进入异常事件在达到下列三个条件的情况下生成：

(a) **当前值**低于**低阈值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**低阈值使能**标志，

(c) **事件使能**属性设置为**进入异常**标志。

12.1.20.2 生成进入正常事件条件

一旦**当前值**低于**低阈值**之后，只有在**当前值**上升到高于(**低阈值** + **阈值宽度**)之后，才能生成**进入正常**事件。**进入正常**事件在达到下列三个条件的情况下生成：

(a) **当前值**上升到高于（**低阈值** + **阈值宽度**）之后一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**低阈值使能**标志，

(c) **事件使能**属性设置为**进入正常**标志。

12.1.21 阈值宽度属性

这个属性的类型是**实数**类型，定义**高阈值**属性和**低阈值**属性之间的一个宽度范围值。如果要生成一个**进入正常**的事件，**当前值**属性值必须维持在这个范围之内。**进入正常**事件在达到下列五个条件的情况下生成：

(a) **当前值**低于（**高阈值** - **阈值宽度**），

(b) **当前值**高于（**低阈值** + **阈值宽度**），

(c) **当前值**维持在这个范围之内一段由**时间延迟**属性确定的最小时间间隔的时间，

(d) **阈值使能**属性设置为**高阈值使能**或者**低阈值使能**之一的标志，

(e) **事件使能**属性设置为**进入正常**标志。

如果对象支持内部报告则必须具备这个属性。

12.1.22 阈值使能属性

这个属性的类型是 BACnet **阈值使能**类型，用两个标志分别表示使能或者禁止对于**高阈值**异常事件和**低阈值**异常事件以及各自返回到正常事件的报告。如果对象支持内部报告则必

须具备这个属性。

12.1.23 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示使能或者禁止对于**进入异常**、**进入故障**（TO_FAULT）和**进入正常**事件的报告。在**模拟输入**对象环境中，向**高阈值**和**低阈值**的**事件状态**的转换称为“异常”事件。如果对象支持内部报告则必须具备这个属性。

12.1.24 确认转换属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。在**模拟输入**对象环境中，向**高阈值**和**低阈值**的**事件状态**的转换称为“异常”事件。这些标志将在相应事件出现的情况下被清除，并在下列任一条件下设置：

(a) 收到相应的确认；

(b) 如果**事件使能**属性中相应的标志未设置时，事件发生（即在这种情况下，不会产生事件通告，因此也不会产生确认）；

(c) 如果**事件使能**属性中设置了相应的标志，并且在通告类对象的**确认转换**属性中未设置相应标志时，事件发生（即无确认产生）。

如果对象支持内部报告则必须具备这个属性。

12.1.25 通告类型属性

这个属性的类型是 BACnet **通告类型**类型，表示对象生成的通告是**事件**（Events）还是**报警**（Alarms）。如果对象支持内部报告则必须具备这个属性。

12.2 模拟输出对象类型（Analog Output Object Type）

模拟输出对象类型定义为一个标准对象，其属性表示一个模拟输出的外部可见一致性代码。这个对象及其属性见表 12-2，本节进行详细规范。

表 12-2　模拟输出对象的属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	实数 REAL	W
描述 Description	字符串 CharacterString	O
设备类型 Device_Type	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R
事件状态 Event_State	BACnet 事件状态 BACnet EventState	R
可靠性 Reliability	BACnet 可靠性 BACnetReliability	O
脱离服务 Out_Of_Service	布尔 BOOLEAN	R
单位 Units	BACnet 工程单位 BACnetEngineeringUnits	R
最小值 Min_Pres_Value	实数 REAL	O
最大值 Max_Pres_Value	实数 REAL	O
分辨率 Resolution	实数 REAL	O

优先值数组 Priority_Array	BACnet 优先值数组 BACnetPriorityArray	R
释放缺省 Relinquish_Default	实数 REAL	R
COV 增量 COV_Increment	实数 REAL	O[1]
时间延迟 Time_Delay	无符号整型 Unsigned	O[2]
通告类 Notification_Class	无符号整型 Unsigned	O[2]
高阈值 High_Limit	实数 REAL	O[2]
低阈值 Low_Limit	实数 REAL	O[2]
阈值宽度 Deadband	实数 REAL	O[2]
阈值使能 Limit_Enable	BACnet 阈值使能 BACnetLimitEnable	O[2]
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	O[2]
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	O[2]
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	O[2]

[1] 如果对象支持 COV 报告则这个属性是必需的。

[2] 如果对象支持内部报告则这个属性是必需的。

12.2.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.2.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.2.3 对象类型属性

这个属性的类型是 **BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**模拟输出（ANALOG_OUTPUT）**。

12.2.4 当前值属性（可命令的）

这个属性的类型是**实数**类型，以工程单位表示输出的当前值（见 12.2.11 节）。

12.2.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.2.6 设备类型属性

这个属性的类型是**字符串**类型，是一段对映射为这个模拟输出对象的物理设备的文本描述。通常用于描述与模拟输出对象对应的设备型号。

12.2.7 状态标志属性

这个属性的类型是 **BACnet 状态标志**类型，表示四种布尔标志，这些标志表示模拟输出设备运行时刻的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中，故障，管制，脱离服务**}

其中，

报警中 如果**事件状态**（见 12.2.8 节）属性值为**正常**则为 FALSE(0)，否则为 TRUE(1) 。

故障 如果**可靠性**（见 12.2.9 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制 如果某值被与 **BACnet 设备**本身的有关机制所管制，则为 TRUE(1)。在此情况下，“管制”表示**当前值**和**可靠性**属性值不再随物理设备的输出变化而变化。

脱离服务 如果**脱离服务**属性值（见 12.2.10 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.2.8 事件状态属性

这个属性的类型是 **BACnet 事件状态**类型，用于检测对象是否处于事件活动状态。如果对象支持内部报告，这个属性表示对象的事件状态。如果对象不支持内部报告，这个属性的值为**正常**。

12.2.9 可靠性属性

这个属性的类型是 **BACnet 可靠性**类型，表示**当前值**或物理输出设备的运行是否“可靠”，如果不可靠，则指明原因。这个属性有下列值：

{**未发现故障，开路，短路，无输出，其它不可靠**}

12.2.10 脱离服务属性

这个属性的类型是**布尔**类型，表示该对象代表的物理节点是（TRUE）否（FALSE）不与设备相关。当**脱离服务**为 TRUE 时，**当前值**属性与物理输出设备分离。当**脱离服务**为 TRUE 时，**可靠性**属性和对应的**状态标志**属性中的**故障**标志的状态也与物理输出设备分离。当**脱离服务**为 TRUE 时，当用于模拟专门的固定条件或用于测试时，**当前值**与**可靠性**属性可以为任何值。而其它依靠**当前值**或**可靠性**属性的功能将响应这些变化，就像这些变化发生在物理输出中一样。在**脱离服务**为 TRUE 的情况下，**当前值**属性还要被 BACnet 命令优先机制控制，参见 19 节。

12.2.11 单位属性

这个属性的类型是 **BACnet 工程单位**类型，表示这个对象的测量单位。详见第 21 节中 BACnetEngineeringUnits ASN.1 部分，其中定义了一系列标准工程单位。

12.2.12 最小值属性

这个属性的类型是**实数**类型，表示**当前值**属性的最小可靠数值（用工程单位表示）。

12.2.13 最大值属性

这个属性的类型是**实数**类型，表示**当前值**属性的最大可靠数值（用工程单位表示）。

12.2.14 分辨率属性

这个属性的类型是**实数**类型，表示**当前值**属性中以工程单位可分辨的最小变化（只读）。

12.2.15 优先值数组属性

这个属性是优先值的只读数组。详见第 19 节有关优先机制的描述。

12.2.16 释放缺省属性

当**优先值数组**中的所有命令优先值为 NULL 时，这个属性的值是**当前值**属性的缺省值。详见第 19 节。

12.2.17 COV 增量属性

这个属性的类型是**实数**类型，定义**当前值**属性的最小改变值，这个最小改变值将导致向 COV 客户发布 **COV 通告**。如果对象支持 COV 报告，则必须具备这个属性。

12.2.18 时间延迟属性

这个属性的类型是**无符号整型**类型，定义从**当前值**属性处于由**高阈值**和**低阈值**确定的范围之外开始，到生成一个**进入异常**事件之间的最小时间间隔（以秒为单位），或者从**当前**

值属性进入由**高阈值**和**低阈值**确定的范围（包括**阈值宽度**属性值确定的范围）之内时，到生成一个**进入正常**事件的最小时间间隔（以秒为单位）。如果对象支持内部报告则必须具备这个属性。

12.2.19 通告类属性

这个属性的类型是**无符号整型**类型，用于定义这个对象处理和生成事件通告时的通告类别。这个属性缺省引用有相同**通告类**属性值的**通告类**对象。如果对象支持内部报告则必须具备这个属性。

12.2.20 高阈值属性

这个属性的类型是**实数**类型，定义生成一个事件的**当前值**属性的上限值。如果对象支持内部报告则必须具备这个属性。

12.2.20.1 生成进入异常事件条件

进入异常事件在达到下列三个条件的情况下生成：

(d) **当前值**超过**高阈值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(e) **阈值使能**属性设置为**高阈值使能**标志，

(f) **事件使能**属性设置为**进入异常**标志。

12.2.20.2 生成进入正常事件条件

一旦**当前值**超过**高阈值**之后，只有在**当前值**下降到低于(**高阈值 - 阈值宽度**)之后，才能生成**进入正常**事件。**进入正常**事件在达到下列三个条件的情况下生成：

(a) **当前值**下降到低于（**高阈值 - 阈值宽度**）之后一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**高阈值使能**标志，

(c) **事件使能**属性设置为**进入正常**标志。

12.2.21 低阈值属性

这个属性的类型是**实数**类型，定义生成一个事件的**当前值**属性的下限值。如果对象支持内部报告则必须具备这个属性。

12.2.21.1 生成进入异常事件条件

进入异常事件在达到下列三个条件的情况下生成：

(a) **当前值**低于**低阈值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**低阈值使能**标志，

(c) **事件使能**属性设置为**进入异常**标志。

12.2.21.2 生成进入正常事件条件

一旦**当前值**低于**低阈值**之后，只有在**当前值**上升到高于(**低阈值** + **阈值宽度**)之后，才能生成**进入正常**事件。**进入正常**事件在达到下列三个条件的情况下生成：

(a) **当前值**上升到高于（**低阈值** + **阈值宽度**）之后一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**低阈值使能**标志，

(c) **事件使能**属性设置为**进入正常**标志。

12.2.22 阈值宽度属性

这个属性的类型是**实数**类型，定义**高阈值**属性和**低阈值**属性之间的一个宽度范围值。如果要生成一个**进入正常**的事件，**当前值**属性值必须维持在这个范围之内。**进入正常**事件在达到下列五个条件的情况下生成：

(a) **当前值**低于（**高阈值** - **阈值宽度**），

(b) **当前值**高于（**低阈值** + **阈值宽度**），

(c) **当前值**维持在这个范围之内一段由**时间延迟**属性确定的最小时间间隔的时间，

(d) **阈值使能**属性设置为**高阈值使能**或者**低阈值使能**之一的标志，

(e) **事件使能**属性设置为**进入正常**标志。

如果对象支持内部报告则必须具备这个属性。

12.2.23 阈值使能属性

这个属性的类型是 BACnet **阈值使能**类型，用两个标志分别表示使能或者禁止对于**高阈值**异常事件和**低阈值**异常事件以及各自返回到正常事件的报告。如果对象支持内部报告则必须具备这个属性。

12.2.24 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示使能或者禁止对于**进入异常**、**进入故障**和**进入正常**事件的报告。在**模拟输出**对象环境中，向**高阈值**和**低阈值**的**事件状态**的转换称为“异常”事件。如果对象支持内部报告则必须具备这个属性。

12.2.25 确认转换属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。在**模拟输出**对象环境中，向**高阈值**和**低阈值**的**事件状态**的转换称为“异常”事件。这些标志将在相应事件出现的情况下被清除，并在下列任一条件

下设置：

(a) 收到相应的确认；

(b) 如果**事件使能**属性中相应的标志未设置时，事件发生（即在这种情况下，不会产生事件通告，因此也不会产生确认）；

(c) 如果**事件使能**属性中设置了相应的标志，并且在通告类对象的**确认转换**属性中未设置相应标志时，事件发生（即无确认产生）。

如果对象支持内部报告则必须具备这个属性。

12.2.26 通告类型属性

这个属性的类型是 BACnet **通告类型**类型，表示对象生成的通告是**事件**还是**报警**。如果对象支持内部报告则必须具备这个属性。

12.3 模拟值对象类型（Analog Value Object Type）

模拟值对象类型定义为一个标准对象，其属性表示一个模拟值的外部可见一致性代码。**BACnet 设备**的一个“模拟值”是驻留在这个设备的内存中的一个控制系统参数。这个对象及其属性见表 12-3，本节进行详细规范。

表 12-3　模拟值对象的属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	实数 REAL	W
描述 Description	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R
事件状态 Event_State	BACnet 事件状态 BACnet EventState	R
可靠性 Reliability	BACnet 可靠性 BACnetReliability	O
脱离服务 Out_Of_Service	布尔 BOOLEAN	R
单位 Units	BACnet 工程单位 BACnetEngineeringUnits	R
优先值数组 Priority_Array	BACnet 优先值数组 BACnetPriorityArray	O[1]
释放缺省 Relinquish_Default	实数 REAL	O[1]
COV 增量 COV_Increment	实数 REAL	O[2]
时间延迟 Time_Delay	无符号整型 Unsigned	O[3]
通告类 Notification_Class	无符号整型 Unsigned	O[3]
高阈值 High_Limit	实数 REAL	O[3]
低阈值 Low_Limit	实数 REAL	O[3]
阈值宽度 Deadband	实数 REAL	O[3]
阈值使能 Limit_Enable	BACnet 阈值使能 BACnetLimitEnable	O[3]
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	O[3]
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	O[3]
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	O[3]

[1]如果可选属性中的一个存在，那么必需都存在。

[2]如果对象支持 COV 报告则这个属性是必需的。

[3]如果对象支持内部报告则该属性是必需的。

12.3.1 对象标识符属性

这个属性的类型是 BACnet **对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 BACnet **设备**中是唯一的。

12.3.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 BACnet **设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.3.3 对象类型属性

这个属性的类型是 BACnet **对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**模拟值（ANALOG_VALUE）**。

12.3.4 当前值属性（可命令的）

这个属性的类型是**实数**类型，以工程单位表示输出的当前值（见 12.3.10 节）。由于**当前值**是可命令的，可选的优先值数组必须存在。

12.3.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.3.6 状态标志属性

这个属性的类型是 BACnet **状态标志**类型，表示四种布尔标志，这些标志表示模拟值设备运行时刻的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中，故障，管制，脱离服务**}

其中，

报警中　　如果**事件状态**（见 12.3.7 节）属性值为**正常**则为 FALSE(0)，否则为 TRUE(1) 。

故障　　如果**可靠性**（见 12.3.8 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制　　如果某值被与 **BACnet 设备**本身的有关机制所管制，则为 TRUE(1)。在此情况下，“管制”表示**当前值**属性值不再能够通过 BACnet 服务而被改变。

脱离服务　　如果**脱离服务**属性值（见 12.3.9 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.3.7 事件状态属性

这个属性的类型是 **BACnet 事件状态**类型，用于检测对象是否处于事件活动状态。如果对象支持内部报告，这个属性表示对象的事件状态。如果对象不支持内部报告，这个属性的值为**正常**。

12.3.8 可靠性属性

这个属性的类型是 **BACnet 可靠性**类型，表示**当前值**是否“可靠”，如果不可靠，则指明原因。这个属性有下列值：

{**未发现故障，超出范围，低于范围，其它不可靠**}

12.3.9 脱离服务属性

这个属性的类型是**布尔**类型，表示**模拟值**对象的**当前值**属性值能（TRUE）否（FALSE）被含有这个对象的 BACnet 设备有关的软件所修改。当**脱离服务**为 TRUE 时，**当前值**属性还可以随意的写入。如果有**优先值数组**和**释放缺省**属性，那么**当前值**属性的写入将受到 BACnet 命令优先机制的控制。参见 19 节。

12.3.10 单位属性

这个属性的类型是 **BACnet 工程单位**类型，表示这个对象的测量单位。详见第 21 节中 BACnetEngineeringUnits ASN.1 部分，其中定义了一系列标准工程单位。

12.3.11 优先值数组属性

这个属性是只读数组，包括这个对象有效时的优先命令。详见第 19 节关于优先机制的描述。这个属性必须与**释放缺省**属性同时具备。

12.3.12 释放缺省属性

当**优先值数组**中的所有命令优先值为 NULL 时，这个属性的值是**当前值**属性的缺省值。详见第 19 节。该属性必须与**优先值数组**属性同时具备。

12.3.13 COV 增量属性

这个属性的类型是**实数**类型，定义**当前值**属性的最小改变值，这个最小改变值将导致向

COV 客户发布 **COV 通告**。如果对象支持 COV 报告，则必须具备这个属性。

12.3.14 时间延迟属性

这个属性的类型是**无符号整型**类型，定义从**当前值**属性处于由**高阈值**和**低阈值**确定的范围之外开始，到生成一个**进入异常**事件之间的最小时间间隔（以秒为单位），或者从**当前值**属性进入由**高阈值**和**低阈值**确定的范围（包括**阈值宽度**属性值确定的范围）之内时，到生成一个**进入正常**事件的最小时间间隔（以秒为单位）。如果对象支持内部报告则必须具备这个属性。

12.3.15 通告类属性

这个属性的类型是**无符号整型**类型，用于定义这个对象处理和生成事件通告时的通告类别。这个属性缺省引用有相同**通告类**属性值的**通告类**对象。如果对象支持内部报告则必须具备这个属性。

12.3.16 高阈值属性

这个属性的类型是**实数**类型，定义生成一个事件的**当前值**属性的上限值。如果对象支持内部报告则必须具备这个属性。

12.3.16.1 生成进入异常事件条件

进入异常事件在达到下列三个条件的情况下生成：

(g) **当前值**超过**高阈值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(h) **阈值使能**属性设置为**高阈值使能**标志，

(i) **事件使能**属性设置为**进入异常**标志。

12.3.16.2 生成进入正常事件条件

一旦**当前值**超过**高阈值**之后，只有在**当前值**下降到低于(**高阈值 - 阈值宽度**)之后，才能生成**进入正常**事件。**进入正常**事件在达到下列三个条件的情况下生成：

(a) **当前值**下降到低于（**高阈值 - 阈值宽度**）之后一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**高阈值使能**标志，

(c) **事件使能**属性设置为**进入正常**标志。

12.3.17 低阈值属性

这个属性的类型是**实数**类型，定义生成一个事件的**当前值**属性的下限值。如果对象支持内部报告则必须具备这个属性。

12.3.17.1 生成进入异常事件条件

进入异常事件在达到下列三个条件的情况下生成：

(a) **当前值**低于**低阈值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**低阈值使能**标志，

(c) **事件使能**属性设置为**进入异常**标志。

12.3.17.2 生成进入正常事件条件

一旦**当前值**低于**低阈值**之后，只有在**当前值**上升到高于(**低阈值** + **阈值宽度**)之后，才能生成**进入正常**事件。**进入正常**事件在达到下列三个条件的情况下生成：

(a) **当前值**上升到高于（**低阈值** + **阈值宽度**）之后一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **阈值使能**属性设置为**低阈值使能**标志，

(c) **事件使能**属性设置为**进入正常**标志。

12.3.18 阈值宽度属性

这个属性的类型是**实数**类型，定义**高阈值**属性和**低阈值**属性之间的一个宽度范围值。如果要生成一个**进入正常**的事件，**当前值**属性值必须维持在这个范围之内。**进入正常**事件在达到下列五个条件的情况下生成：

(a) **当前值**低于（**高阈值** - **阈值宽度**），

(b) **当前值**高于（**低阈值** + **阈值宽度**），

(c) **当前值**维持在这个范围之内一段由**时间延迟**属性确定的最小时间间隔的时间，

(d) **阈值使能**属性设置为**高阈值使能**或者**低阈值使能**之一的标志，

(e) **事件使能**属性设置为**进入正常**标志。

如果对象支持内部报告则必须具备这个属性。

12.3.19 阈值使能属性

这个属性的类型是 BACnet **阈值使能**类型，用两个标志分别表示使能或者禁止对于**高阈值**异常事件和**低阈值**异常事件以及各自返回到正常事件的报告。如果对象支持内部报告则必须具备这个属性。

12.3.20 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示使能或者禁止对于**进入异常**、**进入故障**和**进入正常**事件的报告。在**模拟值**对象环境中，向**高阈值**和**低阈值**的**事件状态**的转换称为“异常”事件。如果对象支持内部报告则必须具备这个属性。

12.3.21 确认转换属性

这个属性的类型是**BACnet 事件转换比特**类型,用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。在**模拟值**对象环境中，向**高阈值**和**低阈值**的**事件状态**的转换称为“异常”事件。这些标志将在相应事件出现的情况下被清除，并在下列任一条件下设置：

(a) 收到相应的确认；

(b) 如果**事件使能**属性中相应的标志未设置时，事件发生（即在这种情况下，不会产生事件通告，因此也不会产生确认）；

(c) 如果**事件使能**属性中设置了相应的标志，并且在通告类对象的**确认转换**属性中未设置相应标志时，事件发生（即无确认产生）。

如果对象支持内部报告则必须具备这个属性。

12.3.22 通告类型属性

这个属性的类型是 **BACnet 通告类型**类型，表示对象生成的通告是**事件**还是**报警**。如果对象支持内部报告则必须具备这个属性。

12.4 二进制输入对象类型（Binary Input Object Type）

二进制输入对象类型定义为一个标准对象,它的属性表示二进制输入的外部可见一致性代码。“二进制输入”是物理设备或硬件的输入，该输入只存在两种状态，即“**活动（ACTIVE）**”状态和“**非活动（INACTIVE）**”状态。二进制输入的主要用途是指明机械设备状态，如：风机或水泵是否运行。**活动**表示设备开或运转，**非活动**表示设备关或未运行。

在某些应用中,电路使应用层逻辑状态的**活动**和**非活动**与底层硬件的物理状态相反。如：当继电器通电时，常开型继电器导致**活动**状态，而常合型继电器却导致**非活动**状态。二进制输入对象通过使用**极性（Polarity）**属性确定具体状态。参见 12.4.4 节和 12.4.11 节。

这个对象及其属性见表 12-4，本节进行详细规范。

表 12-4　二进制输入对象的属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	BACnet 二进制值 BACnetBinaryPV	R[1]
描述 Description	字符串 CharacterString	O

设备类型 Device_Type	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R
事件状态 Event_State	BACnet 事件状态 BACnet EventState	R
可靠性 Reliability	BACnet 可靠性 BACnetReliability	O
脱离服务 Out_Of_Service	布尔 BOOLEAN	R
极性 Polarity	BACnet 极性 BACnetPolarity	R
非活动文本 Inactive_Text	字符串 CharacterString	O[2]
活动文本 Active_Text	字符串 CharacterString	O[2]
状态改变时间 Change_Of_State_Time	BACnet 日期时间 BACnetDateTime	O[3]
状态改变次数 Change_Of_State_Count	无符号整型 Unsigned	O[3]
改变时间置 0 Time_Of_State_Count_Reset	BACnet 日期时间 BACnetDateTime	O[3]
当前值活动累计 Elapsed_Active_Time	32 位无符号整型 Unsigned32	O[4]
活动时间置 0 Time_Of_Active_Time_Reset	BACnet 日期时间 BACnetDateTime	O[4]
时间延迟 Time_Delay	无符号整型 Unsigned	O[5]
通告类 Notification_Class	无符号整型 Unsigned	O[5]
报警值 Alarm_Value	BACnet 二进制值 BACnetBinaryPV	O[5]
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	O[5]
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	O[5]
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	O[5]

[1] 当**脱离服务**为 TRUE 时该属性必须是可写的。

[2] 可选属性**非活动文本**与**活动文本**必须同时存在。

[3] 可选属性**状态改变时间、状态改变次数、改变时间置 0** 必须同时存在。

[4] 可选属性**当前值活动累计、活动时间置 0** 必须同时存在。

[5] 如果对象支持内部报告，则必须具备这个属性。

12.4.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.4.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 BACnet **设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.4.3 对象类型属性

这个属性的类型是 BACnet **对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**二进制输入（BINARY_INPUT）**。

12.4.4 当前值属性

这个属性的类型是 BACnet **二进制值**类型，反映了**二进制输入**对象的逻辑状态，逻辑状态只有**非活动**或**活动**两种。**当前值**与输入物理状态间的关系由**极性**属性决定。可能的状态见下表 12-5。

表 12-5　BACnet 极性关系

当前值属性	极性属性	输入物理状态	设备物理状态
非活动	**正向（NORMAL）**	OFF 或**非活动**	未运行
活动	**正向（NORMAL）**	ON 或**活动**	运行
非活动	**反向（REVERSE）**	ON 或**活动**	未运行
活动	**反向（REVERSE）**	OFF 或**非活动**	运行

当**脱离服务**为 TRUE 时，**当前值**属性必须可写（见 12.4.10 节）。

12.4.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.4.6 设备类型属性

这个属性的类型是**字符串**类型，是一段对映射为这个二进制输入对象的物理设备的文本描述。通常用于描述与二进制输入对应的设备的型号。

12.4.7 状态标志属性

这个属性的类型是 BACnet **状态标志**类型，表示四种布尔标志，这些标志表示二进制输入设备运行时刻的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中，故障，管制，脱离服务**}

其中，

报警中　　如果**事件状态**（见 12.4.8 节）属性值为**正常**则为 FALSE(0)，否则为 TRUE(1) 。

故障　　如果**可靠性**（见 12.4.9 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制　　如果某值被与 **BACnet 设备**本身的有关机制所管制，则为 TRUE(1)。在此情况下，“管制”表示**当前值**和**可靠性**属性值不再随物理设备的输入变化而变化。

脱离服务　　如果**脱离服务**属性值（见 12.4.10 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.4.8 事件状态属性

这个属性的类型是 **BACnet 事件状态**类型，用于检测对象是否处于事件活动状态。如果对象支持内部报告，这个属性表示对象的事件状态。如果对象不支持内部报告，这个属性的值为**正常**。

12.4.9 可靠性属性

这个属性的类型是 **BACnet 可靠性**类型，表示**当前值**或物理输入设备的运行是否“可靠”，如果不可靠，则指明原因。这个属性有下列值：

{**未发现故障**，**无传感器**，**开路**，**短路**，**其它不可靠**}

12.4.10 脱离服务属性

这个属性的类型是**布尔**类型，表示该对象代表的物理输入是（TRUE）否（FALSE）不与设备相关。当**脱离服务**为 TRUE 时，**当前值**属性与物理输入设备分离，并不会随着物理输入设备的改变而变化。当**脱离服务**为 TRUE 时，**可靠性**属性和对应的**状态标志**属性中的**故障**标志的状态也与物理输入设备分离。当**脱离服务**为 TRUE 时，当用于模拟专门的固定条件或用于测试时，**当前值**与**可靠性**属性可以为任何值。而其它依靠**当前值**或**可靠性**属性的功能将响应这些变化，就像这些变化发生在输入中一样。

12.4.11 极性属性

这个属性的类型是 **BACnet 极性**类型，表述了输入的物理状态和**当前值**代表的逻辑状态之间的关系。当**脱离服务**为 FALSE 时，如果这个属性为**正向**， **当前值**中的**活动**就是物理输入中的**活动**或 ON 状态；如果该属性为**反向**，**当前值**中的**活动**是物理输入中的**非活动**或 OFF 状态。见表 12-5。因此，当**脱离服务**为 FALSE 时，对于某一恒定的物理输入状态，**极性**属

性的改变将导致**当前值**属性的变化。如果**脱离服务**为 TRUE，**极性**属性将对**当前值**属性无影响。

12.4.12 非活动文本属性

这个属性的类型是**字符串**类型,从人操作的角度描述了**当前值**在**非活动**状态下的预期效果。字符串的内容由生产商自行确定，但一定要表示**非活动**状态的可读描述。例如：如果物理输入设备连接在一个转换开关上，那么**非活动文本**可能被赋值为“风机 1 关”。**非活动文本**属性与**活动文本**属性必须同时存在。

12.4.13 活动文本属性

这个属性的类型是**字符串**类型，从人操作的角度描述了**当前值**在**活动**状态下的预期效果。字符串的内容由生产商自行确定，但一定要表示**活动**状态的可读的描述。例如：如果物理输入连接在一个转换开关上，那么**活动文本**可能被赋值为“风机 1 开”。 **活动文本**属性与**非活动文本**属性必需同时存在。

12.4.14 状态改变时间属性

这个属性的类型是 **BACnet 日期时间**类型，表示最近一次状态发生改变的日期和时间。“状态改变”是指任何引起**当前值**属性改变的事件。当**脱离服务**为 FALSE 时，**极性**属性的改变将导致**当前值**属性的变化，因而被认为是状态改变。如果**脱离服务**为 TRUE，**极性**属性将对**当前值**属性无影响。可选属性**状态改变时间**、**状态改变次数**、**改变时间置 0** 必须同时存在。

12.4.15 状态改变次数属性

这个属性的类型是**无符号整型**类型，表示从该属性最近一次被设置为 0 以来，**当前值**属性改变状态的次数。该属性取值范围为 0-65535 或更大。“状态改变”是指任何引起**当前值**属性改变的事件。当**脱离服务**为 FALSE 时，**极性**属性的改变将导致**当前值**属性的变化，因而被认为是状态改变。如果**脱离服务**为 TRUE，**极性**属性将对**当前值**属性无影响。可选属性**状态改变时间**、**状态改变次数**、**改变时间置 0** 必须同时存在。

12.4.16 改变时间置 0 属性

这个属性的类型是 **BACnet 日期时间**类型，表示**状态改变次数**属性最近一次被设置为 0 的日期和时间。可选属性**状态改变时间**、**状态改变次数**、**改变时间置 0** 必须同时存在。

12.4.17 当前值活动累计属性

这个属性的类型是 32 位**无符号整型**类型，表示从该属性最近一次被设置为 0 以来，**当**

前值属性为**活动**的累积时间（以秒为单位）。可选属性**当前值活动累计**、**活动时间置 0** 必须同时存在。

12.4.18 活动时间置 0 属性

这个属性的类型是 BACnet **日期时间**类型，表示**当前值活动累计**属性最近一次被设置为 0 的日期和时间。可选属性**当前值活动累计**、**活动时间置 0** 必须同时存在。

12.4.19 时间延迟属性

这个属性的类型是**无符号整型**类型，表示生成**进入异常**事件时**当前值**必须保持等于**报警值**属性的最小时间（以秒为单位），或者生成**进入正常**事件时**当前值**必须保持不等于**报警值**属性的最小时间（以秒为单位）。如果对象支持内部报告则必需具备该属性。

12.4.20 通告类属性

这个属性的类型是**无符号整型**类型，用于定义这个对象处理和生成事件通告时的通告类别。这个属性缺省引用有相同**通告类**属性值的**通告类**对象。如果对象支持内部报告则必须具备这个属性。

12.4.21 报警值属性

这个属性的类型是 BACnet **二进制值**类型，表示生成事件时，**当前值**属性的值。如果对象支持内部报告则必需具备该属性。

12.4.21.1 生成进入异常事件条件

进入异常事件在下列两个条件下生成：

(a) **当前值**必须等于**报警值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入异常**标志。

12.4.21.2 生成进入正常事件条件

一旦**当前值**等于**报警值**，要生成**进入正常**事件，**当前值**必须不等于**报警值**。**进入正常**事件在下列两个条件下生成：

(a) **当前值**必须不等于**报警值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入正常**标志。

12.4.22 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示使能或者禁止对于

进入异常、**进入故障**和**进入正常**事件的报告。如果对象支持内部报告则必须具备这个属性。

12.4.23 确认转换属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。这些标志将在相应事件出现的情况下被清除，并在下列任一条件下设置：

(a) 收到相应的确认；

(b) 如果**事件使能**属性中相应的标志未设置时，事件发生（即在这种情况下，不会产生事件通告，因此也不会产生确认）；

(c) 如果**事件使能**属性中设置了相应的标志，并且在通告类对象的**确认转换**属性中未设置相应标志时，事件发生（即无确认产生）。

如果对象支持内部报告则必须具备这个属性。

12.4.24 通告类型属性

这个属性的类型是 BACnet **通告类型**类型，表示对象生成的通告是**事件**还是**报警**。如果对象支持内部报告则必须具备这个属性。

12.5 二进制输出对象类型（Binary Output Object Type）

二进制输出对象类型定义为一个标准对象，它的属性表示二进制输出的外部可见一致性代码。“二进制输出”是物理设备或硬件的输出，该输出只存在两种状态，即“**活动**”状态和“**非活动**”状态。二进制输出的主要用途是切换机械设备状态，如：风机或水泵的开和关。**活动**表示设备开或运转，**非活动**表示设备关或未运行。

在某些应用中，电路使应用层逻辑状态的**活动**和**非活动**与底层硬件的物理状态相反。如：当继电器通电时，常开型继电器导致**活动**状态，而常合型继电器却导致**非活动**状态。二进制输出对象通过使用**极性**属性确定具体状态。参见 12.5.4 节和 12.5.11 节。

这个对象及其属性见表 12-6，本节进行详细规范。

表 12-6　二进制输出对象的属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	BACnet 二进制值 BACnetBinaryPV	W
描述 Description	字符串 CharacterString	O
设备类型 Device_Type	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R
事件状态 Event_State	BACnet 事件状态 BACnet EventState	R
可靠性 Reliability	BACnet 可靠性 BACnetReliability	O
脱离服务 Out_Of_Service	布尔 BOOLEAN	R
极性 Polarity	BACnet 极性 BACnetPolarity	R
非活动文本 Inactive_Text	字符串 CharacterString	O[1]
活动文本 Active_Text	字符串 CharacterString	O[1]
状态改变时间 Change_Of_State_Time	BACnet 日期时间 BACnetDateTime	O[2]
状态改变次数 Change_Of_State_Count	无符号整型 Unsigned	O[2]
改变时间置 0	BACnet 日期时间 BACnetDateTime	O[2]

Time_Of_State_Count_Reset		
当前值活动累计 Elapsed_Active_Time	32 位无符号整型 Unsigned32	0[3]
活动时间置 0 Time_Of_Active_Time_Reset	BACnet 日期时间 BACnetDateTime	0[3]
非活动最小值 Minimum_Off_Time	32 位无符号整型 Unsigned32	0
活动最小值 Minimum_On_Time	32 位无符号整型 Unsigned32	0
优先值数组 Priority_Arra	BACnet 优先值数组 BACnetPriorityArra	R
释放缺省 Relinquish_Default	BACnet 二进制值 BACnetBinaryPV	R
时间延迟 Time_Delay	无符号整型 Unsigned	0[4]
通告类 Notification_Class	无符号整型 Unsigned	0[4]
反馈值 Feedback_Value	BACnet 二进制值 BACnetBinaryPV	0[4]
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	0[4]
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	0[4]
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	04

[1] 可选属性**非活动文本**与**活动文本**必须同时存在。

[2] 可选属性**状态改变时间**、**状态改变次数**、**改变时间置 0** 必须同时存在。

[3] 可选属性**当前值活动累计**、**活动时间置 0** 必须同时存在。

[4] 如果对象支持内部报告，则必须具备这个属性。

12.5.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.5.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.5.3 对象类型属性

这个属性的类型是 **BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关

系。在本对象中，这个属性的值为**二进制输出（BINARY_OUTPUT）**。

12.5.4 当前值属性

这个属性的类型是 **BACnet 二进制值**类型，反映了**二进制输出**对象的逻辑状态，逻辑状态只有**非活动**或**活动**两种。**当前值**与输出物理状态间的关系由**极性**属性决定。可能的状态见下表 12-5。

表 12-7　BACnet 极性关系

当前值属性	极性属性	输入物理状态	设备物理状态
非活动	**正向（NORMAL）**	OFF 或**非活动**	未运行
活动	**正向（NORMAL）**	ON 或**活动**	运行
非活动	**反向（REVERSE）**	ON 或**活动**	未运行
活动	**反向（REVERSE）**	OFF 或**非活动**	运行

12.5.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.5.6 设备类型属性

这个属性的类型是**字符串**类型，是一段对映射为这个二进制输出对象的物理设备的文本描述。通常用于描述与二进制输出对应的设备的型号。

12.5.7 状态标志属性

这个属性的类型是 **BACnet 状态标志**类型，表示四种布尔标志，这些标志表示二进制输出设备运行时刻的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中，故障，管制，脱离服务**}

其中，

报警中　如果**事件状态**（见 12.5.8 节）属性值为**正常**则为 FALSE(0)，否则为 TRUE(1) 。

故障　如果**可靠性**（见 12.5.9 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制　如果某值被与 **BACnet 设备**本身的有关机制所管制，则为 TRUE(1)。在此情况下，“管制”表示**当前值**和**可靠性**属性值不再随物理设备的输出变化而变化。

脱离服务　　如果**脱离服务**属性值（见 12.5.10 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.5.8 事件状态属性

这个属性的类型是 **BACnet 事件状态**类型，用于检测对象是否处于事件活动状态。如果对象支持内部报告，这个属性表示对象的事件状态。如果对象不支持内部报告，这个属性的值为**正常**。

12.5.9 可靠性属性

这个属性的类型是 **BACnet 可靠性**类型，表示**当前值**或物理输出设备的运行是否“可靠”，如果不可靠，则指明原因。这个属性有下列值：

{**未发现故障**，**无输出**，**开路**，**短路**，**其它不可靠**}

12.5.10 脱离服务属性

这个属性的类型是**布尔**类型，表示该对象代表的物理输出是（TRUE）否（FALSE）不与设备相关。当**脱离服务**为 TRUE 时，**当前值**属性与物理输出设备分离。当**脱离服务**为 TRUE 时，**可靠性**属性和对应的**状态标志**属性中的**故障**标志的状态也与物理输出设备分离。当**脱离服务**为 TRUE 时，当用于模拟专门的固定条件或用于测试时，**当前值**与**可靠性**属性可以为任何值。而其它依靠**当前值**或**可靠性**属性的功能将响应这些变化，就像这些变化发生在输出中一样。在**脱离服务**为 TRUE 的情况下，**当前值**属性还要被 BACnet 命令优先机制控制，参见 19 节。

12.5.11 极性属性

这个属性的类型是 **BACnet 极性**类型，表述了输出的物理状态和**当前值**代表的逻辑状态之间的关系。当**脱离服务**为 FALSE 时，如果这个属性为**正向**， **当前值**中的**活动**就是物理输出中的**活动**或 ON 状态；如果该属性为**反向**，**当前值**中的**活动**是物理输出中的**非活动**或 OFF 状态。见表 12-7。如果**脱离服务**为 TRUE，**极性**属性将对**当前值**属性无影响。

12.5.12 非活动文本属性

这个属性的类型是**字符串**类型，从人操作的角度描述了**当前值**在**非活动**状态下的预期效果。字符串的内容由生产商自行确定，但一定要表示**非活动**状态的可读描述。例如：如果物理输出设备连接在一个控制灯开关的继电器上，那么**非活动文本**可能被赋值为“关灯”。**非活动文本**属性与**活动文本**属性必须同时存在。

12.5.13 活动文本属性

这个属性的类型是**字符串**类型，从人操作的角度描述了**当前值**在**活动**状态下的预期效果。字符串的内容由生产商自行确定，但一定要表示**活动**状态的可读的描述。例如：如果物理输出设备连接在一个控制灯开关的继电器上，那么**活动文本**可能被赋值为“开灯”。 **活动文本**属性与**非活动文本**属性必需同时存在。

12.5.14 状态改变时间属性

这个属性的类型是 BACnet **日期时间**类型，表示最近一次状态发生改变的日期和时间。“状态改变”是指任何引起**当前值**属性改变的事件。**极性**属性的改变并不导致状态改变。可选属性**状态改变时间**、**状态改变次数**、**改变时间置 0** 必须同时存在。

12.5.15 状态改变次数属性

这个属性的类型是**无符号整型**类型，表示从该属性最近一次被设置为 0 以来，**当前值**属性改变状态的次数。该属性取值范围为 0-65535 或更大。“状态改变”是指任何引起**当前值**属性改变的事件。**极性**属性的改变并不导致状态改变。可选属性**状态改变时间**、**状态改变次数**、**改变时间置 0** 必须同时存在。

12.5.16 改变时间置 0 属性

这个属性的类型是 BACnet **日期时间**类型，表示**状态改变次数**属性最近一次被设置为 0 的日期和时间。可选属性**状态改变时间**、**状态改变次数**、**改变时间置 0** 必须同时存在。

12.5.17 当前值活动累计属性

这个属性的类型是 32 位**无符号整型**类型，表示从该属性最近一次被设置为 0 以来，**当前值**属性为**活动**的累积时间（以秒为单位）。可选属性**当前值活动累计**、**活动时间置 0** 必须同时存在。

12.5.18 活动时间置 0 属性

这个属性的类型是 BACnet **日期时间**类型，表示**当前值活动累计**属性最近一次被设置为 0 的日期和时间。可选属性**当前值活动累计**、**活动时间置 0** 必须同时存在。

12.5.19 非活动最小值属性

这个属性的类型是 32 位**无符号整型**类型，表示向**当前值**属性写入**非活动**状态后**当前值**保持该状态的最短时间（以秒为单位）。具体机制见 19.3 节。

12.5.20 活动最小值属性

这个属性的类型是 32 位**无符号整型**类型，表示向**当前值**属性写入**活动**状态后**当前值**保

持该状态的最短时间（以秒为单位）。具体机制见 19.3 节。

12.5.21 优先值数组属性

这个属性是只读数组，该数组包含对该对象有效的优先命令。详见第 19 节关于优先机制的描述。

12.5.22 释放缺省属性

当**优先值数组**中的所有命令优先值为 NULL 时，该属性是用于**当前值**属性的缺省值。详见 19 节。

12.5.23 时间延迟属性

这个属性的类型是**无符号整型**类型，表示生成**进入异常**事件时**当前值**必须保持不等于**反馈值**属性的最小时间（以秒为单位），或者生成**进入正常**事件时**当前值**必须保持等于**反馈值**属性的最小时间（以秒为单位）。如果对象支持内部报告则必需具备该属性。

12.5.24 通告类属性

这个属性的类型是**无符号整型**类型，用于定义这个对象处理和生成事件通告时的通告类别。这个属性缺省引用有相同**通告类**属性值的**通告类**对象。如果对象支持内部报告则必须具备这个属性。

12.5.25 反馈值属性

这个属性的类型是 BACnet **二进制值**类型，表示生成事件时，**当前值**属性的值必须与之不同的一个反馈值的状态。如果对象支持内部报告则必需具备该属性。如何确定**反馈值**的方法由生产商自行确定。

12.5.25.1 生成进入异常事件条件

进入异常事件在下列两个条件下生成：

(a) **当前值**必须不等于**反馈值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入异常**标志。

12.5.25.2 生成进入正常事件条件

一旦**当前值**不等于**反馈值**，要生成**进入正常**事件，**当前值**必须等于**反馈值**。**进入正常**事件在下列两个条件下生成：

(a) **当前值**必须等于**反馈值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入正常**标志。

12.5.26 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示使能或者禁止对于**进入异常**、**进入故障**和**进入正常**事件的报告。如果对象支持内部报告则必须具备这个属性。

12.5.27 确认转换属性

这个属性的类型是 BACnet **事件转换比特**类型,用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。这些标志将在相应事件出现的情况下被清除，并在下列任一条件下设置：

(a) 收到相应的确认；

(b) 如果**事件使能**属性中相应的标志未设置时，事件发生（即在这种情况下，不会产生事件通告，因此也不会产生确认）；

(c) 如果**事件使能**属性中设置了相应的标志，并且在通告类对象的**确认转换**属性中未设置相应标志时，事件发生（即无确认产生）。

如果对象支持内部报告则必须具备这个属性。

12.5.28 通告类型属性

这个属性的类型是 BACnet **通告类型**类型，表示对象生成的通告是**事件**还是**报警**。如果对象支持内部报告则必须具备这个属性。

12.6 二进制值对象类型（Binary Value Object Type）

二进制值对象类型定义为一个标准对象，它的属性表示二进制值的外部可见一致性代码。“二进制值”是驻留在 BACnet **设备**内存中的控制系统参数。这个参数只存在两种状态即：**“活动”**状态和**“非活动”**状态。

二进制值对象及其属性见表 12-8，本节进行详细规范。

表 12-8　二进制值对象的属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	BACnet 二进制值 BACnetBinaryPV	R[1]
描述 Description	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R
事件状态 Event_State	BACnet 事件状态 BACnet EventState	R

可靠性 Reliability	BACnet 可靠性 BACnetReliability	0
脱离服务 Out_Of_Service	布尔 BOOLEAN	R
非活动文本 Inactive_Text	字符串 CharacterString	O[2]
活动文本 Active_Text	字符串 CharacterString	O[2]
状态改变时间 Change_Of_State_Time	BACnet 日期时间 BACnetDateTime	O[3]
状态改变次数 Change_Of_State_Count	无符号整型 Unsigned	O[3]
改变时间置 0 Time_Of_State_Count_Reset	BACnet 日期时间 BACnetDateTime	O[3]
当前值活动累计 Elapsed_Active_Time	32 位无符号整型 Unsigned32	O[4]
活动时间置 0 Time_Of_Active_Time_Reset	BACnet 日期时间 BACnetDateTime	O[4]
非活动最小值 Minimum_Off_Time	32 位无符号整型 Unsigned32	0
活动最小值 Minimum_On_Time	32 位无符号整型 Unsigned32	0
优先值数组 Priority_Arra	BACnet 优先值数组 BACnetPriorityArra	O[5]
释放缺省 Relinquish_Default	BACnet 二进制值 BACnetBinaryPV	O[5]
时间延迟 Time_Delay	无符号整型 Unsigned	O[6]
通告类 Notification_Class	无符号整型 Unsigned	O[6]
报警值 Alarm_Value	BACnet 二进制值 BACnetBinaryPV	O[6]
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	O[6]
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	O[6]
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	O[6]

[1] 当**脱离服务**为 TRUE 时该属性必须是可写的。

[2] 可选属性**非活动文本**与**活动文本**必须同时存在。

[3] 可选属性**状态改变时间**、**状态改变次数**、**改变时间置 0** 必须同时存在。

[4] 可选属性**当前值活动累计**、**活动时间置 0** 必须同时存在。

[5] 可选属性**优先值数组**、**释放缺省**必须同时存在。**当前值**属性也必须是可写的。

[6] 如果对象支持内部报告，则必须具备这个属性。

12.6.1 对象标识符属性

这个属性的类型是 BACnet **对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 BACnet **设备**中是唯一的。

12.6.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 BACnet **设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.6.3 对象类型属性

这个属性的类型是 BACnet **对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**二进制值**（BINARY_VALUE）。

12.6.4 当前值属性（可命令的）

这个属性的类型是 BACnet **二进制值**类型，反映了**二进制值**对象的逻辑状态，逻辑状态只有**非活动**或**活动**两种。由于**当前值**是可命令的，可选的优先值数组必须存在。当**脱离服务**为 TRUE 时，**当前值**属性是可写的。参见 12.6.9 节

12.6.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.6.6 状态标志属性

这个属性的类型是 BACnet **状态标志**类型，表示四种布尔标志，这些标志表示二进制值设备运行时刻的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中**，**故障**，**管制**，**脱离服务**}

其中，

报警中　　如果**事件状态**（见 12.6.7 节）属性值为**正常**则为 FALSE(0)，否则为 TRUE(1) 。

故障　　如果**可靠性**（见 12.6.8 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制　　如果某值被与 BACnet **设备**本身的有关机制所管制，则为 TRUE(1)。在此情

况下，“管制”表示**当前值**属性值不再能够通过 BACnet 服务而被改变。

脱离服务　　如果**脱离服务**属性值（见 12.6.9 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.6.7 事件状态属性

这个属性的类型是 **BACnet 事件状态**类型，用于检测对象是否处于事件活动状态。如果对象支持内部报告，这个属性表示对象的事件状态。如果对象不支持内部报告，这个属性的值为**正常**。

12.6.8 可靠性属性

这个属性的类型是 **BACnet 可靠性**类型，表示**当前值**是否“可靠”，如果不可靠，则指明原因。这个属性有下列值：

{**未发现故障，其它不可靠**}

12.6.9 脱离服务属性

这个属性的类型是**布尔**类型，表示**二进制值**对象的**当前值**属性值能（TRUE）否（FALSE）被含有这个对象的 BACnet 设备有关的软件所修改。当**脱离服务**为 TRUE 时，**当前值**属性还可以随意的写入。如果有**优先值数组**和**释放缺省**属性，那么**当前值**属性的写入将受到 BACnet 命令优先机制的控制。参见 19 节。

12.6.10 非活动文本属性

这个属性的类型是**字符串**类型，从人操作的角度描述了**当前值**在**非活动**状态下的预期效果。字符串的内容由生产商自行确定，但一定要表示**非活动**状态的可读描述。**非活动文本**属性与**活动文本**属性必须同时存在。

12.6.11 活动文本属性

这个属性的类型是**字符串**类型，从人操作的角度描述了**当前值**在**活动**状态下的预期效果。字符串的内容由生产商自行确定，但一定要表示**活动**状态的可读的描述。**活动文本**属性与**非活动文本**属性必需同时存在。

12.6.12 状态改变时间属性

这个属性的类型是 **BACnet 日期时间**类型，表示最近一次状态发生改变的日期和时间。“状态改变”是指任何引起**二进制值**对象的逻辑状态改变的事件。可选属性**状态改变时间**、

状态改变次数、改变时间置 0 必须同时存在。

12.6.13 状态改变次数属性

这个属性的类型是**无符号整型**类型，表示从该属性最近一次被设置为 0 以来，**二进制值**对象改变状态的次数。该属性取值范围为 0-65535 或更大。可选属性**状态改变时间、状态改变次数、改变时间置 0** 必须同时存在。

12.6.14 改变时间置 0 属性

这个属性的类型是 BACnet **日期时间**类型，表示**状态改变次数**属性最近一次被设置为 0 的日期和时间。可选属性**状态改变时间、状态改变次数、改变时间置 0** 必须同时存在。

12.5.15 当前值活动累计属性

这个属性的类型是 32 位**无符号整型**类型，表示从该属性最近一次被设置为 0 以来，**当前值**属性为**活动**的累积时间（以秒为单位）。可选属性**当前值活动累计、活动时间置 0** 必须同时存在。

12.6.16 活动时间置 0 属性

这个属性的类型是 BACnet **日期时间**类型，表示**当前值活动累计**属性最近一次被设置为 0 的日期和时间。可选属性**当前值活动累计、活动时间置 0** 必须同时存在。

12.6.17 非活动最小值属性

这个属性的类型是 32 位**无符号整型**类型，表示向**当前值**属性写入**非活动**状态后**当前值**保持该状态的最短时间（以秒为单位）。

根据第 19 节，如果**当前值**属性是可命令的，19.3 节中给出了实现的机制。否则可以由生产商自行确定实行的机制。

12.6.18 活动最小值属性

这个属性的类型是 32 位**无符号整型**类型，表示向**当前值**属性写入**活动**状态后**当前值**保持该状态的最短时间（以秒为单位）。具体机制见 19.3 节。

根据第 19 节，如果**当前值**属性是可命令的，19.3 节中给出了实现的机制。否则可以由生产商自行确定实行的机制。

12.6.19 优先值数组属性

这个属性是只读数组，该数组包含对该对象有效的优先命令。参见第 19 节有关优先机制的规范。可选属性**优先值数组**与**释放缺省**必须同时存在。此时**当前值**也必须是可写的。

12.6.20 释放缺省属性

当**优先值数组**中的所有命令优先值为 NULL 时，该属性是用于表示**当前值**属性的缺省值。参见第 19 节。可选属性**优先值数组**与**释放缺省**必须同时存在。此时**当前值**也必须是可写的。

12.6.21 时间延迟属性

这个属性的类型是**无符号整型**类型，表示生成**进入异常**事件时**当前值**必须保持不等于**报警值**属性的最小时间（以秒为单位），或者生成**进入正常**事件时**当前值**必须保持等于**报警值**属性的最小时间（以秒为单位）。如果对象支持内部报告则必需具备该属性。

12.6.22 通告类属性

这个属性的类型是**无符号整型**类型，用于定义这个对象处理和生成事件通告时的通告类别。这个属性缺省引用有相同**通告类**属性值的**通告类**对象。如果对象支持内部报告则必须具备这个属性。

12.6.23 报警值属性

这个属性的类型是 BACnet **二进制值**类型，表示生成**进入异常**事件时，**当前值**属性的值必须具有的值。如果对象支持内部报告则必需具备该属性。

12.6.23.1 生成进入异常事件条件

进入异常事件在下列两个条件下生成：

(a) **当前值**必须等于由**报警值**属性所规定的值一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入异常**标志。

12.6.23.2 生成进入正常事件条件

一旦**当前值**等于**报警值**，要生成**进入正常**事件，**当前值**必须不等于**报警值**。**进入正常**事件在下列两个条件下生成：

(a) **当前值**必须不等于**报警值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入正常**标志。

12.6.24 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示使能或者禁止对于**进入异常**、**进入故障**和**进入正常**事件的报告。如果对象支持内部报告则必须具备这个属性。

12.6.25 确认转换属性

这个属性的类型是**BACnet 事件转换比特**类型，用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。这些标志将在相应事件出现的情况下被清除，并在下列任一条件下设置：

(a) 收到相应的确认；

(b) 如果**事件使能**属性中相应的标志未设置时，事件发生（即在这种情况下，不会产生事件通告，因此也不会产生确认）；

(c) 如果**事件使能**属性中设置了相应的标志，并且在通告类对象的**确认转换**属性中未设置相应标志时，事件发生（即无确认产生）。

如果对象支持内部报告则必须具备这个属性。

12.6.26 通告类型属性

这个属性的类型是 **BACnet 通告类型**类型，表示对象生成的通告是**事件**还是**报警**。如果对象支持内部报告则必须具备这个属性。

12.7 日期表对象类型（Calendar Object Type）

日期表对象类型定义为一个标准对象，用于描述日期列表，例如，“节假日”、“特别日”或简单的日期列表。这个对象及其属性见表 12-9。本节进行详细规范。

表 12-9 日期表对象的属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
描述 Description	字符串 CharacterString	O
当前值 Present_Value	布尔 BOOLEAN	R
日期列表 Date_List	BACnet 日期表目列表 List of BACnetCalendarEntry	R

12.7.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.7.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一

的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.7.3 对象类型属性

这个属性的类型是 **BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**日期表（CALENDER）**。

12.7.4 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.7.5 当前值属性

这个属性的类型是**布尔**类型，表示当前日期是否在日期表中。如果当前日期在**日期列表**属性中，则**当前值**为 TRUE，否则为 FALSE。

12.7.6 日期列表属性

这个属性的类型是 **BACnet 日期表目列表**类型，其中列表中的每个表目或者是一个单独的日期（用 Date 表示），或者是一个日期范围（用 BACnetDataRange 表示），或者是月份/月的周数/周的天数（用 BACnetWeekNDay 表示）。如果当前日期与日期表的表目标准匹配，那么**日期表**对象的**当前值**为 TRUE。当上述不同结构中的某个域没有定义时，该域的作用相当于一个“通配域”，用来匹配当前日期。例如，在日期范围中，如果**开始日期（startDate）**域未定义，则表示“从任何日期至**截止日期（endDate）**并且包括**截止日期**的日期”；如果**截止日期**域未定义，则表示“从**开始日期**开始的日期”。

如果日期表的表目为 BACnetWeekNDay 形式，并且没有定义“月份（month）”和“月的周数（week-of-month）”域，但是定义了“周的天数（day-of-week）”域，那么**日期表**对象将在一年中所有符合该“周的天数”的时候为 TRUE。如果 **BACnet 设备**允许向**日期列表**属性进行写入操作，则 **BACnet 日期表目列表**可以选择所有形式。

12.8 命令对象类型（Command Object Type）

命令对象类型定义为一个标准对象，其属性反映了多操作命令过程的外部可见一致性代码。**命令**对象的作用是，根据写入到**命令**对象自己的**当前值**属性中的“操作代码（action code）”，向一组对象属性写入一组值。无论何时，只要**命令**对象的**当前值**属性被写入，就会触发**命令**对象采取一组改变其它对象的属性值的操作。

通常，**命令**对象用于表示有多变量的复杂场合，尤其适用于表示有多状态的场合。例如，某个楼宇中有个特殊区域，可能有三种状态：“无人”状态、“保暖”状态和“有人”状态。为建立各状态的运行环境，需要将大量对象的属性设置成为一组已知数值。如，当处于“无

人”状态时，温度设定值可能为 65℉，并且灯是关的；而当处于“有人”状态时，温度设定值可能为 72℉，并且灯是开的，等等。

命令对象定义了某一状态与一组数值间的关系，这些数值将写入实现这一状态的一组不同对象的属性中。通常，**命令**对象是被动的。当**命令**对象的**处理中**属性为 FALSE 时，表示该对象正在等待其**当前值**属性被写入一个数值。当**当前值**属性被写入一个数值后，**命令**对象执行一系列操作。当**处理中**属性为 TRUE 时，表示**命令**对象已开始执行操作序列中的某一个操作，这个操作是根据写入到**当前值**属性中的值选取的。当**处理中**属性为 TRUE 时，如果使用**写属性**服务向**当前值**属性写入值，均会返回 Result(-)，表示拒绝写入。

当前值属性值确定在**命令**对象的操作序列中，当前应该执行哪个操作。这些操作定义在一个通过**当前值**可以检索到的一个操作序列表数组中。**操作**属性包含这个操作序列表。操作序列表可以为空，在这种情况下，除了**处理中**返回 FALSE 和**所有写入成功**属性被设置为 TRUE 外，不会产生任何操作。如果操作序列表不为空，那么对于操作序列表中的每个操作，**命令**对象将向某个 **BACnet 设备**中的某个对象的某个属性写入一个相应的值。但对远程设备的写操作不作规定。

另外，**命令**对象不保证每次写操作都会成功，并且只要有一个写操作失败时，就不会使那些成功写入的属性回复到以前的值。如果发生任何写操作失败，**所有写入成功**属性设置为 FALSE，并且 **BACnet 操作命令**的**写入成功**标志也设置为 FALSE。对于操作失败的 **BACnet 操作命令**，如果**失败退出**标志为 TRUE，那么操作序列表中所有后续的 **BACnet 操作命令**都应把它们的**写入成功**标志设置为 FALSE。如果某个写入成功，那么该 **BACnet 操作命令**的**写入成功**标志设置为 TRUE。如果所有的写入都成功，则**所有写入成功**属性设置为 TRUE。一旦所有的写操作都通过**命令**对象处理完成，则**处理中**属性重新回到 FALSE，**命令**对象处于被动状态并等待下一指令。

特别值得注意的一种情况是，写入**当前值**属性的那个值不是触发任何其它操作的值，而是触发写自己本身的操作。例如在这种情况下，如果**当前值**属性为 5 且又被写入 5，那么操作序列表中的第五个操作将再次执行。向**当前值**属性中写入 0 时，则无操作产生，与调用空操作序列表的结果相同。

命令对象是一个功能强大的对象，可以有许多应用。但如果配置不合适，**命令**对象也会导致系统混乱甚至使系统瘫痪的负面效应。由于**命令**对象可以操作其它对象的属性，它也可以操作本身。因此，**处理中**属性用于联锁以防止**命令**对象自我操作。但**命令**对象还是可以命令另一个**命令**对象，而这另一个**命令**对象又可以操作其它**命令**对象，等等。因此存在这样的可能性，一些**命令**对象形成了命令对象组，在“循环引用”的情况下，有可能发生导致系统混乱的负作用。当引用在另一个 **BACnet 设备**中的对象时，时间延迟可能增大，这将导致在循环引用错误配置的**命令**对象间的不确定行为的发生。因此，配置那些引用其它 **BACnet 设备**中对象的**命令**对象时，要特别小心。

命令对象及其属性见表 12-10，本节进行详细规范。

表 12-10　命令对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
描述 Description	字符串 CharacterString	O
当前值 Present_Value	无符号整型 Unsigned	W
处理中 In_Processs	布尔 BOOLEAN	R
所有写入成功 All_Writes_Successful	布尔 BOOLEAN	R
操作 Action	BACnet 操作序列表的 BACnet 数组 BACnetARRAY[N] of BACnetActionList	R
操作文本 Action_Text	字符串的 BACnet 数组 BACnetARRAY[N] of CharacterString	O

12.8.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.8.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.8.3 对象类型属性

这个属性的类型是 **BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**命令（COMMAND）**。

12.8.4 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.8.5 当前值属性

这个属性的类型是**无符号整型**类型，表示**命令**对象将采取或已采取何种操作。无论何时，只要写入**当前值**，都将触发**命令**对象执行改变一组其它对象属性值的一系列操作。

当前值可以写入从 0 到**操作**属性所支持的最大操作值的任意值。当**当前值**被写入时，**命令**对象执行一系列操作。**当前值**属性值确定在**命令**对象的操作序列中，当前应该执行哪个操作。这些操作定义在**操作**属性中，后者是一个关于要执行的操作的序列表数组，这个数组可以通过向**当前值**属性写入的值进行索引。操作序列表数组可以为空，表示无操作发生。如果操作序列表数组不为空，那么对于操作序列表中的每个操作，**命令**对象将向某个 BACnet **设备**中的某个对象的某个属性写入一个相应的值。

12.8.6 处理中属性

这个属性的类型是**布尔**类型，当写入**当前值**属性时，设置为 TRUE，表示**命令**对象正在执行某个操作序列。一旦所有的写入完成，该属性被设置为 FALSE。

12.8.7 所有写入成功属性

这个属性的类型是**布尔**类型，表示**当前值**属性写入时触发的操作序列是否成功完成。当**处理中**属性为 TRUE 时，这个属性设置为 FALSE。如果所有的操作都成功完成，那么设置本属性为 TRUE，同时设置**处理中**属性为 FALSE。因此，在**处理中**为 TRUE 时，这个属性不能指出当前或以前操作是否成功完成。

12.8.8 操作属性

这个属性的类型是 BACnet **操作序列表的** BACnet **数组**类型，定义为“操作序列表”的数组。这些操作序列表可以通过写入在**当前值**属性中的值索引。操作序列表可以为空，在这种情况下，除了**处理中**返回 FALSE 和**所有写入成功**属性被设置为 TRUE 外，不会产生任何操作。如果操作序列表不为空，那么对于操作序列表中的每个操作，**命令**对象将根据 BACnet **操作命令**中的规定向某个 BACnet **设备**中的某个对象的某个属性写入一个相应的值。每个写操作执行的顺序与使用**读属性**服务读取 BACnet **操作命令**序列表中的元素所得到的顺序相同。**当前值**属性值为 0 时，表示没有进行任何操作，其行为与空序列表相同。通过读取**操作**属性的 0 号下标的数组元素，可以获得所定义的操作序列表的数目。

每一个 BACnet **操作命令**就是一个关于向一个对象的一个属性写入一个值的规范说明。BACnet **操作命令**由九个部分组成，分别是：BACnet 设备标识符（可选）、对象标识符、属性标识符、条件属性数组索引、要写入的数值、条件优先级、写入延迟时间（可选）、中途退出标志和成功写入标志。各个部分及其数据类型如下：

组件	数据类型
设备标识符 Device_Identifier	BACnet 对象标识符 BACnetObjectIdentifier（可选）
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier
属性标识符 Property_Identifier	BACnet 属性标识符 BACnetPropertyIdentifier
属性数组索引 Property_Array_Index	无符号整型 Unsigned（条件）

属性值 Property_Value	任何值 Any
优先级 Priority	无符号整型 Unsigned (1..16)(条件)
向后延迟 Post_Delay	无符号整型 Unsigned (可选)
失败退出 Quit_On_Failure	布尔 BOOLEAN
写入成功 Write_Successful	布尔 BOOLEAN

如果**设备标识符**不存在，写操作应作用于包含**命令**对象设备中的对象。支持**命令**对象类型的设备不必支持向外部设备的写操作。如果**属性标识符**引用了数组属性，则应存在**属性数组索引**属性，以确定要被写入的属性数组的索引。如果要被写入的属性为可命令属性，则应提供优先值，否则忽略该属性。如果**失败退出**标志为 TRUE，则如果是由于任何原因而导致的写操作失败，设备都将提前终止操作序列的运行，否则，则在失败后应继续执行操作序列中的下一个写操作。不论哪种情况，**所有写入成功**属性在所有操作过程中均应为 FALSE。如果**向后延迟**存在，就表示在每个不论成功与否的写操作之后，与执行下一个写操作之前的一段时间延迟（以秒为单位），或者在完成所有写操作之后，与设置**处理中**为 FALSE 之前的一段延迟时间（以秒为单位）。

如果不论什么原因致使写操作失败，则**写入成功**标志都要设置为 FALSE。如果**失败退出**标志为 TRUE，则第一次写操作失败也将提前终止操作序列的运行。在这种情况下，要将这个序列表中后续表目中的**写入成功**都设置为 FALSE。如果写操作成功，设置**写入成功**为 TRUE。

12.8.9 操作文本属性

这个属性的类型是**字符串的 BACnet 数组[N]**类型，用文本字符串描述**当前值**的可能取值。字符串的内容不受限制。

12.9 设备对象类型（Device Object Type）

设备对象类型定义为一个标准对象，其属性表示 **BACnet 设备**的外部可见一致性代码。每个 **BACnet 设备**有且只有一个**设备**对象。每个**设备**对象由它的**对象标识符**属性确定，该属性在 **BACnet 设备**中乃至整个 BACnet 互联网中都是唯一的。

设备对象及其属性见表 12-11，本节中进行详细规范。

表 12-11　设备对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R

系统状态 System_Status	BACnet 系统状态 BACnetDeviceStatus	R
生产商名称 Vendor_Name	字符串 CharacterString	R
生产商标识符 Vendor_Identifier	16 位无符号整型 Unsigned16	R
型号名称 Model_Name	字符串 CharacterString	R
固件版本 Firmware_Revision	字符串 CharacterString	R
应用软件版本 Application_Software_Version	字符串 CharacterString	R
位置 Location	字符串 CharacterString	0
描述 Description	字符串 CharacterString	0
协议版本 Protocol_Version	无符号整型 Unsigned	R
协议一致类别 Protocol_Conformance_Class	无符号整型 Unsigned(1 .. 6)	R
协议服务支持 Protocol_Service_Supported	BACnet 服务支持 BACnetServiceSupported	R
协议对象类型支持 Protocol_Object_Types_Supported	BACnet 对象类型支持 BACnet ObjectTypesSupported	R
对象列表 Object_List	BACnet 对象标识符的 BACnet 数组 BACnetARRAY[N] of BACnetObjectIdentifier	R
最大 APDU 长度支持 Max_APDU_ Length_Accepted	无符号整型 Unsigned	R
分段支持 Segmentation_ Supported	BACnet 分段 BACnetSegmentation	R
虚拟终端类型支持 VT_Classes_Supported	BACnet 虚拟终端类列表 List of BACnetVTClass	0[1]
活动虚拟终端会话 Active_VT_Sessions	BACnet 虚拟终端会话列表 List of BACnetVTSession	0[1]
本地时间 Local_Time	时间 Time	0
本地日期 Local_Date	时期 Date	0
时差 UTC_Offset	整型 INTEGER	0
夏令时状态 Daylight_Savings_Status	布尔 BOOLEAN	0
APDU 分段超时	无符号整型 Unsigned	0[2]

APDU_Segment_Timeout		
APDU 超时 APDU_Timeout	无符号整型 Unsigned	R
APDU 重传次数 Number_Of_APDU_Retries	无符号整型 Unsigned	R
会话密钥列表 List_Of_Session_Keys	BACnet 会话密钥列表 List of BACnetSessionKey	O
时间同步容器 Time_Synchronization_Recipients	BACnet 容器列表 List of BACnet Recipient	O[3]
最大主节点数 Max_Master	无符号整型 Unsigned(1 .. 127)	O[4]
最大信息帧数 Max_Info_Frame	无符号整型 Unsigned	O[4]
设备地址捆绑 Device_Address_Binding	BACnet 地址捆绑列表 List of BACnet AddressBinding	R

[1] 虚拟终端类型支持属性与活动虚拟终端会话必需同时存在。如果 PICS 中有对 VT 服务的支持，那么这两个属性是必需的。

[2] 如果支持任何形式的分段，那么该属性是必需的。

[3] 如果 PICS 指出该设备是时间主设备（Time Master），那么该属性是必需的。如果存在，则该属性必须是可写的。

[4] 如果设备是 MS/TP 主节点，那么这些属性是必需的。

12.9.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。对于**设备**对象，它的对象标识符在整个 BACnet 互联网中是唯一的。

12.9.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.9.3 对象类型属性

这个属性的类型是 **BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**设备（DEVICE）**。

12.9.4 系统状态属性

这个属性的类型是 **BACnet 系统状态**类型，表示 **BACnet 设备**当前的物理状态和逻辑的状态。这个属性可以取如下值：

{**可操作的（OPERATIONAL），只读可操作的（OPERATIONAL_READ_ONLY），下载必需的**

（DOWNLOAD_REQUIRED），下载正在处理中（DOWNLOAD_IN_PROGRESS），不可操作的（NON_OPERATIONAL）｝

在一个具体设备中这些状态的具体含义，以及它们与设备内部其它操作的同步机制，或者与 BACnet 服务运行的同步机制的实现方式，都由生产商自行确定，本标准不作定义。

12.9.5 生产商名称属性

这个属性的类型是**字符串**类型，标明了 **BACnet 设备**的制造商。

12.9.6 生产商标识符属性

这个属性的类型是 **16 位整型**类型，是由 ASHRAE 分配的唯一的生产商标志代码，用于区别生产商对协议的专用扩展。详见第 23 节。

12.9.7 型号名称属性

这个属性的类型是**字符串**类型，由生产商确定的表示设备型号的字符串。

12.9.8 固件版本属性

这个属性的类型是**字符串**类型，由生产商分配的表示安装在设备中的固化软件修订版本号的字符串。

12.9.9 应用软件版本属性

这个属性的类型是**字符串**类型，表示设备中安装的应用软件版本号。字符串内容由生产商自行确定，但建议是日期时间标记、程序员姓名、主文件版本号等。

12.9.10 位置属性

这个属性的类型是**字符串**类型，表示设备的物理位置。

12.9.11 描述属性

这个属性的类型是**字符串**类型，是描述 **BACnet 设备**的功能或其它本地信息的可打印字符。

12.9.12 协议版本属性

这个属性的类型是**无符号整型**类型，表示 BACnet 设备支持的 BACnet 协议版本。每次修正都将版本号加 1，初始发布的版本号为 1。

12.9.13 协议一致性类别属性

这个属性的类型是**无符号整型**类型，取值范围为 1-6，表示设备支持的一组标准协议服务和对象类型。当设备的这个属性取某个值时，表示这个设备至少支持与这个取值所代表的一致性类别的规定相符合的一组标准服务、对象类型和属性，同时，设备的协议实现还可以支持附加的服务、对象类型和属性。所有标准服务和对象类型组在下面两个属性中规定。

12.9.14 协议服务支持属性

这个属性的类型是 BACnet **协议服务支持**类型，表示这个设备的协议实现所支持的标准协议服务。除了在**协议一致性类别**属性中规定的服务之外，设备的实现还可自由支持附加标准服务。

12.9.15 协议对象类型支持属性

这个属性的类型是 BACnet **协议对象类型支持**类型，表示设备协议实现所支持的标准对象类型。表示这个设备的协议实现所支持的标准协议服务。除了在**协议一致性类别**属性中规定的对象类型之外，设备的实现还可自由支持附加的标准的和非标准的对象类型，以及它们各自所拥有的属性。某个对象的所有属性可使用 ALL 的属性引用的**读多个属性**服务获取。

12.9.16 对象列表属性

这个属性的类型是 BACnet **对象标识符的** BACnet **数组** [N]类型，列出设备中可通过 BACnet 服务访问的所有对象标识符。

12.9.17 最大 APDU 长度支持属性

这个属性的类型是**无符号整型**类型，表示一个不可分割的 APDU 所能容纳的最大字节数。这个属性值不小于 50。这个属性值还受到底层数据链路技术的限制。参见第 6 节至 11 节。

12.9.18 分段支持属性

这个属性的类型是 BACnet **分段**类型，表示 BACnet **设备**是否支持报文分段、分段传输和分段接收。取值为：

{**支持报文分段**（SEGMENTED_BOTH），**支持传输分段**（SEGMENTED_TRANSMIT），**支持分段接收**（SEGMENTED_RECEIVE），**不分段**（ NO_SEGMENTATION） }

12.9.19 虚拟终端类型支持属性

这个属性的类型是 BACnet **虚拟终端类列表**类型，表示一组终端的一致性代码。为了区别不同类型的终端或不同类型的操作接口程序，BACnet **设备**可以支持多种形式的操作。至少支持 17.5 节中定义的“缺省终端”VT_class。

虚拟终端类型支持属性与**活动虚拟终端会话**必须同时存在。如果 PICS 中有对 VT 服务的支持，那么这两个属性是必需的。

12.9.20 活动虚拟终端会话属性

这个属性的类型是 BACnet **虚拟终端会话列表**类型，列表中每项包含**本地 VT 会话标识符**（Local VT Session Identifier）、**远程 VT 会话标识符**（Remote VT Session Identifier）和**远程 VT 地址**（Remote VT Address）。这个属性提供了网络可见的虚拟终端会话（简称 VT-Sessions）指示，而这些虚拟终端会话在任何时间里都是激活的。无论何时通过 VT-Open 服务创建虚拟终端会话，都要在**活动虚拟终端会话**列表中加入新的元素。类似的，无论何时终止虚拟终端会话，都要从**活动虚拟终端会话**列表中删除相应的信息。

虚拟终端类型属性与**活动虚拟终端会话**必需同时存在。如果 PICS 中有对 VT 服务的支持，那么这两个属性是必需的。

12.9.21 本地时间属性

这个属性的类型是**时间**类型，表示设备信息的时间。如果 BACnet **设备**没有时间和日期的任何信息，那么这个属性可以被忽略。

12.9.22 本地日期属性

这个属性的类型是**日期**类型，表示设备信息的日期。如果 BACnet **设备**没有时间和日期的任何信息，那么该属性可以被忽略。

12 .9.23 时差属性

这个属性的类型是**整型**类型，表示本地时间与国际标准时间的时差（以分为单位，-720 到+720）。0 度子午线以西的时区为正值，以东的时区为负值。

12 .9.24 夏令时状态属性

这个属性的类型是**布尔**类型，用 TRUE 或 FALSE 表示 BACnet **设备**所在地是否采用夏令时。

12.9.25 APDU 分段超时属性

这个属性的类型是**无符号整型**类型，表示 APDU 分段超时重传的等待时间（以毫秒为单位）。缺省值为 2000 毫秒。如果 APDU **重传次数**属性不为 0，那么这个属性值也不为 0。参见 5.3 节。如果支持任何形式的分段，那么这个属性是必须存在的。

为了实现可靠通信，建议所有互联设备的**设备**对象的 APDU **分段超时**属性取相同值。

12.9.26 APDU 超时属性

这个属性的类型是**无符号整型**类型，以毫秒表示对 APDU 超时重传的等待时间。对于允许修改该参数的设备，缺省值为 3000 毫秒。否则为 60000 毫秒。如果 **APDU 重传次数**属性不为 0，那么这个属性值也不为 0。参见 5.3 节。

为了实现可靠通信，建议所有互联设备的设备对象的 **APDU 超时**属性取相同值。

12.9.27 APDU 重传次数属性

这个属性的类型是**无符号整型**类型，表示 APDU 重传的最大次数。缺省值为 3。如果设备不允许重传，那么该属性值为 0。如果该属性取值大于 0，则 **APDU 超时**属性值为非 0 值。参见 5.3 节。

12.9.28 会话密钥列表属性

这个属性的类型是 **BACnet 会话密钥列表**类型，用于与有保密要求的 **BACnet 设备**进行通信的密钥表。除了作为“**密钥服务器（Key Server）**” 的设备之外，任何设备不可写也不可读这个属性。会话密钥由 56 位密钥和对等设备请求用来进行安全通信的 **BACnet 地址**组成。

12.9.29 时间同步容器属性

这个属性的类型是 **BACnet 容器列表**类型，用于控制和限制使用**时间同步**服务的设备。如果该表的长度为 0，则设备禁止自动发送**时间同步**请求。如果列表长度不为 0，则设备可以自动发送**时间同步**请求，但只能向所列出的设备或地址发送。如果这个属性存在，则这个属性是可写的。如果 PICS 指出该设备是**时间主设备**，那么应具有这个属性。

12.9.30 最大主节点数属性

这个属性的类型是**无符号整型**类型，如果设备是 MS/TP 网络的主节点，该属性存在，表示最大主节点数（小于等于 127）。如果该属性不能通过 BACnet 服务写入，那么它的值将为 127。参见 9.5.3 节。

12.9.31 最大信息帧数属性

这个属性的类型是**无符号整型**类型，如果设备是 MS/TP 网络的节点，该属性存在，表示节点在占有令牌时能够发送的信息帧的最大数量。如果该属性不可写，或用户不可配置时，它的值应为 1。参见 9.5.3 节。

12.9.32 设备地址捆绑属性

这个属性的类型是 **BACnet 地址捆绑列表**类型，列表的表目是 BACnet **设备**对象的**对象**

标识符和 **BACnet 地址**。列表中的每一项确定了实际设备的地址，这些地址用于通过 BACnet 服务请求访问远程设备。如果对象没有设备标识符和设备地址捆绑时，列表为空。

12.10 事件登记对象类型（Event Enrollment Object Type）

事件登记对象类型定义为一个标准对象，表示 BACnet 系统内管理事件的信息。“事件”是指满足预先规定条件的所有对象的任何属性值的变化。**事件登记**对象主要用于定义一个事件和提供在事件发生与通告消息向一个或多个接收者进行传输这两者之间的联系。

事件登记对象包含有：事件类型描述，事件发生时须确定的参数，和时间发生时必须通告的设备。一个**通告类**对象也可以用于确定事件通告的接收者。如果一个设备是通告的接收者或者是由**事件登记**对象所引用的**通告类**对象中的接收者，那么该设备被登记为事件通告设备。

第 13 节描述了**事件登记**对象与**报警**和**事件**应用服务间的相互作用。**事件登记**对象及其属性见表 12-1，本节中进行详细规范。

表 12-12 事件登记对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
描述 Description	字符串 CharacterString	O
事件类型 Event_Type	BACnet 事件类型 BACnet EventType	R
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	R
事件参数 Event_Parameters	BACnet 事件参数 BACnet EventParameter	R
对象属性引用 Object_Property_Reference	BACnet 对象属性引用 BACnetObjectPropertyReference	R
事件状态 Event_State	BACnet 事件状态 BACnet EventState	R
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	R
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	R
通告类 Notification_Class	无符号整型 Unsigned	O[1]
接收者 Recipient	BACnet 接收者 BACnetRecipient	O[2]
进程标识符 Process_Identifier	无符号整型 Unsigned	O[2]

优先级 Priority	无符号整型 Unsigned	0[2]
发送确认的通告 Issue_Confirmed_Notifications	布尔 BOOLEAN	0[2]

[1] 如果接收者属性不存在，那么通告类属性必须存在。

[2] 如果通告类属性不存在，那么所有这些属性必须存在。

12.10.1 对象标识符属性

这个属性的类型是 BACnet **对象标识符**类型，是一个用来标识对象的数值代码。**对象标识符**在设备中是唯一的。

12.10.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 BACnet **设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.10.3 对象类型属性

这个属性的类型是 BACnet **对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**事件登记（EVENT_ENROLLMENT）**。

12.10.4 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容未作规定。

12.10.5 事件类型属性

这个属性的类型是 BACnet **事件类型**类型，表示事件算法类型，这个算法用来侦测事件的发生并且向事件登记设备报告。**事件类型**属性是一个枚举类型，可以取下列值：

{**比特位串改变，状态改变，值改变，命令失败，偏移阈值，偏离范围**}

事件算法，参数表和事件类型间存在一定的关系。**事件类型**属性表示一个用于侦测事件状态的算法，对于每个**事件类型**的算法在第 13 节中定义。**事件参数**属性提供算法所需的参数。

表 12－13 列出了**事件类型**、**事件状态**、**事件参数**的有效组合。

表 12-13 **事件类型**、**事件状态**和它们的参数

事件类型	事件状态	事件参数
比特位串改变 CHANG_OF_BITSTRING	正常 NORMAL 异常 OFFNORMAL	时间延迟（Time_Delay） 比特掩码（Bitmask）

		比特位串值列表 （List_Of_Bitstring_Values）
状态改变 CHANGE_OF_STATE	正常 NORMAL 异常 OFFNORMAL	时间延迟（Time_Delay） 值列表（List_Of_Values）
值改变 CHANGE_OF_VALUE	正常 NORMAL 异常 OFFNORMAL	时间延迟（Time_Delay） 比特掩码（Bitmask） 引用的属性增量 （ Referenced_Property_Increment）
命令失败 COMMAND_FAILURE	正常 NORMAL 异常 OFFNORMAL	时间延迟（Time_Delay） 反馈属性引用 （ Feedback_Property_Reference）
偏移阈值 FLOATING_LIMIT	正常 NORMAL 高阈值 HIGH_LIMIT 低阈值 LOW_LIMIT	时间延迟（Time_Delay） 设置点引用 （Setpoint_Reference） 低偏离阈值（Low_Diff_Limit） 高偏离阈值（Hi_Diff_Limit） 阈值宽度（Deadband）
偏离范围 OUT_OF_RANGE	正常 NORMAL 高阈值 HIGH_LIMIT 低阈值 LOW_LIMIT	时间延迟（Time_Delay） 低阈值（Low_Limit） 高阈值（High_Limit） 阈值宽度（Deadband）

12.10.6 通告类型属性

这个属性的类型是 BACnet **通告类型**类型，表示由**事件类型**属性定义的监测算法所产生的通告是**事件**类型，还是**报警**类型。

12.10.7 事件参数属性

这个属性的类型是 BACnet **事件参数**类型。每个**事件类型**属性规定用于监测引用对象的算法，而**事件参数**属性提供这个算法所需的参数值。**事件参数**中每个值的含义根据表 12-13 中**事件类型**属性确定。这些参数的意义如下：

比特掩码：这个参数的类型是**比特位串**类型，应用于引用属性为**比特位串**类型的**比特位串改变**事件算法和**值改变**事件算法。这是一个位掩码，用于指示算法所引用的属性位。比特位

为 1 表明引用属性中的这个比特位被算法监测。比特位为 0 表明引用属性中的这个比特位对侦测**比特流改变**或**值改变**事件不起作用。

比特位串值列表：这个参数是一个比特位串的列表，应用于**比特位串改变**事件算法。这个比特位串列表定义了引用属性为**异常**时的一组状态。只有**比特掩码**指定的比特位才是有意义的。如果引用属性值改变成为**比特位串值列表**中的某个值时，**事件登记**对象的**事件状态**属性将转换成为**进入异常**，并产生相应的通告。

值列表：这个参数是一个 BACnet **属性状态**的列表，应用于**状态改变**事件算法。这个事件算法应用于有离散值或枚举值的引用属性。**值列表**为这个属性的子集。如果引用属性值变为**值列表**中的某一值，**事件登记**对象的**事件类型**属性值变为**进入异常**，并产生相应的通告。

引用的属性增量：这个参数是**实数**类型，应用于**值改变**事件算法。表示要产生一个事件时引用属性的变化量。

时间延迟：这个参数是**无符号整型**类型，表示用秒为单位的时间，应用于所有的事件类型。表示产生事件通告时，由事件算法监测的条件必需保持的时间。

反馈属性引用：这个参数是 BACnet **对象属性引用**类型，应用于**命令失败**算法。这个参数标识那些可以提供反馈值从而保证命令属性已经改变了它的值的对象和属性。这个属性仅引用有枚举值或是布尔类型的对象属性。

设置点引用： 这个参数是 BACnet **对象属性引用**类型，应用于**偏移阈值**事件算法。表示引用属性间隔的设置点引用。

阈值宽度、高偏离阈值、低偏离阈值、高阈值、低阈值：这些参数都是**实数**类型参数，应用于**偏移阈值**和**偏离范围**事件算法。它们的用法见 13 节中的有关的算法说明。

12.10.8 对象属性引用属性

这个属性的类型是 BACnet **对象属性引用**类型。这个属性指向被**事件登记**对象所引用的那个对象和属性。将由**事件类型**属性所规定的算法应用于引用的属性，可以确定事件的**事件类型**。

12.10.9 事件状态属性

这个属性的类型是 BACnet **事件状态**类型，表示事件的当前状态。这个属性的允许值对

于不同的**事件类型**是不同的。参见表 12-13。**事件状态**属性的值与**事件使能**标志无关。参见 12.10.10 节。

12.10.10 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示允许产生**进入异常**、**进入故障**和**进入正常**事件。当标志位被设置时，表示相应的事件可以向所有登记设备发送通告。当标志位清除时，表示不会报告相应的事件。不论**事件使能**属性值为多少，对象的**事件状态**属性都可以不断更新。

12.10.11 确认转换属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。

12.10.12 通告类属性

这个属性的类型是**无符号整型**类型，缺省引用含有**事件登记**对象的设备中的**通告类**对象。这个属性用来规定一个或多个对于事件起始的对象的事件处理，事件报告和通告确认的方法。如果**通告类**属性不存在，则应该存在**接收者**属性、**进程标识符**属性、**优先级**属性、和**发送确认的通告**属性。如果这个属性存在，但**接收者**属性为非空值（表示**接收者**属性正用于通告）时，该属性可以为任何值，并且不应使用该属性。

12.10.13 接收者属性

这个属性的类型是 BACnet **接收者**类型，表示不使用**通告类**对象时，登记**事件通告**的接收设备。如果这个属性存在，则**进程标识符**属性、**优先级**属性和**发送确认的通告**属性也应存在。如果这个属性存在，并且**通告类**属性用于通告时，那么这个属性应为空值。如果这个属性为非空值（表示一个接收者正用于通告），那么**通告类**属性可以为任何值，并且不应使用本属性。

12.10.14 进程标识符属性

这个属性的类型是**无符号整型**类型。在不使用**通告类**对象时，本属性表示驻留在登记为事件通告的接收者设备中的进程的数字句柄。如果**进程标识符**属性存在，则**接收者**属性、**优先级**属性和**发送确认的通告**属性也应存在。如果**进程标识符**属性存在且**接收者**属性为空值（表示**通告类**属性正用于通告），那么**进程标识符**属性可以为任何值，并且不应使用**进程标识符**属性。

12.10.15 优先级属性

这个属性的类型是**无符号整型**类型，表示在不使用**通告类**对象的情况下，产生事件通告的优先级。优先级是保证有严格时刻要求的事件或报警通告不会出现不必要的延迟。本属性取值范围为 0-255。数字越小，优先级越高。如果本属性存在，那么**接收者**属性、**进程标识符**属性和**发送确认的通告**属性也应存在。如果本属性存在且**接收者**属性为空值（表示**通告类**属性正用于通告），那么本属性可以为任何值，并且不应使用本属性。

12.10.16 发送确认的通告属性

这个属性的类型是**布尔**类型，表示在不使用**通告类**对象的情况下，证实（TRUE）或不证实（FALSE）事件通告。如果本属性存在，那么**接收者**属性、 **进程标识符**属性和**优先级**属性也应存在。如果本属性存在且**接收者**属性为空值（表示**通告类**属性正用于通告），那么本属性可以为任何值，并且不应使用本属性。

12.11 文件对象类型（File Object Type）

文件对象类型定义为一个标准对象，用于定义可以通过**文件服务**（见第 14 节）访问的数据文件的属性。**文件对象**对象及其属性见表 12-14，本节中进行详细规范。

表 12-14　文件对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
描述 Description	字符串 CharacterString	O
文件类型 File_Type	字符串 CharacterString	R
文件大小 File_Size	无符号整型 Unsigned	R
修改日期 Modification_Date	BACnet 日期时间 BACnetDateTime	R
存档 Archive	布尔 BOOLEAN	W
只读 Read_Only	布尔 BOOLEAN	R
文件访问方法 File_Access_Method	BACnet 文件访问方法 BACnetFileAccessMethod	R

12.11.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的

12.11.2 对象名称属性

这个对象的类型是**字符串**类型，表示对象名称，在有这个对象的 BACnet **设备**中是唯一的。字符串最小长度为 1，而且字符必需为可打印字符。

12.11.3 对象类型属性

这个对象的类型是 BACnet **队形类型**类型，表示对象本身的类型名称。在本对象中，其值为**文件（FILE）**。

12.11.4 描述属性

这个对象的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容未作规定。

12.11.5 文件类型属性

这个属性的类型是**字符串**类型，表明文件用途。下列文本字符串是 ASHRAE 定义的文件结构所保留的：EVENTLOG 和 VALUELOG。

12.11.6 文件大小属性

这个属性的类型**无符号整型**类型，以字节表明文件数据的大小。

12.11.7 修改日期属性

这个属性的类型是 BACnet **日期时间**类型，表示本对象最后被修改的时间。当**文件**对象创建或写入时可以认为被修改。

12.11.8 存档属性

这个属性的类型是**布尔**类型，表明**文件**对象是否被保存或备分。只有在从**文件**对象被最后一次存档时刻以来，文件数据一直没有被任何内部程序或者**文件访问服务**修改过，本属性值才能为 TRUE。

12.11.9 只读属性

这个属性的类型是**布尔**类型，表示是（TURE）否（FALSE）不允许使用 BACnet **基本写文件**服务修改文件数据。

12.11.10 文件访问方法属性

这个属性的类型是 BACnet **文件访问方法**类型，表示本对象支持的文件访问的类型。本属性的可能取值为：

{**记录访问（RECORD_ACCESS）、流访问（STREAM_ACCESS）、记录和流访问**

（RECORD_AND_STREAM_ACCESS）}

12.12 组对象类型（Group Object Type）

组对象类型定义为一个标准对象，其属性表示一个其它对象的集合以及这些对象的一个或多个属性。**组**对象提供一种快速的方式，可以一次确定组的成员，从而简化 **BACnet 设备**间的信息交换。一个组对象可以是任何对象类型的组合。**组**对象及其属性见表 12-15 ，本节中进行详细规范。

表 12-15 组对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
描述 Description	字符串 CharacterString	O
组成员列表 List_Of_Group_Members	读访问规范列表 List of ReadAccessSpecification	R
当前值 Present_Value	读访问结果列表 List of ReadAccessResult	R

12.12.1 对象标识符塑像

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的

12.12.2 对象名称属性

这个对象的类型是**字符串**类型，表示对象名称，在有这个对象的 **BACnet 设备**中是唯一的。字符串最小长度为 1，而且字符必需为可打印字符。

12.12.3 对象类型属性

这个对象的类型是 **BACnet 队形类型**类型，表示对象本身的类型名称。在本对象中，其值为**组（GROUP）**。

12.12.4 描述属性

这个对象的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容未作规定。

12.12.5 组成员列表属性

这个属性的类型是**读访问规范列表**类型，是一个或多个读访问规范表。在一个协议事务处理中，如果定义了**组**对象，则本属性用来定义将要被引用的组的成员。每个读访问规范由两部分组成：1）一个**对象标识符**，2）一个**属性引用表**。所有组成员都应该是同一设备中的对象。参见第 21 节有关 ReadAccessSpecification 的 ASN.1 说明。

不允许**组**对象嵌套，即本属性不能引用**组**对象的**当前值**属性。

12.12.6 当前值属性

这个属性的类型是**读访问结果列表**类型，是**组成员列表**属性中定义的所有属性值的表，为“只读”属性。本属性不能用于向组成员写入一组值。在每次采用取出成员属性的方法进行读属性操作之后，都应该重构**当前值**属性列表。（注：这个规定是为了减少因存储**当前值**而可能产生的并发问题。）

12.13 环对象类型（Loop Object Type）

环对象类型定义为一个标准对象，其属性表示任何形式的反馈控制环路的外部可见一致性代码。**环**对象通过提供三个独立的无单位增益常数，可以具有广泛的适用性。每个增益常数由控制算法具休确定，如何使用不同的算法确定增益常数的方法，由生产商自行确定。**环**对象及其属性见表 12-16，本节中进行详细规范。图 12-1 示出**环**对象属性与该对象引用的其它对象间的关系。

表 12-16 环对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	实数 REAL	R
描述 Description	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R
事件状态 Event_State	BACnet 事件状态 BACnet EventState	R
可靠性 Reliability	BACnet 可靠性 BACnetReliability	O
脱离服务 Out_Of_Service	布尔 BOOLEAN	R
更新间隔 Update_Interval	无符号整型 Unsigned	O
输出单位 Output_Units	BACnet 工程单位 BACnetEngineeringUnits	R
操作变量引用 Manipulated_Variable_Referencet	BACnet 对象属性引用 BACnetObjectPropertyReference	R
控制变量引用 Controlled_Variable_Referencet	BACnet 对象属性引用 BACnetObjectPropertyReference	R
控制变量值 Controlled_Variable_Value	实数 REAL	R
控制变量单位 Controlled_Variable_Units	BACnet 工程单位 BACnetEngineeringUnits	R
设置点引用 Setpoint_Reference	BACnet 设置点引用 BACnetSetpointReference	R
设置点 Setpoint	实数 REAL	R
操作 Action	BACnet 操作 BACnetAction	R

比例常数 Proportional_Constant	实数 REAL	0[1]
比例常数单位 Proportional_Constant_Units	BACnet 工程单位 BACnetEngineeringUnits	0[1]
积分常数 Integral_Constant	实数 REAL	0[2]
积分常数单位 Integral_Constant_Units	BACnet 工程单位 BACnetEngineeringUnits	0[2]
微分常数 Derivative_Constant	实数 REAL	0[3]
微分常数单位 Derivative_Constant_Units	BACnet 工程单位 BACnetEngineeringUnits	0[3]
偏差 Bias	实数 REAL	0
最大输出 Maximum_Output	实数 REAL	0
最小输出 Minimum_Output	实数 REAL	0
写入优先级 Priority_For_Writing	无符号整型 Unsigned	R
COV 增量 COV_Increment	实数 REAL	0[4]
时间延迟 Time_Delay	无符号整型 Unsigned	0[5]
通告类 Notification_Class	无符号整型 Unsigned	0[5]
误差阈值 Error_Limit	实数 REAL	0[5]
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	0[5]
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	0[5]
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	0[5]

[1] 如果这些可选属性中的一个存在，那么必须都存在。

[2] 如果这些可选属性中的一个存在，那么必须都存在。

[3] 如果这些可选属性中的一个存在，那么必须都存在。

[4] 如果对象支持 COV 报告，那么该属性是必需的。

[5] 如果对象支持内部报告，则这些属性是必需的。

12.13.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.13.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.13.3 对象类型属性

这个属性的类型是 BACnet **对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**环（LOOP）**。

12.13.4 当前值属性

这个属性的类型是**实数**类型，本属性以**输出单位**属性定义的单位表示环算法的当前输出值。

12.13.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.13.6 状态标志属性

这个属性的类型是 BACnet **状态标志**类型，表示四种布尔标志，这些标志表示环对象运行的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中**，**故障**，**管制**，**脱离服务**}

其中，

报警中 如果**事件状态**（见 12.13.7 节）属性值为**正常**则为 FALSE(0)，否则为 TRUE(1) 。

故障 如果**可靠性**（见 12.13.8 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制 如果某值被与 BACnet **设备**本身的有关机制所管制，则为 TRUE(1)。在此情况下，“管制”表示**当前值**和**可靠性**属性值不再能够通过 BACnet 服务而改变。

脱离服务 如果**脱离服务**属性值（见 12.13.9 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.13.7 事件状态属性

这个属性的类型是 BACnet **事件状态**类型，用于确定本对象是否有与之有关的活动事件状态。如果对象支持内部报告，这个属性表示对象的事件状态。如果对象不支持内部报告，这个属性的值为**正常**。

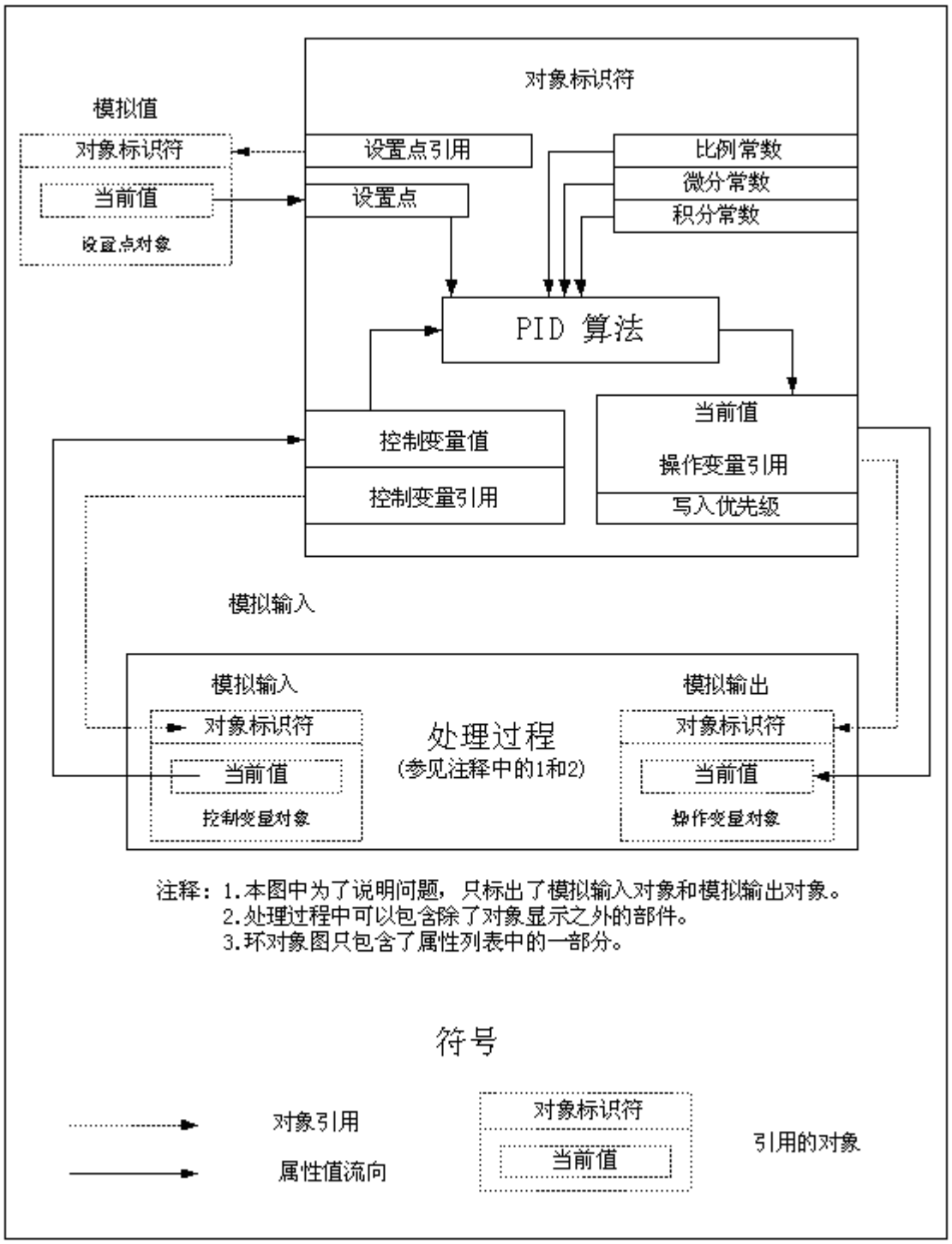


图12-1 环对象结构和所引用的对象

12.13.8 可靠性属性

这个属性的类型是 BACnet **可靠性**类型，表示环的**当前值**是否可靠。如果不可靠，指明原因。本属性可能取值如下：

〔**未发现故障，开路，其它不可靠**〕

12.13.9 脱离服务属性

这个属性的类型是**布尔**类型，表示本对象所表示的算法是（TRUE）否（FALSE）能够使用。

12.13.10 更新间隔属性

这个属性的类型是**无符号整型**类型，用微秒表示环对象算法更新输出值（**当前值**属性）的间隔。

*注：本对象没有定义过程变量取样或算法运行时间间隔参数的属性。**更新间隔**属性值在不同算法中可能与上述参数相同，也可能不相同。取样或运行间隔时间由生产商自行确定，因而没必要在本对象中定义。*

12.13.11 输出范围属性

这个属性的类型是 BACnet **工程单位**类型，表示控制环对象输出（**当前值**属性）的工程单位。

12.13.12 操作变量引用属性

这个属性的类型是 BACnet **对象属性引用**类型。**操作变量引用**属性指向控制环对象输出（**当前值**属性）写入的对象和属性，这个引用通常是用于定位一个设备的**模拟输出**对象的**当前值**属性，也可以是诸如阶段多态的**二进制输出**对象。

12.13.13 控制变量引用属性

这个属性的类型是 BACnet **对象属性引用**类型，**控制变量引用**属性指向**环**对象中用来设置**控制变量值**属性值的属性。这个引用通常是用于过程测量的参数（如温度）的**模拟输出**对象的**当前值**属性，也可以是诸如阶段多态的**二进制输出**对象。

12.13.14 控制变量值属性

这个属性的类型是**实数**类型，是由**控制变量引用**属性所引用的对象的属性的值。控制环对象通过比较本属性与**设置点**属性来计算误差。

12.13.15 控制变量单位属性

这个属性的类型是 BACnet **工程单位**类型，表示**环**对象的**控制变量值**属性的工程单位。

12.13.16 设置点引用属性

这个属性的类型是 BACnet **设置点引用**类型，是一个长度为 0 或 1 的引用表。长度为 0

表示控制环对象的设置点是固定的，并且设置在这个对象的**设置点**属性内。长度为 1 表示在一个对象的属性中包含有为这个环对象设置的设置点值，并且定义了那个属性。

12.13.17 设置点属性

这个属性的类型是**实数**类型，用**控制变量单位**属性定义的单位，表示**环**对象设置点的值，或者表示由**设置点引用**属性所引用对象的属性值。

12.13.18 操作属性

这个属性的类型是 BACnet **操作**类型，定义环对象是**直接**（DIRECT）操作还是**逆向**（REVERSE）操作。

12.13.19 比例常数属性

这个属性的类型是**实数**类型，是环对象算法的比例增益参数的值。本属性表示在比例控制模式中的任何形式的增益，如，总增益量，节流范围等。**比例常数**属性与**比例常数单位**属性必须同时存在。

12.13.20 比例常数单位属性

这个属性的类型是 BACnet **工程单位**类型，表示本对象的**比例常数**属性的工程单位。**比例常数**属性与**比例常数单位**属性必须同时存在。

12.13.21 积分常数属性

这个属性的类型是**实数**类型，是环对象算法的积分参数的值。本属性表示在积分控制模式的任何形式的增益量，如，复位时间，速率等。**积分常数**属性与**积分常数单位**属性必须同时存在。

12.13.22 积分常数单位属性

这个属性的类型是 BACnet **工程单位**类型，表示本对象的**积分常数**属性的工程单位。**积分常数**属性与**积分常数单位**属性必须同时存在。

12.13.23 微分常数属性

这个属性的类型是**实数**类型，是环对象算法的微分参数的值。本属性表示在微分控制模式的任何形式的增益量，如，微分时间，速率时间等。**微分常数**属性与**微分常数单位**属性必须同时存在。

12.13.24 微分常数单位属性

这个属性的类型是 BACnet 工程单位类型，表示本对象的**微分常数**属性的工程单位。**微分常数**属性与**微分常数单位**属性必须同时存在。

12.13.25 偏差属性

这个属性的类型是**实数**类型，用输出**单位属性**中的单位表示环对象算法的偏差值。

12.13.26 最大输出属性

这个属性的类型是**实数**类型，是被 PID 算法所限制的**当前值**属性的最大值。这个属性通常用于防止算法的控制行为超出控制设备的范围，以及防止发生“停摆”现象。

12.13.27 最小输出属性

这个属性的类型是**实数**类型，是被 PID 算法所限制的**当前值**属性的最小值。这个属性通常用于防止算法的控制行为超出控制设备的范围，以及防止发生“停摆”现象。

12.13.28 写入优先级属性

这个属性的类型是**无符号整型**类型。**环**对象可能用于控制一个对象的可命令属性，本属性提供了一个可供命令优先级机制使用的优先级的值。本属性在这个环对象控制的**控制变量引用**属性中的**优先级数组（Priority_Array）**中定义了一个范围，取值范围为 1-16。

12.13.29 COV 增量属性

这个属性的类型是**实数**类型，定义**当前值**属性的最小改变值，这个最小改变值将导致向 COV 客户发布 **COV 通告**。如果对象支持 COV 报告，则必须具备这个属性。

12.13.30 时间延迟属性

这个属性的类型是**无符号整型**类型。定义从**(设置点**属性值 － **控制变量值**属性值)（即误差（Error））处于由**误差阈值**属性确定的范围之外开始，到生成一个**进入异常**事件之间的最小时间间隔（以秒为单位），或者从**(设置点**属性值 － **控制变量值**属性值)（即误差）处于由**误差阈值**属性确定的范围之内时，到生成一个**进入正常**事件的最小时间间隔（以秒为单位）。如果对象支持内部报告则必须具备这个属性。

12.13.31 通告类属性

这个属性的类型是**无符号整型**类型，用于定义这个对象处理和生成事件通告时的通告类别。这个属性缺省引用有相同**通告类**属性值的**通告类**对象。如果对象支持内部报告则必须具备这个属性。

12.13.32 误差阈值属性

这个属性的类型是**实数**类型，定义生成**进入异常**事件时，**(设置点**属性值 － **控制变量值**属性值) （即误差）必须超过的绝对值。如果对象支持内部报告，则该属性是必需的。

12.13.33 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示使能或者禁止对于**进入异常**、**进入故障**和**进入正常**事件的报告。如果对象支持内部报告则必须具备这个属性。

12.13.34 确认转换属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。这些标志将在相应事件出现的情况下被清除，并在下列任一条件下设置：

(a) 收到相应的确认；

(b) 如果**事件使能**属性中相应的标志未设置时，事件发生（即在这种情况下，不会产生事件通告，因此也不会产生确认）；

(c) 如果**事件使能**属性中设置了相应的标志，并且在通告类对象的**确认转换**属性中未设置相应标志时，事件发生（即无确认产生）。

如果对象支持内部报告则必须具备这个属性。

12.13.35 通告类型属性

这个属性的类型是 BACnet **通告类型**类型，表示对象生成的通告是**事件**还是**报警**。如果对象支持内部报告则必须具备这个属性。

12.14 多态输入对象类型（Multi-state Input Object Type）

多态输入对象类型定义了一个标准对象，它的**当前值**属性表示对象驻留的 BACnet **设备**内算法处理的结果。算法处理方法由生产商自行确定，本协议中未作定义。例如，**多态输入**对象的状态或**当前值**可能是多个二进制输入的逻辑组合，或者是多个模拟输入的阈值，或者是一个数学计算的结果。**当前值**属性是一个表示状态的整型数。**状态文本**属性对每个状态进行了描述。**多态输入**对象及其属性归纳在表 12-17 中。

*注：不要混淆**当前值**状态与**事件状态**属性（它反映了多态输入的异常状态）。*

表 12-17　多态输入对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R

对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	无符号整型 Unsigned	R[1]
描述 Description	字符串 CharacterString	O
设备类型 Device_Type	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R
事件状态 Event_State	BACnet 事件状态 BACnet EventState	R
可靠性 Reliability	BACnet 可靠性 BACnetReliability	O
脱离服务 Out_Of_Service	布尔 BOOLEAN	R
状态数 Number_of_States	无符号整型 Unsigned	R
状态文本 State_Text	字符串的 BACnet 数组 BACnetARRAY[N] of CharacterString	O
时间延迟 Time_Delay	无符号整型 Unsigned	O[2]
通告类 Notification_Class	无符号整型 Unsigned	O[2]
报警值 Alarm_Values	无符号整型列表 List of Unsigned	O[2]
故障值 Fault_Values	无符号整型列表 List of Unsigned	O[2]
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	O[2]
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	O[2]
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	O[2]

[1] 当**脱离服务**为 TRUE 时该属性必需是可写的。

[2] 如果对象支持内部报告则该属性是必需的。

12.14.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.14.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.14.3 对象类型属性

这个属性的类型是 **BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**多态输入（MULTISTATE_INPUT）**。

12.14.4 当前值属性

这个属性的类型是**无符号整型**类型，反映了输入的逻辑状态。逻辑状态为 n 个状态之一，n 是**状态数**属性中定义的状态数目。当前状态的确定方法由生产商自行确定。**当前值**属性值应该总是大于 0。当**脱离服务**为 TRUE 时，本属性必须是可写的 。参见 12.14.10 节。

12.14.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.14.6 设备类型属性

这个属性的类型是**字符串**类型，是一段对多态输入的文本描述。通常用于描述与多态输入对应的设备的型号。

12.14.7 状态标志属性

这个属性的类型是 BACnet **状态标志**（BACnetStatusFlags）类型，表示四种布尔标志，这些标志表示多态输入设备运行时刻的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中**，**故障**，**管制**，**脱离服务**}

其中，

报警中　　如果**事件状态**（见 12.14.8 节）属性值为**正常**（**NORMAL**）则为 FALSE(0)，否则为 TRUE(1) 。

故障　　如果**可靠性**（见 12.14.9 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制　　如果某值被与 BACnet **设备**本身的有关机制所管制，则为 TRUE(1)。在此情况下，“管制”表示**当前值**和**可靠性**属性值不再随物理设备的输入变化而变化。

脱离服务　　如果**脱离服务**属性值（见 12.14.10 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.14.8 事件状态属性

这个属性的类型是 BACnet **事件状态**类型，用于检测对象是否处于事件活动状态。如果对象支持内部报告，这个属性表示对象的事件状态。如果对象不支持内部报告，这个属性的

值为**正常**。

12.14.9 可靠性属性

这个属性的类型是 BACnet **可靠性**类型，表示**当前值**或物理输入设备的运行是否“可靠”，如果不可靠，则指明原因。这个属性有下列值：

{**未发现故障**，**无传感器**，**超出范围**，**低于范围**，**开路**，**短路**，**其它不可靠**}

12.14.10 脱离服务属性

这个属性的类型是**布尔**类型，表示该对象代表的物理输入是（TRUE）否（FALSE）不与设备相关。当**脱离服务**为 TRUE 时，**当前值**属性与物理输入设备分离，并不会随着物理输入设备的改变而变化。当**脱离服务**为 TRUE 时，**可靠性**属性和对应的**状态标志**属性中的**故障**标志的状态也与物理输入设备分离。当**脱离服务**为 TRUE 时，当用于模拟专门的固定条件或用于测试时，**当前值**与**可靠性**属性可以为任何值。而其它依靠**当前值**或**可靠性**属性的功能将响应这些变化，就像这些变化发生在输入中一样。

12.14.11 状态数属性

这个属性的类型是**无符号整型**类型，定义了**当前值**可能具有的状态数目。本属性取值总大于 0。

12.14.12 状态文本属性

这个属性是一个由字符串构成的 BACnet **数组**，描述所有的**当前值**的可能状态。描述的数目要与**状态数**属性中定义的数目一致。**当前值**中的整型值作为数组的索引。

12.14.13 时间延迟属性

这个属性的类型是**无符号整型**类型，定义从**当前值**属性等于**报警值**属性开始，到生成一个**进入异常**事件之间的最小时间间隔（以秒为单位），或者从**当前值**属性不等于**报警值**属性开始，到生成一个**进入正常**事件的最小时间间隔（以秒为单位）。如果对象支持内部报告则必须具备这个属性。

12.14.14 通告类属性

这个属性的类型是**无符号整型**类型，用于定义这个对象处理和生成事件通告时的通告类别。这个属性缺省引用有相同**通告类**属性值的**通告类**对象。如果对象支持内部报告则必须具备这个属性。

12.14.15 报警值属性

这个属性是一个无符号整型列表，定义生成一个**进入异常**事件时，**当前值**属性必须等于的所有状态值。如果对象支持内部报告则必须具备这个属性。

12.14.15.1 生成进入异常事件条件

进入异常事件在达到下列三个条件的情况下生成：

(a) **当前值**属性必须等于至少一个**报警值**列表中的值，

(b) **当前值**必须等于这个报警值一段由**时间延迟**属性确定的最小时间间隔的时间，

(c) **事件使能**属性设置为**进入异常**标志。

12.14.15.2 生成进入正常事件条件

一旦**当前值**等于某个**报警值**之后，只有在**当前值**不等于任何一个报警值之后，才能生成**进入正常**事件。**进入正常**事件在达到下列两个条件的情况下生成：

(a) **当前值**必须不等于任何一个**报警值**属性中的值一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入正常**标志。

12.14.16 故障值属性

这个属性是一个无符号整型列表，定义生成一个**进入故障**事件时，**当前值**属性必须等于的所有状态值。如果对象支持内部报告则必须具备这个属性。

12.14.16.1 生成进入故障事件条件

进入异常事件在达到下列三个条件的情况下生成：

(a) **当前值**属性必须等于至少一个**故障值**列表中的值，

(b) **当前值**必须等于这个故障值一段由**时间延迟**属性确定的最小时间间隔的时间，

(c) **事件使能**属性设置为**进入故障**标志。

12.14.16.2 生成进入正常事件条件

一旦**当前值**等于某个**故障值**之后，只有在**当前值**不等于任何一个故障值之后，才能生成**进入正常**事件。**进入正常**事件在达到下列两个条件的情况下生成：

(a) **当前值**必须不等于任何一个**故障值**属性中的值一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入正常**标志。

12.14.17 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示使能或者禁止对于**进入异常**、**进入故障**和**进入正常**事件的报告。如果对象支持内部报告则必须具备这个属性。

12.14.18 确认转换属性

这个属性的类型是 BACnet **事件转换比特**类型,用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。这些标志将在相应事件出现的情况下被清除，并在下列任一条件下设置：

(a) 收到相应的确认；

(b) 如果**事件使能**属性中相应的标志未设置时，事件发生（即在这种情况下，不会产生事件通告，因此也不会产生确认）；

(c) 如果**事件使能**属性中设置了相应的标志，并且在通告类对象的**确认转换**属性中未设置相应标志时，事件发生（即无确认产生）。

如果对象支持内部报告则必须具备这个属性。

12.14.19 通告类型属性

这个属性的类型是 BACnet **通告类型**类型，表示对象生成的通告是**事件**还是**报警**。如果对象支持内部报告则必须具备这个属性。

12.15 多态输出对象类型（Multi-state Output Object Type）

多态输出对象类型定义了一个标准对象，它的属性表示这个对象驻留的 BACnet **设备**内的处理程序或一个或多个物理输出的期望状态。特定状态的实际功能由生产商自行确定，本协议中未作定义。例如，特定**当前值**属性是一个表示状态的无符号整型数。**状态文本**属性对每个状态进行了描述。**多态输出**对象及其属性归纳在表 12-18 中。本节进行详细规范。

表 12-18　多态输出对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	无符号整型 Unsigned	W
描述 Description	字符串 CharacterString	O
设备类型 Device_Type	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R

事件状态 Event_State	BACnet 事件状态 BACnet EventState	R
可靠性 Reliability	BACnet 可靠性 BACnetReliability	O
脱离服务 Out_Of_Service	布尔 BOOLEAN	R
状态数 Number_of_States	无符号整型 Unsigned	R
状态文本 State_Text	字符串的 BACnet 数组 BACnetARRAY[N] of CharacterString	O
优先级数组 Priority_Array	BACnet 优先级数组 BACnetPriorityArra	R
释放缺省 Relinquish_Default	无符号整型 Unsigned	R
时间延迟 Time_Delay	无符号整型 Unsigned	O[1]
通告类 Notification_Class	无符号整型 Unsigned	O[1]
反馈值 Feedback_Values	无符号整型 Unsigned	O[1]
事件使能 Event_Enable	BACnet 事件转换比特 BACnetEventTransitionBits	O[1]
确认转换 Acked_Transitions	BACnet 事件转换比特 BACnetEventTransitionBits	O[1]
通告类型 Notify_Type	BACnet 通告类型 BACnetNotifyType	O[1]

[1] 如果对象支持内部报告则该属性是必需的。

12.15.1 对象标识符属性

这个属性的类型是**BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的**BACnet 设备**中是唯一的。

12.15.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的**BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.15.3 对象类型属性

这个属性的类型是**BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**多态输出（MULTISTATE_OUTPUT）**。

12.15.4 当前值属性

这个属性的类型是**无符号整型**类型，反映了输出的逻辑状态。逻辑状态为 n 个状态之一，n 是**状态数**属性中定义的状态数目。当前状态的确定方法由生产商自行确定。**当前值**属性值应该总是大于 0。

12.15.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.15.6 设备类型属性

这个属性的类型是**字符串**类型，是一段对多态输出的文本描述。通常用于描述与多态输出对应的设备的型号。

12.15.7 状态标志属性

这个属性的类型是 BACnet **状态标志**类型，表示四种布尔标志，这些标志表示多态输出设备运行时刻的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中，故障，管制，脱离服务**}

其中，

报警中 如果**事件状态**（见 12.15.8 节）属性值为**正常（NORMAL）**则为 FALSE(0)，否则为 TRUE(1) 。

故障 如果**可靠性**（见 12.15.9 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制 如果某值被与 BACnet **设备**本身的有关机制所管制，则为 TRUE(1)。在此情况下，“管制”表示**当前值**和**可靠性**属性值不再反映物理输出的变化。

脱离服务 如果**脱离服务**属性值（见 12.15.10 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.15.8 事件状态属性

这个属性的类型是 BACnet **事件状态**类型，用于检测对象是否处于事件活动状态。如果对象支持内部报告，这个属性表示对象的事件状态。如果对象不支持内部报告，这个属性的值为**正常**。

12.15.9 可靠性属性

这个属性的类型是 BACnet **可靠性**类型，表示**当前值**或物理输出设备的运行是否“可靠”，如果不可靠，则指明原因。这个属性有下列值：

{**未发现故障，开路，短路，无输出，其它不可靠**}

12.15.10 脱离服务属性

这个属性的类型是**布尔**类型，表示该对象代表的物理输出或者处理结果是（TRUE）否（FALSE）不与设备相关。当**脱离服务**为 TRUE 时，**当前值**属性的变化与物理输出设备分离。当**脱离服务**为 TRUE 时，**可靠性**属性和对应的**状态标志**属性中的**故障**标志的状态也与物理输出设备分离。当**脱离服务**为 TRUE 时，当用于模拟专门的固定条件或用于测试时，**当前值**与**可靠性**属性可以为任何值。而其它依靠**当前值**或**可靠性**属性的功能将响应这些变化，就像这些变化发生在输出中一样。如果**脱离服务**为 TRUE，**当前值**属性还要受 BACnet 命令优先机制的控制。

12.15.11 状态数属性

这个属性的类型是**无符号整型**类型，定义了**当前值**可能具有的状态数目。本属性取值总大于 0。

12.15.12 状态文本属性

这个属性是一个由字符串构成的 **BACnet 数组**，描述所有的**当前值**的可能状态。描述的数目要与**状态数**属性中定义的数目一致。**当前值**中的整型值作为数组的索引。

12.15.13 优先级数组属性

这个属性时一个只读数组，包含对这个对象有效的优先命令。参见 19 节中对优先机制的描述。

12.15.14 释放缺省属性

当**优先值数组**中的所有命令优先值为 NULL 时，这个属性是用于**当前值**属性的缺省值。详见 19 节。

12.15.15 时间延迟属性

这个属性的类型是**无符号整型**类型，定义从**当前值**属性不等于**反馈值**属性开始，到生成一个**进入异常**事件之间的最小时间间隔（以秒为单位），或者从**当前值**属性等于**反馈值**属性开始，到生成一个**进入正常**事件的最小时间间隔（以秒为单位）。如果对象支持内部报告则必须具备这个属性。

12.15.16 通告类属性

这个属性的类型是**无符号整型**类型，用于定义这个对象处理和生成事件通告时的通告类别。这个属性缺省引用有相同**通告类**属性值的**通告类**对象。如果对象支持内部报告则必须

具备这个属性。

12.15.17 反馈值属性

这个属性是一个无符号整型列表，定义生成一个**进入异常**事件时，**当前值**属性必须不等于的所有状态值。如果对象支持内部报告则必须具备这个属性。确定**反馈值**的方法由生产商自行确定。

12.15.17.1 生成进入异常事件条件

进入异常事件在达到下列两个条件的情况下生成：

(a) **当前值**必须不等于**反馈值**一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入异常**标志。

12.15.17.2 生成进入正常事件条件

一旦**当前值**不等于**反馈值**之后，只有在**当前值**等于**反馈值**之后，才能生成**进入正常**事件。**进入正常**事件在达到下列两个条件的情况下生成：

(a) **当前值**必须等于**反馈值**属性的值一段由**时间延迟**属性确定的最小时间间隔的时间，

(b) **事件使能**属性设置为**进入正常**标志。

12.15.18 事件使能属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示使能或者禁止对于**进入异常**、**进入故障**和**进入正常**事件的报告。如果对象支持内部报告则必须具备这个属性。

12.15.19 确认转换属性

这个属性的类型是 BACnet **事件转换比特**类型，用三个标志分别表示收到对于**进入异常**、**进入故障**和**进入正常**事件的确认。这些标志将在相应事件出现的情况下被清除，并在下列任一条件下设置：

(a) 收到相应的确认；

(b) 如果**事件使能**属性中相应的标志未设置时，事件发生（即在这种情况下，不会产生事件通告，因此也不会产生确认）；

(c) 如果**事件使能**属性中设置了相应的标志，并且在通告类对象的**确认转换**属性中未设置相应标志时，事件发生（即无确认产生）。

如果对象支持内部报告则必须具备这个属性。

12.15.20 通告类型属性

这个属性的类型是 BACnet **通告类型**类型，表示对象生成的通告是**事件**还是**报警**。如果对象支持内部报告则必须具备这个属性。

12.16 通告类对象类型（Notification Class Object Type）

通告类对象类型定义了一个标准对象，表示在 BACnet 系统内事件通告发布所需的信息。**通告类**对事件起始的对象特别有用，它提供了如何处理通告、怎样规划通告目标、以及怎样获得确认的方式。

一个通告类规范事件通告根据**进入异常**、**进入故障**、**进入正常**事件在处理中如何实现优先机制，事件目录是否需要确认（几乎总是由操作员完成），以及哪个目标设备和处理程序应当接收通告这些事宜。

优先机制提供了一种方式，保证重要的报警或事件通告不会出现不必要的延迟。优先值取值范围是 0-255。较小的数值表示较高的优先级。在通告类中优先级可以分配给**进入异常**、**进入故障**、**进入正常**事件。

确认的目的是保证通告已被其它代理处理，而不仅仅是简单的被设备正确收到。在大多数情况下，确认来自操作员。在通告类中，**进入异常**、**进入故障**、**进入正常**事件可以要求单独确认也可以不要求。

经常需要将事件通告发送给多个目的地址，或者根据日期与时间的不同发送事件通告给不同的地址。**通告类**可以根据时间、日期和处理类型定义一个目的地址列表。目的地址定义了一组日期（从星期一到星期日），在这组日期中，这个目的地址对于**通告类**对象来说是可用的。另外，每个目的地址还定义了起止时间，表示在一周内这些天的这些时候，目的地址对于**通告类**对象可用。如果使用**通告类**对象的事件发生，并且发生的日期处于给定目的地址定义的有效日期内，而且发生的时间也处于给定目的地址所定义的时间窗口内，那么就会向该地址发送通告。目的地址可以进一步与**进入异常**、**进入故障**、**进入正常**事件传输相结合，满足具体的应用要求。

目的地址定义了接收通告并进行处理的接收设备。使用只对目标设备有意义的数字句柄来表示处理过程。对于这些句柄的管理由生产商自行确定。接收设备可以通过它独一无二的**设备对象标识符**或者 BACnet **地址**来确定。对于后者，将用到专门的节点地址、多播地址、广播地址。目的地址还可以进一步规定是否需要使用确认或不确认的事件通告。

通告类对象及其属性归纳在表 12-19 中。

表 12-19　通告类对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R

对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
描述 Description	字符串 CharacterString	O
通告类 Notification_Class	无符号整型 Unsigned	R
优先级 Priority	无符号整型的 BACnet 数组 BACnetARRAY[3] of Unsigned	R
确认要求 Acked_Required	BACnet 事件转换比特 BACnetEventTransitionBits	R
接收者列表 Recipient_List	BACnet 目标列表 List of BACnetDestination	R

12.16.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.16.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.16.3 对象类型属性

这个属性的类型是 **BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**通告类（NOTIFICATION_CLASS）**。

12.16.4 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.16.5 通告类属性

这个属性的类型是**无符号整型**类型，表示这个通告类的数字值。事件起始的对象将使用这个数字间接引用**通告类**对象。生产商可以选择使本属性值等于**通告类**对象的实例数。具体的实现方式由生产商自行确定。

12.16.6 优先级属性

这个属性的类型是**无符号整型的 BACnet 数组[3]** 类型，表示用于**进入异常、进入故障、进入正常**事件的事件通告的使用优先级。取值范围为 0-255。较低的数值表示较高的优先级。

12.16.7 确认要求属性

这个属性的类型是**BACnet 事件转换比特**类型，用三个标志分别表示对于**进入异常、进入故障、进入正常**事件转换所生成的通告，是否需要确认。

12.16.8 接收者列表属性

这个属性的类型是**BACnet 目标列表**类型，表示有一个或多个接收地址的列表，当事件起始的对象使用这个类探测到事件发生时，通告将被发送到列表中的目的地址。目的地址定义了一个参数结构，见表 12-20。

表 12-20　BACnet 目标列表的组成部分

参数	类型	描述
有效日期 Valid Days	BACnetDaysOfWeek	目的地址可用的日期（用星期表示）
起止时间 From time ToTime	Time	目的地址在可用日期内的可用时间（用时间窗口表示）
接收者 Recipient	BACnetRecipient	接收通告的目的设备
进程标识符 Process Identifier	Unsigned	接收通告的设备内的处理进程的句柄
产生确认的通告 Issue Confirmed Notifications	Boolean	发送需确认通告时为 TRUE 发送无需确认通告时为 FALSE
转换 Transitions	BACnetEventTransitions	三个标识{**进入异常、进入故障、进入正常**}转换的标志，表示接收者适合的传输

12.17 程序对象类型（Program Object Type）

程序对象类型定义了一个标准对象，它的属性表示应用程序的外部可视一致性代码。在本协议中，应用程序是指对一个在 **BACnet 设备**中的处理过程的抽象表示，这个处理过程执行一个指令集，对某个数据结构集合进行操作。这些指令集的运行逻辑以及这些数据结构的形式和内容都不是本协议的规范内容，由生产商自行确定。

程序对象的属性提供了一个应用程序的一些参数的表征。本标准中定义了一些属性，这些属性表示了在不同的 **BACnet 设备**之间的一致行为。通过这些标准的属性，可以观察和控制运行应用程序的进程的运行状态。所有的**程序**对象都必须具有这些标准属性。除了这些标准属性之外，**程序**对象还提供了生产商定义的属性，这些属性可以作为程序的输入或输出。如果存在生产商定义的属性，本标准不对生产商定义的属性的工作方式作任何定义，这些属性只是专用于某个应用程序或生产商的。

程序对象及其属性归纳在表 12-21 中。

表 12-21　程序对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
程序状态 Program_State	BACnet 程序状态 BACnetProgramState	R
程序改变 Program_Change	BACnet 程序请求 BACnetProgramRequest	W
暂停原因 Reason_For_Halt	BACnet 程序错误 BACnetProgramError	O[1]
暂停描述 Description_Of_Halt	字符串 CharacterString	O[1]
程序位置 Program_Location	字符串 CharacterString	O
描述 Description	字符串 CharacterString	O
实例 Instance_Of	字符串 CharacterString	O
状态标志 Status_Flags	BACnet 状态标志 BACnetStatusFlags	R
可靠性 Reliability	BACnet 可靠性 BACnetReliability	O
脱离服务 Out_Of_Service	布尔 BOOLEAN	R

[1] 可选属性暂停原因与暂停描述必须同时存在。

12.17.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.17.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.17.3 对象类型属性

这个属性的类型是 **BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**程序（PROGRAM）**。

12.17.4 程序状态属性

这个属性的类型是 **BACnet 程序状态**类型，反映本对象所代表的运行应用程序运的进程的当前逻辑状态。本属性为只读属性，可能取值为：

空闲（IDLE）　　进程未运行

装入程序（LOADING）　　正在装入应用程序

运行（RUNNING）　　进程当前正在运行

等待（WAITING） 进程正在等待某个外部事件

已暂停（HALTED） 由于某个错误导致进程暂停

卸载（UNLOADING） 请求进程终止

12.17.5 程序改变属性

这个属性的类型是 BACnet **程序请求**类型，用于请求改变本对象所代表的进程的运行状态。**程序改变**属性提供了改变进程的运行状态的方法。进程也可以将改变它自己的状态作为一个运行结果。本属性可能取值为：

就绪（READY） 这是正常状态，表示准备接收改变请求

装入程序（LOAD） 如果还未装载应用程序，则请求装入应用程序

运行（RUN） 如果还未运行进程，则请求进程开始执行

暂停（HALT） 请求进程暂停运行

重启（RESTART） 请求进程从初始点开始重新运行

卸载（UNLOAD） 请求进程暂停运行并且卸载

正常情况下**程序改变**属性值为**就绪**，表示程序已经就绪准备接收改变它自己状态的新的请求。如果**程序改变**属性值不为**就绪**，则不可写入，而且任何写入操作都会导致返回 Result(-)。如果**程序改变**属性值具有其它列出的值，那么先前改变状态的请求不会被执行，所以新的请求也不会被接受。当最后开始执行先前的改变状态的请求时，本属性值将变为**就绪**，同时新的状态将体现在**程序状态**属性中。根据当前的**程序状态**，关于**程序改变**的某个请求值可能是无效，这时如果进行任何写操作，也将返回 Result(-)。图 12-2 描述了有效的状态转换和导出的新的**程序状态**。

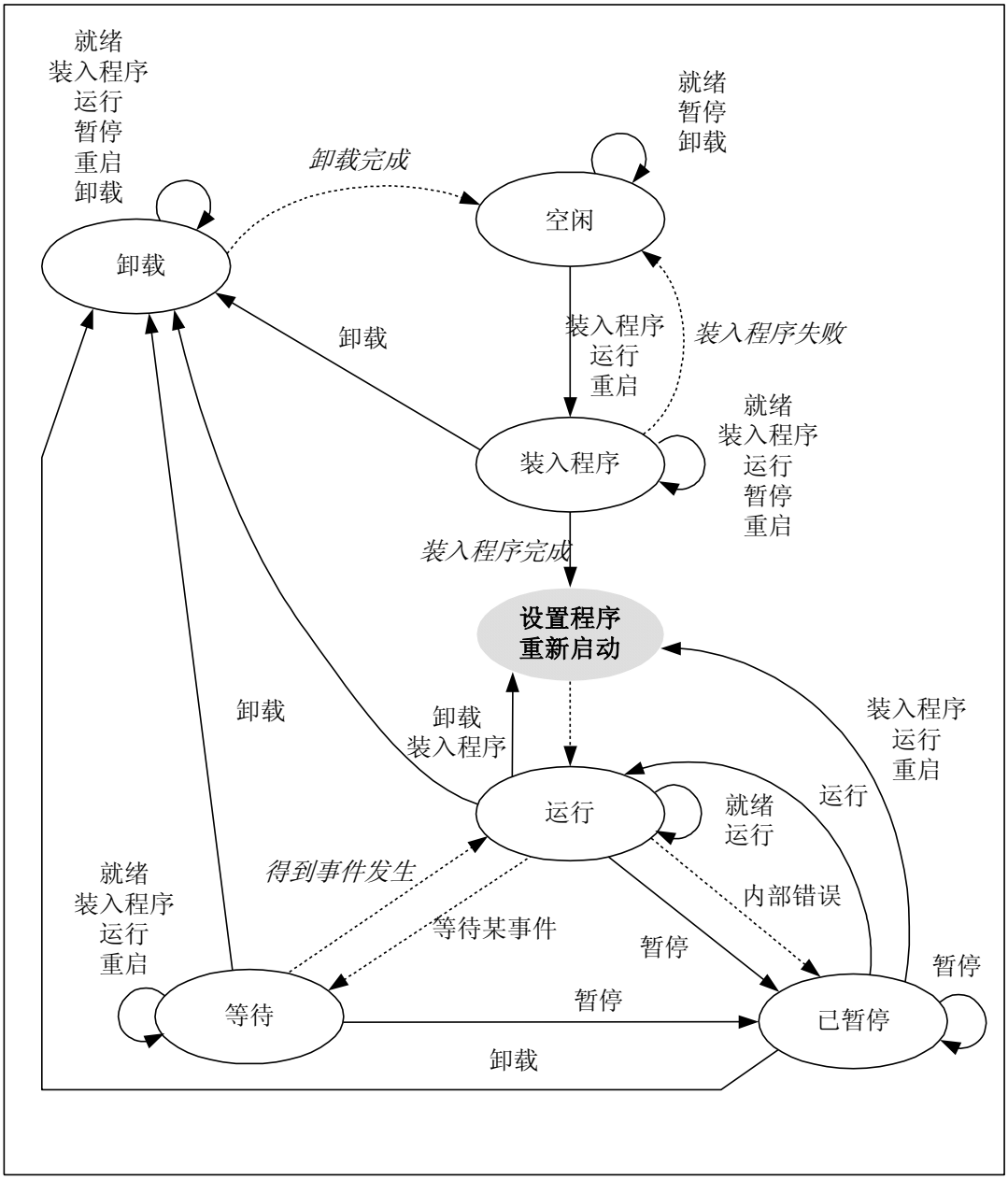


图12-2 程序对象的状态转换

值得注意的是，在装入程序的过程中，可能由于发生错误或者收到一个**暂停**请求而终止程序的装入。在任一种情况下，都可能得到“**程序状态＝已暂停**”，并且在此有一个不完整的或不可行的程序。在这种情况下，**重启**请求意味着重新**装入程序**。如果已经装入了完整的程序，但由于任何原因导致**已暂停**，那么**重启**只是简单的从初始点开始重新运行程序。

在有些情况下 **BACnet 设备**可能已经内装了应用程序，所以不要求装入应用程序，这时，装入程序意味着“准备运行”，具体规范由生产商自行确定。

12.17.6 暂停原因属性

如果运行本对象代表的应用程序的进程发生任何形式的错误，导致进程中止运行，则本属性反映出进程中止的原因。**暂停原因**属性是一种枚举类型，称之为 **BACnet 程序错误**，可

能取值如下：

正常（NORMAL） 进程没有因为任何错误原因而暂停

装入程序故障（LOAD_FAULT） 应用程序不能完整载入

内部原因（INTERNAL） 进程由于内部机制而中止

程序原因（PROGRAM） 进程由于**程序改变**请求而中止

其它原因（OTHER） 进程由于其它原因而中止

可选属性**暂停原因**属性与**暂停描述**属性必须同时存在。

12.17.7 暂停描述属性

这个属性的类型是**字符串**类型，用于描述程序中止的原因。**暂停描述**属性提供的信息与**暂停原因**属性相同，但**暂停描述**属性以可供人阅读的格式表示。字符串的内容由生产商自行确定。可选属性**暂停原因**与**暂停描述**必需同时存在。

12.17.8 程序位置属性

这个属性的类型是**字符串**类型，被应用程序用于指出它在程序代码中的位置，例如，一行数字，或者程序标记，或者分区的名称。字符串内容由生产商自行确定。

12.17.9 描述属性

这个属性的类型是**字符串**类型，用于描述被执行的应用程序或其它的生产商所期望的描述信息。

12.17.10 实例属性

这个属性的类型是**字符串**类型，描述被运行的应用程序的本地名称。字符串的内容由生产商自行确定。

12.17.11 状态标志属性

这个属性的类型是 BACnet **状态标志**类型，表示四种布尔标志，这些标志表示程序运行时刻的状况。三个标志与这个对象的其它属性值有关。通过读取与这些标志相关的属性值可以获取更详细的状态。在本协议不对标志之间的关系进行定义。这四个标志是：

{**报警中，故障，管制，脱离服务**}

其中，

报警中 这个标志的值是逻辑 FALSE(0) 。

故障 如果**可靠性**（见 12.17.12 节）属性存在并且其值不是**未发现故障**，则为 TRUE(1)，否则为 FALSE(0)。

管制　　如果某值被与 BACnet **设备**本身的有关机制所管制，则为 TRUE(1)。在此情况下，“管制”表示通过 BACnet **设备**不能改变**程序改变**属性、**程序状态**属性、或者其它程序相关的属性中的任何一个。

脱离服务　　如果**脱离服务**属性值（见 12.17.15 节）为 TRUE 则为 TRUE（1），否则为 FALSE(0)。

12.17.12 可靠性属性

这个属性的类型是 BACnet **可靠性**类型，表示程序对象中与应用相关的属性值、或者执行应用程序的进程是否“可靠”，如果不可靠，则指明原因。这个属性有下列值：

{**未发现故障**，**进程错误**，**其它不可靠**}

12.17.13 脱离服务属性

这个属性的类型是**布尔**类型，表示本对象所表示的进程是（TRUE）否（FALSE）不处在服务中。在这里，“处在服务中”表示应用程序能够正确载入和初始化，不论进程是否正在运行。如果**程序状态**属性为**空闲**，则**脱离服务**属性为 TRUE。

12.18 时间表对象类型（Schedule Object Type）

时间表对象类型定义了一个标准对象，用于描述一个周期性的时间表。这个时间表中确定了某事件在一个日期范围内可能重复发生，同时表示有些日期是事件不发生的日期。**时间表**对象还用于把预定时间和在该时间内向特定对象的特定属性写入的特定值这样两个事务绑定。**时间表**对象及其属性归纳在表 12-22 中。

时间表按天来分配，有两种形式：正常日（工作日）和例外日。

如果包含时间表的 BACnet **设备**重启，或者设备内的时间被改变，那么时间表将恢复。恢复模型规定每天的时间表是自然循环。如果 BACnet **设备**在午夜后但在当天的 BACnet **时**

间值（BACnetTimeValues）列表中定义的第一时间前重启，那么当天的列表中最后一个值用作恢复值。如果期望是其它的值，那么时间 00:00 应当作为列表的第一个时间。

表 12-22　时间表对象属性

属性标识符	属性数据类型	一致性代码
对象标识符 Object_Identifier	BACnet 对象标识符 BACnetObjectIdentifier	R
对象名称 Object_Name	字符串 CharacterString	R
对象类型 Object_Type	BACnet 对象类型 BACnetObjectType	R
当前值 Present_Value	任何类型的值 Any	R
描述 Description	字符串 CharacterString	O
有效周期 Effective_Period	BACnet 日期范围 BACnetDateRange	R
按周时间表 Weekly_Schedule	按日时间表的数组 BACnetARRAY[7] of BACnetDailySchedule	O[1]
例外时间表 Exception_Schedule	特别事件的数组 BACnetARRAY[N] of BACnetSpecialEvent	O[1]
对象属性引用列表 List_Of_Object_Property_References	BACnet 对象属性引用列表 List of BACnetObjectPropertyReference	R
写入优先级 Priority_For_Writing	无符号整型 Unsigned(1..16)	R

[1]这些属性中至少一个是必需的。

12.18.1 对象标识符属性

这个属性的类型是 **BACnet 对象标识符**类型，是一个用来标识对象的数值代码。在有这个属性的 **BACnet 设备**中是唯一的。

12.18.2 对象名称属性

这个属性的类型是**字符串**类型，表示对象名称，在有这个属性的 **BACnet 设备**中是唯一的。字符串最小长度为一个字符。**对象名称**中的字符必需为可打印字符。

12.18.3 对象类型属性

这个属性的类型是 **BACnet 对象类型**类型，表示在某个特别对象类型分类中的隶属关系。在本对象中，这个属性的值为**时间表（SCHEDULE）**。

12.18.4 当前值属性

这个属性表明了时间表的当前值。例如，最近向**对象属性引用列表**属性成员所引用对象的属性写入的数值。这些数值可以是任何数据类型。因此，模拟、数字、枚举值都可以列在时间表中。

12.18.5 描述属性

这个属性的类型是**字符串**类型，是一个由可打印字符组成的字符串，其内容不作规定。

12.18.6 有效周期属性

这个属性定义了**时间表**对象处于活动状态的日期范围。实现季节时间表的方法是，定义几个**时间表**对象，它们没有重叠的**有效周期**，这样可以控制相同的属性引用。

12.18.7 按周时间表属性

这个属性是一个 BACnet **数组**，严格包含七个成员。每个成员包含一个 BACnet **按日时间表（BACnetDailySchedule）**。BACnet **按日时间表**由 BACnet **时间值**列表组成，该列表为时间-数值对，描述了当**例外时间表（Exception_Schedule）**无效时，一周中的某一天的时间表操作序列。数组成员 1-7 分别对应星期一到星期日。本属性为可选属性，但是**按周时间表**与非空的**例外时间表**之一应被时间表对象的每个实例支持。

12.18.8 例外时间表属性

这个属性是由 BACnet **特别事件**组成的 BACnet **数组**。每个 BACnet **特别事件**描述了一个时间表操作序列，这些操作将在特定的日期取代正常的操作。

```
BACnetSpecialEvent ::=(Period, List of BACnetTimeValues, EventPriority)
Period ::=Choice of {BACnetCalendarEntry | CalendarReference}
EventPriority ::=Unsigned(1..16)
```

周期（Period）可以是 BACnet **日期表表目（BACnetCalendarEntry）**，或者引用**日期表**对象（参见 12.7 节）。如果**时间表**对象用到了**例外时间表**属性，则也可能用到 BACnet **日期表表目**，日期表中可能定义了那些被多个**时间表**对象引用的普通假日。每个 BACnet **日期表表目**为单个日期(Date)、日期范围(BACnetDataRange)或月/周-月/日-周(BACnetWeekNDay)。如果当前日期符合日期表的任何表目内容，则激活**例外时间表**属性并且可以使用 BACnet **时间值**列表中的值。

在使用 BACnet **日期表表目**确定当前日期是否落在**例外时间表**的区域内时，BACnet **日期表表目**的各种结构的单个部分可能是一个“通配符”(表示时间范围很宽)。举例来说，对于日期范围，如果**起始日期**是一个通配符（表示没有**起始日期**），则意味着可以从任何日期开始到截止日期。如果**截至日期**是一个通配符（表示没有**截止日期**），则表示从起始日期开始到任何日期为止。如果日期表的表目原来是一个具有“月”域和“周－月”域都是通配符的

BACnetWeekNDay 类型（对于月和周-月未定义但有特定的日-周），那么将在全年的那个日-周都应用**例外时间表**。

每个 **BACnet 特别事件**都包含一个**事件优先级（EventPriority）**。**事件优先级**决定了在同样的**例外时间表**中，这个 **BACnet 特别事件**相对于其它 **BACnet 特别事件**的重要性。由于在同一个**例外时间表**中的**特别事件（SpecialEvent）**有重叠的时段，那么采用一种机制确定在所给日期应用哪个**特别事件**是必要的。如果在所给日期中有多于一个的**特别事件**应用，那么有较高的事件优先级的将有效，其它的将在当天被忽略。如果多个**特别事件**有同样的优先级，那么最早发生的将有效。最高的事件优先级为 1，最低的事件优先级为 6。事件优先级与**时间表**对象的**写入优先级**属性无关。

如果 **BACnet 设备**支持向**例外时间表**属性写入，那么将支持 **BACnet 特别事件**中所有可能的选择。

12.18.9 对象属性引用列表属性

这个属性定义了在特定日期的特定时间被写入特定值的属性的**对象标识符**和**属性标识符**。

12.18.10 写入优先级属性

这个属性定义了引用属性被命令时的优先级，对应于**写属性**服务的“优先级”参数。本属性为**无符号整型**，取值范围为 1-16。1 表示最高的优先级，16 表示最低的优先级。参见 19 节。

14. 文件访问服务

本节定义一组访问和操作在 BACnet 设备中的文件的服务。在本节的定义中，文件表示一个任意长度和意义的字节集合的网络可见形式。这只是一个抽象的概念，并不表示在服务器设备中用到了磁盘，磁带或其它的大容量存储设备。这些服务可以访问 BACnet 协议标准定义的特定文件，也可以访问生产商定义的文件。

每一个能被**文件访问服务**访问的文件，在 BACnet 设备中都有一个对应的**文件**对象。这个**文件**对象标识特定的文件的名称。另外，**文件**对象还提供诸如文件总长度、文件创建日期、和文件类型这样的访问“头信息”。**文件访问服务**给出两种文件模型：连续字节流模型和编号记录的连续序列模型。

文件访问服务提供基本读和写操作。在本协议中，“基本的（atomic）”表示在读和写操作的执行期间，不允许有其它的**基本读文件**操作或者**基本写文件**操作作用在同一个文件上。具有 BACnet 设备内部操作性质的服务如何同步的问题由生产商自行解决，本标准不作任何

规定。

14.1 基本读文件（AtomicReadFile）服务

一个客户端的 BACnet 用户使用**基本读文件**服务对某个文件进行一个“打开－读出－关闭”的操作。访问的文件可以是字节流，也可以是编号记录。

14.1.1 结构

表 14-1 表示**基本读文件**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 14-1　基本读文件服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
文件标识符(File Identifier)	M	M(=)		
流访问(Stream Access)	S	S(=)		
文件开始位置(File Start Position)	M	M(=)		

请求的字节数目(Requested Octet Count)	M	M(=)		
记录访问(Record Access)	S	S(=)		
文件开始记录(File Start Record)	M	M(=)		
请求的记录数目(Requested Record Count)	M	M(=)		
Result(+)			S	S(=)
文件结束(End Of File)			M	M(=)
流访问(Stream Access)			S	S(=)
文件开始位置(File Start Position)			M	M(=)
文件数据(File Data)			C	C(=)
记录访问(Record Access)			S	S(=)
文件开始记录(File Start Record)			M	M(=)
返回的记录数目(Returned Record Count)			M	M(=)
文件记录数据(File Record Count)			C	C(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

14.1.2 参数

这个参数为**基本读文件**有证实服务请求传送参数值。

14.1.2.1 文件标识符

这个参数是标识被读文件的**文件**对象的**对象标识符**。

14.1.2.2 流访问

这个参数指明要求面向流的文件访问。流访问包含‘**文件开始位置**’参数和‘**请求的字节数目**’参数。

14.1.2.2.1 文件开始位置

这个参数是整型类型，表示从读操作开始执行的文件开始位置之后的字节的标号。‘**文件开始位置**’0 表示文件的第一个字节。

14.1.2.2.2 请求的字节数目

这个参数是无符号整型类型，表示从文件开始被读的字节的数目。

14.1.2.3 记录访问

这个参数指明要求面向记录的文件访问。记录访问包含‘**文件开始记录**’参数和‘**请求的记录数目**’参数。

14.1.2.3.1 文件开始记录

这个参数是整型类型，表示从读操作开始执行的文件开始位置之后的记录的标号。‘**文件开始记录**’0 表示文件的第一个记录。

14.1.2.3.2 请求的记录数目

这个参数是无符号整型类型，表示从文件开始被读的记录的数目。

14.1.3 Result(+)

参数‘**Result(+)**’指明服务请求已经成功，一个成功的请求包含下列参数。

14.1.3.1 文件结束

参数‘**文件结束**’是布尔类型，如果这个响应包含文件的最后一个字节，则本参数为 TRUE，否则，本参数都为 FALSE。这个参数用来检查文件是否结束，因为返回的字节的标号可能小于‘**请求的字节数目**’，或者‘**返回的记录数目**’可能小于‘**请求的记录数目**’，存在这种可能性的原因是文件本身所具有的数据数目就小于请求的数目。这个参数还可以向这个服务的用户提供一种独立于文件数据的检查文件末端的方法。

14.1.3.2 流访问

这个参数指明被请求的是面向流的文件访问。流访问包含‘**文件开始位置**’参数和‘**文件数据**’参数。

14.1.3.2.1 文件开始位置

这个参数是整型类型，表示从被读数据开始的文件开始位置之后的字节的标号。‘**文件开始位置**’0 表示文件的第一个字节。

14.1.3.2.2 文件数据

这个参数是一个请求的文件数据**字节串（OCTET STRING）**。

14.1.3.3 记录访问

这个参数指明被请求的是面向记录的文件访问。记录访问包含‘**文件开始记录**’参数、‘**返回的记录数目**’参数和‘**文件记录数据**’参数。

14.1.3.3.1 文件开始记录

这个参数是整型类型，表示从被读数据开始的文件开始位置之后的记录的标号。‘**文件开始记录**’0 表示文件的第一个记录。

14.1.3.3.2 返回的记录数目

这个参数是无符号整型类型，表示从文件开始实际被读的记录的数目，这个数目可能小于‘**请求的记录数目**’。

14.1.3.3.3 文件记录数据

这个参数是一个请求的文件数据**字节串**的列表。

14.1.4 Result(-)

参数‘**Result(-)**’指明服务请求失败，失败的原因在‘**错误类型**’参数中说明。

14.1.4.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

14.1.5 服务过程

响应的 BACnet 用户首先检验‘**文件标识符**’的有效性，如果**文件**对象未知、如果当前存在另外一个正在运行的**基本读文件**或者**基本写文件**服务、如果**文件**对象由于其它的原因当前不可访问，则返回一个‘**Result(-)**’，并给出一个适当地错误类和代码。如果‘**文件开始位置**’参数、或者‘**文件开始记录**’参数或者小于 0、或者大于文件的实际尺寸，则返回一个‘**Result(-)**’，并给出一个适当地错误类和代码。如果不是上述情况，响应的 BACnet 用户就读出由‘**请求的字节数目**’参数规定的字节数，或者读出由‘**请求的记录数目**’参数规定的记录数。如果文件中保存的字节数或者记录数小于请求的数目，则返回的‘**文件数据**’参数的长度、或者‘**返回的记录数目**’参数指明实际读出的数目。如果返回的响应包含文件的最后一个字节或者记录，则‘**文件结束**’参数应该为 TURE，否则，这个参数为 FALSE。

14.2 基本写文件（AtomicWriteFile）服务

一个客户端的 BACnet 用户使用**基本写文件**服务对某个**字节流**进行一个“打开－写入－关闭”的操作，将它写入到文件的某个位置，或者对某个**字节流**列表进行一个“打开－写入－关闭”的操作，将它写入到文件中记录的某个组里。访问的文件可以是字节流，也可以是编号记录。

14.2.1 结构

表 14-2 表示**基本写文件**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 14-2　基本写文件服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
文件标识符(File Identifier)	M	M(=)		
流访问(Stream Access)	S	S(=)		
文件开始位置(File Start Position)	M	M(=)		
文件数据(File Data)	M	M(=)		

记录访问(Record Access)	S	S(=)		
文件开始记录(File Start Record)	M	M(=)		
记录数目(Record Count)	M	M(=)		
文件记录数据(File Record Data)	M	M(=)		
Result(+)			S	S(=)
流访问(Stream Access)			S	S(=)
文件开始位置(File Start Position)			M	M(=)
记录访问(Record Access)			S	S(=)
文件开始记录(File Start Record)			M	M(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

14.2.2 参数

这个参数为**基本写文件**有证实服务请求传送参数值。

14.2.2.1 文件标识符

这个参数是标识被写文件的**文件**对象的**对象标识符**。

14.2.2.2 流访问

这个参数指明要求面向流的文件访问。流访问包含‘**文件开始位置**’参数和‘**文件数据**’参数。

14.2.2.2.1 文件开始位置

这个参数是整型类型，表示从写操作开始执行的文件开始位置之后的字节的标号。‘**文件开始位置**’0 表示文件的第一个字节。‘**文件开始位置**’-1 表示当前文件的末端，即，一个文件操作的附加（append）。

14.2.2.2.2 文件数据

这个参数是一个请求被写入到文件中的**字节串**。

14.2.2.3 记录访问

这个参数指明要求面向记录的文件访问。记录访问包含‘**文件开始记录**’参数、‘**记录数目**’参数和‘**文件记录数据**’参数。

14.2.2.3.1 文件开始记录

这个参数是整型类型，表示从写操作开始执行的文件开始位置之后的记录的标号。‘**文件开始记录**’0 表示文件的第一个记录。‘**文件开始记录**’-1 表示当前文件的末端，即，一个文件操作的附加。

14.2.2.3.2 记录数目

这个参数是无符号整型类型，表示从文件开始处开始，将被写入到文件中的记录的数目。

14.2.2.3.3 文件记录数据

这个参数是一个请求写入到文件中的**字节串**的列表。

14.2.3 Result(+)

参数‘**Result(+)**’指明服务请求已经成功，一个成功的请求包含下列参数。

14.2.3.1 流访问

这个参数指明被请求的是面向流的文件访问。流访问包含‘**文件开始位置**’参数。‘**文件开始位置**’参数是整型类型，表示从实际被写数据写入的文件开始位置之后的字节的标号。‘**文件开始位置**’0 表示文件的第一个字节。

14.2.3.2 记录访问

这个参数指明被请求的是面向记录的文件访问。记录访问包含‘**文件开始记录**’参数。‘**文件开始记录**’参数是整型类型，表示从实际被写数据写入的文件开始位置之后的记录的标号。‘**文件开始记录**’0 表示文件的第一个字节。

14.2.4 Result(-)

参数‘**Result(-)**’指明服务请求失败，失败的原因在‘**错误类型**’参数中说明。

14.2.4.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

14.2.5 服务过程

响应的 BACnet 用户首先检验‘**文件标识符**’的有效性，如果**文件**对象未知、如果当前存在另外一个正在运行的**基本读文件**或者**基本写文件**服务、如果**文件**对象由于其它的原因当前不可访问，则返回一个‘Result(-)’，并给出一个适当地错误类和代码。如果‘**文件开始位置**’参数、或者‘**文件开始记录**’参数大于文件的实际尺寸，则文件将被扩展到指定的尺

寸，但是插入的字节或者记录的内容和多少是个本地事务。如果这些参数中有一个具有特别值-1，则将写操作处理为从当前文件的末端进行附加写入。然后，响应的 BACnet 用户向文件中写入由‘**字节数目**’参数规定的字节数，或者写入由‘**记录数目**’参数规定的记录数。如果由于任何原因导致写操作失败，则返回一个‘**Result(-)**’，并给出一个适当地错误类和代码。如果全部成功写入，则返回一个‘**Result(+)**’。‘**文件开始位置**’参数、或者‘**文件开始记录**’参数指明被写数据的实际位置或者记录。

15. 对象访问服务

本节定义九个应用服务，这些服务共同提供一组访问和操作 BACnet 对象的方法。一个 BACnet 对象，不论它在所驻留的设备中具有什么功能，都可以使用本节定义的服务，访问它的属性。这些服务可以访问本标准定义的标准对象，也可以访问生产商定义的专有对象。

15.1 添加列表元素（AddListElement）服务

一个客户端的 BACnet 用户使用**添加列表元素**服务向一个具有列表的对象的属性添加一个或者多个列表元素。

15.1.1 结构

表 15-1 表示**添加列表元素**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-1　添加列表元素服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
对象标识符(Object Identifier)	M	M(=)		
属性标识符(Property Identifier)	M	M(=)		
属性数组索引(Property Array Index)	C	C(=)		
元素列表(List of Elements)	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)
第一次失败元素标号(First Failed Element Number)			M	M(=)

15.1.1.1 参数

这个参数为**添加列表元素**有证实服务请求传送参数值。

15.1.1.1.1 对象标识符

这个参数是 BACnet **对象标识符**类型,提供标识被本服务修改其列表属性的对象的方法。

15.1.1.1.2 属性标识符

这个参数是 BACnet **属性标识符**类型，提供唯一标识被本服务修改的属性的方法。

15.1.1.1.3 属性数组索引

如果被按上述方法标识的属性是一个数据类型的数组,则可以使用这个条件的无符号整型参数，指明被本服务修改的引用属性的元素的数组索引。否则，省略这个参数。

15.1.1.1.4 元素列表

这个参数表示一个或者多个将要添加到由‘**属性标识符**’参数规定的属性中的元素。这个参数的元素的数据类型由被‘**对象标识符**’参数规定的对象的类型定义所确定。

15.1.1.2 Result(+)

参数‘Result(+)’指明服务请求已经成功，所有规定的元素都已经被添加到列表中。

15.1.1.3 Result(-)

参数‘Result(-)’指明服务请求失败，没有一个规定的元素被添加到列表中。失败的原因在‘**错误类型**’参数中说明。

15.1.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

15.1.1.3.2 第一次失败元素标号

这个参数是无符号整型类型，传送在请求中所接收的‘**元素列表**’参数中错误元素的标号，标号从 1 开始排列。如果认为造成请求无效的原因‘**元素列表**’参数的不同，则设置‘**第一次失败元素标号**’为 0。

15.1.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，开始对‘**对象标识符**’参数标识的对象进行修改操作。如果标识的对象存在，并且具有‘**属性标识符**’参数规定的属性，就向规定

的属性中添加在‘**元素列表**’参数中定义的所有元素。如果操作成功，发送一个‘**Result(+)**’原语。如果有一个或者多个元素在列表中已经存在，则忽略这个（些）元素，即不向列表中添加这个（些）元素。忽略一个已经存在的元素，并不导致服务的失败。

如果标识的对象不存在，或者标识的属性不存在，或者标识的属性不是一个列表，则服务都失败，发送一个‘**Result(-)**’原语。如果有一个或者多个元素不能添加到列表中，而且它（们）又不是列表本身的成员，则发送一个‘**Result(-)**’原语，并且不向列表中添加任何元素。

这个服务的效果是，要么向列表中添加所有规定的且在列表中不存在的元素，要么不向列表中添加任何元素。

15.2 删除列表元素（RemoveListElement）服务

一个客户端的 BACnet 用户使用**删除列表元素**服务从一个具有列表的对象的属性中删除一个或者多个列表元素。

15.2.1 结构

表 15-2 表示**删除列表元素**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-2　删除列表元素服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
对象标识符(Object Identifier)	M	M(=)		
属性标识符(Property Identifier)	M	M(=)		
属性数组索引(Property Array Index)	C	C(=)		
元素列表(List of Elements)	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)
第一次失败元素标号(First Failed Element Number)			M	M(=)

15.2.1.1 参数

这个参数为**删除列表元素**有证实服务请求传送参数值。

15.2.1.1.1 对象标识符

这个参数是 BACnet **对象标识符**类型，提供标识被本服务修改其列表属性的对象的方法。

15.2.1.1.2 属性标识符

这个参数是 BACnet **属性标识符**类型，提供唯一标识被本服务修改的属性的方法。

15.2.1.1.3 属性数组索引

如果被按上述方法标识的属性是一个数据类型的数组，则可以使用这个条件的无符号整型参数，指明被本服务修改的引用属性的元素的数组索引。否则，省略这个参数。

15.2.1.1.4 元素列表

这个参数表示一个或者多个将要被从由‘**属性标识符**’参数规定的属性中删除的元素。这个参数的元素的数据类型由被‘**对象标识符**’参数规定的对象的类型定义所确定。

15.2.1.2 Result(+)

参数‘Result(+)’指明服务请求已经成功，所有规定的元素都已经从列表中删除。

15.2.1.3 Result(-)

参数‘Result(-)’指明服务请求失败，失败的原因在‘**错误类型**’参数中说明。没有一个的元素从标识的对象中被删除。

15.2.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

15.2.1.3.2 第一次失败元素标号

这个参数是无符号整型类型，传送在请求中所接收的‘**元素列表**’参数中错误元素的标号，标号从 1 开始排列。如果认为造成请求无效的原因‘**元素列表**’参数的不同，则设置‘**第一次失败元素标号**’为 0。

15.2.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，开始对‘**对象标识符**’参数标识的对象进行修改操作。如果标识的对象存在，并且具有‘**属性标识符**’参数规定的属性，就从标识的对象的属性中删除在‘**元素列表**’参数中定义的所有元素。如果标识的对象不存在，或者标识的属性不存在，或者标识的属性不是一个列表，则服务都失败，发送一个‘Result(-)’原语。如果列表中不存在规定的一个或者多个元素，或者由于不具有授权而不能删除元素，则发送一个‘Result(-)’原语，并且不从列表中删除任何元素。

15.3 创建对象（CreateObject）服务

一个客户端的 BACnet 用户使用**创建对象**服务创建一个对象的新实例。这个服务既可以创建标准对象的实例，也可以创建生产商定义的对象的实例。在 PICS 中规定了这个服务所支持的对象类型。对于使用本服务创建的对象的属性，使用两种方式进行初始化，分别是：**创建对象**服务将属性的初始化值作为请求的一部分提供给创建的对象，以及，使用 BACnet **写属性**服务向新创建的对象写入属性值。非标准对象的初始化问题由生产商自行解决。本服务所创建的对象中，那些在初始化属性值中没有提供的性能，与设备有关，由生产商自行解决。

15.3.1 结构

表 15-3 表示**创建对象**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-3　创建对象服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
对象说明符(Object Specifier)	M	M(=)		
初始值元素列表(List of Initial Values)	U	U(=)		
Result(+)			S	S(=)
对象标识符(Object Identifier)			M	M(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)
第一次失败元素标号(First Failed Element Number)			M	M(=)

15.3.1.1 参数

这个参数为**创建对象**有证实服务请求传送参数值。

15.3.1.1.1 对象说明符

这个参数传送关于要创建的对象的类型的信息。数据类型在对象类型和对象标识符之间选择。如果选择使用对象类型，则规定的对象类型成为新创建对象的**对象类型**属性值，响应的 BACnet 用户选择一个对象标识符。如果选择使用对象标识符，则创建一个具有这个选定的对象标识符的对象。

15.3.1.1.2 初始值列表

这个参数传递一个 BACnet **属性值**类型的列表，此表用来提供初始化新创建对象的特定属性值。

15.3.1.2 Result(+)

参数‘**Result(+)**’指明服务请求已经成功，包括使用所有在‘**初始值列表**’参数中定义的属性值对新创建的对象的相关属性进行了初始化。参数‘**Result(+)**’将一个‘**对象标识符**’作为自己的参数返回，这个参数作为新创建的对象的**对象标识符**属性的值。这个标识符在服务器设备中是唯一的。

15.3.1.3 Result(-)

参数‘**Result(-)**’指明服务请求失败，没有一个规定的元素被添加到列表中。失败的原因在‘**错误类型**’参数中说明。

15.3.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

15.3.1.3.2 第一次失败元素标号

这个参数是无符号整型类型，传送在请求中所接收的‘**初始值列表**’参数中错误元素的标号，标号从 1 开始排列。如果认为造成请求无效的原因‘**初始值列表**’参数的不同，则设置‘**第一次失败元素标号**’为 0。

15.3.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，开始进行创建一个具有‘**对象说明符**’参数所规定的类型的新对象的操作。

如果‘**对象说明符**’参数中包含一个对象类型，则新创建对象的**对象标识符**属性被初始化为一个在响应的 BACnet 用户设备中唯一的值。用来产生对象标识符的方法由生产商自行确定。**对象类型**属性被初始化为‘**对象说明符**’参数的值。如果不能创建规定类型的新对象，则返回一个‘**Result(-)**’原语，并且将‘**第一次失败元素标号**’参数设置为 0。

如果‘**对象说明符**’参数中包含一个对象标识符，则响应的 BACnet 用户确定这个标识符的对象是否已经存在。如果已经存在一个这样的对象，就不创建新的对象，返回一个‘**Result(-)**’原语，并且将‘**第一次失败元素标号**’参数设置为 0。如果不存在这样的对象，但是不能创建这个对象，则返回一个‘**Result(-)**’原语，并且将‘**第一次失败元素标号**’参数设置为 0。如果这样的对象不存在，并且可以创建，则创建一个新对象。新对象的**对象标识符**属性具有‘**对象说明符**’参数所规定的值，**对象类型**属性具有一个与**对象标识符**的对象类型域一致的值。参见 20.2.14 节。

如果原语中包含有可选的‘**初始值列表**’参数，则列表中的所有属性都要被初始化为指定值。其它属性的初始化问题由生产商自行解决。如果不能成功完成初始化过程，则返回一个‘Result(-)’原语，将‘**第一次失败元素标号**’参数指定在‘**初始值列表**’参数中的第一个不能被初始化属性，并且不创建对象。如果创建新对象的操作成功，则返回一个‘Result(+)’原语，其中传送了新创建对象的**对象标识符**属性的值。

15.4 删除对象（DeleteObject）服务

一个客户端的 BACnet 用户使用**删除对象**服务删除一个已有的对象。虽然从原理上说可以使用这个服务删除任何类型的对象，但是从控制系统安全方面考虑，希望大部分对象都不能使用这个服务删除。只有某一些对象可以被动态地创建和删除，这些对象包括**组**对象和**事件登记**对象。本服务主要用于删除这些类型对象，也可用于删除上场上定义的可删除的专有对象。

15.4.1 结构

表 15-4 表示**删除对象**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-4　删除对象服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
对象标识符(Object Identifier)	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

15.4.1.1 参数

这个参数为**删除对象**有证实服务请求传送参数值。

15.4.1.1.1 对象标识符

这个参数是 BACnet **对象标识符**类型，指明本服务要删除的对象。

15.4.1.2 Result(+)

参数‘Result(+)’指明服务请求已经成功，指定的对象已经被删除。

15.4.1.3 Result(-)

参数‘**Result(-)**’指明服务请求失败，没有删除指定的对象。失败的原因在‘**错误类型**’参数中说明。

15.4.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

15.4.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，开始进行删除一个具有请求/指示原语中‘**对象标识符**’参数所规定的对象的操作。如果指定的对象存在，并且可以被删除，则删除这个对象，发送一个‘**Result(+)**’原语。如果指定的对象不存在，或者指定的对象存在但是不能被删除，则发送一个‘**Result(-)**’原语。

15.5 读属性（ReadProperty）服务

一个客户端的 BACnet 用户使用**读属性**服务请求一个 BACnet 对象的一个属性值。本服务允许读访问任何一个对象的任何属性，不论对象是否 BACnet 的标准对象。

15.5.1 结构

表 15-5 表示**读属性**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-5 读属性服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
对象标识符(Object Identifier)	M	M(=)		
属性标识符(Property Identifier)	M	M(=)		
属性数组索引(Property Array Index)	U	U(=)		
Result(+)			S	S(=)
对象标识符(Object Identifier)			M	M(=)
属性标识符(Property Identifier)			M	M(=)
属性数组索引(Property Array Index)			U	U(=)
属性值(Property Value)			M	M(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

15.5.1.1 参数

这个参数为**读属性**有证实服务请求传送参数值。

15.5.1.1.1 对象标识符

这个参数是 BACnet **对象标识符**类型，提供标识一个对象的方法，这个对象的属性将要被本服务读出并且返回给客户端 BACnet 用户。

15.5.1.1.2 属性标识符

这个参数是 BACnet **属性标识符**类型，提供唯一标识被本服务读出并且返回的属性的方法。因为本服务用于读出单个对象的单个属性的值，所以，这个参数值不会是**全部，必需的，可选的**这样的属性标识符。

15.5.1.1.3 属性数组索引

如果被按上述方法标识的属性是一个数据类型的数组，则可以使用这个可选的无符号整型参数，指明被本服务引用的属性的元素的数组索引。如果省略这个参数，表示整个数组都被引用。如果被按上述方法标识的属性不是一个数据类型的数组，则省略这个参数。

15.5.1.2 Result(+)

参数‘Result(+)’指明服务请求已经成功。一个成功的结果包含有下列参数：

15.5.1.2.1 对象标识符

这个参数是 BACnet **对象标识符**类型，标识一个对象，这个对象的属性已经被读出并且正在被返回给客户端 BACnet 用户。

15.5.1.2.2 属性标识符

这个参数是 BACnet **属性标识符**类型，标识那个已经被读出并且正在被本服务返回的属性。因为本服务用于读出单个对象的单个属性的值，所以，这个参数值不会是**全部，必需的，可选的**这样的属性标识符。

15.5.1.2.3 属性数组索引

如果被按上述方法标识的属性是一个数据类型的数组，并且在请求中规定了一个‘属性数组索引’参数，则可以使用这个无符号整型参数，指明被本服务引用的属性的元素的数组索引。否则，省略这个参数。

15.5.1.2.4 属性值

如果对特定对象的特定属性访问操作成功，则返回这个参数。这个参数的数据类型与访问的属性的数据类型一样，并且包含请求的属性的值。

15.5.1.3 Result(-)

参数‘**Result(-)**’指明服务请求完全失败。失败的原因在‘**错误类型**’参数中说明。

15.5.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

15.5.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，开始执行对特定对象的特定属性的访问操作。如果访问成功，生成并且返回一个‘**Result(+)**’原语，其中包含有访问的值。如果访问失败，发送一个‘**Result(-)**’原语。

15.6 条件读属性（ReadPropertyConditional）服务

一个客户端的 BACnet 用户使用**条件读属性**服务请求那些满足一个选择准则列表的所有 BACnet 对象的对象标识符和 0 个或者多个特定属性的值，这个选择准则列表也称之为“条件”。条件以布尔表达式的形式表示（Property.Relational_Operator.Constant）。另外，条件列表可以是逻辑‘或’或者逻辑‘与’，用来确定是否将某个对象包含进入其属性将被读的对象集合内。终极概率是所有的（ALL）对象都被选择，它们的某个属性值被读。

本服务与**读属性**服务之间的差别是，在**读属性**服务中，客户端的 BACnet 用户明确标识了要读属性值的对象，而在本服务中，客户端的 BACnet 用户通过提供一个选择准则来标识要读属性值的对象。

15.6.1 结构

表 15-6 表示**条件读属性**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-6　条件读属性服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
对象选择准则(Object Selection Criteria)	M	M(=)		
属性引用列表(List of Property References)	U	U(=)		

Result(+) 读访问结果列表(List of Read Access Results)			S M	S(=) M(=)
Result(-) 错误类型(Error Type)			S M	S(=) M(=)

15.6.1.1 参数

这个参数为**条件读属性**有证实服务请求传送参数值。

15.6.1.1.1 对象选择准则

这个参数包括一个‘**选择逻辑**（Selection Logic)’参数和一个或者多个根据条件而定的‘**选择准则**（Selection Criteria)’列表。每个显式准则包含有四个参数，分别是：(1) 一个‘**属性标识符**’,(2) 一个可选的‘**属性数组索引**’,(3) 一个‘**关系说明符**(Relation Specifier)’，(4)一个适合指定属性的‘**比较值**（Comparison Value)’。参见 15.6.3.1 节。

15.6.1.1.2 属性引用列表

这个参数是是一个可选的参数，如果存在，则是一个或者多个 BACnet **属性引用**类型的列表，每一个列表直接对应一个特定的属性，这个属性是任何一个按照‘**对象选择准则**’选择的对象的属性。说明属性域为**全部**，指明返回选择的对象的所有属性，包括专有属性。说明属性域为**必需的**，指明返回特征为“R”或者“W”的属性。说明属性域为**可选的**，指明返回特征为“O”的属性。参见 12 节中关于某个对象类型的规范。如果这个参数不存在，则在‘Result(+)’原语的‘**读访问结果列表**’中没有返回的属性值，但是在‘**对象标识符**’参数中有对象标识符返回。

15.6.1.2 Result(+)

参数‘Result(+)’指明服务请求已经成功。一个成功的结果包含有下列参数：

15.6.1.2.1 读访问结果列表

对于所有的按照‘**对象选择准则**’选择的对象的每一个指定的属性，这个参数指明读属性操作是否成功。每一个‘**读访问结果**’都提供 ‘**对象标识符**’,‘**属性标识符**’和条件的‘**属性数组索引**’，指明每一个读访问操作。如果读访问操作成功，则下一个参数就是指定的属性的值。如果读属性操作失败，则下一个参数是一个错误代码，表明访问失败的原因。如果按照‘**对象选择准则**’要求没有选中一个对象，则‘**读访问结果列表**’的长度为 0。如果在请求中不存在‘**属性引用列表**’参数，则‘**读访问结果列表**’包含具有空属性值的‘**读访问结果**’列表，只传递与选择准则匹配的对象的标识符。

15.6.1.3 Result(-)

参数‘**Result(-)**’指明服务请求完全失败。失败的原因在‘**错误类型**’参数中说明。

15.6.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

15.6.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，就搜寻自己的对象数据库，选择满足特定选择准则的对象。对于每一个选中的对象，根据特定的‘**属性引用列表**’参数，构造一个‘**读访问结果**’。虽然没有要求请求要被自动执行，但是响应的 BACnet 用户应该保证，除了服从优先级准则之外，要在尽可能最短的时间内读出所有的属性值。请求被连续执行下去，直到所有的选择的对象的指定属性都被访问完成。如果没有搜寻到满足准则的对象，则返回一个‘**Result(+)**’原语，其中‘**读访问结果列表**’的长度为 0。如果搜寻到一个或者多个满足准则的对象，则返回一个‘**Result(+)**’原语，其中在‘**读访问结果列表**’中传送所有的访问结果。

15.6.3 条件读属性服务引用的参数

以下是在**条件读属性**服务原语中出现的参数：

15.6.3.1 对象选择准则参数

表 15-7 表示‘**对象选择准则**’参数，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-7 ‘对象选择准则’参数结构

参数名称	Req Ind	Rsp Cnf	数据类型
选择逻辑(Selection Logic)	M	M(=)	枚举类型
选择准则列表(List of Selection Criteria)	C	C(=)	
属性标识符(Property Identifier)	M	M(=)	BACnet 属性标识符
属性数组索引(Property Array Index)	C	C(=)	无符号整型
关系说明符(Relation Specifier)	M	M(=)	枚举类型
比较值(Comparison Value)	M	M(=)	任何值

15.6.3.1.1 选择逻辑

这个参数指明要返回其属性值的对象的范围，可以取三个值：AND，OR，ALL。对于在这个参数取 AND 或者 OR 的情况，参数表示将多个选择准则组合来确定选择要返回其属性值的

对象的方法。如果这个参数值为 AND，则对于选择的对象，必须所有的选择准则都得出 TRUE 才能选中。如果这个参数为 OR，则对于选择的对象，必须至少有一个选择准则得出 TRUE 才能选中。如果这个参数为 ALL，则包含在响应的 BACnet 用户的对象数据库中的所有对象都被选中，并且在请求原语中省略‘**选择准则列表**’。

15.6.3.1.2 选择准则列表

如果这个参数存在,它包含一个或者多个属性关系列表,用于选择上述规定的特定对象。每个关系包含一个‘**属性标识符**’，一个条件的‘**属性数组索引**’，一个‘**关系说明符**’，和‘**比较值**’。当响应的 BACnet 用户进行评价时,每一组这样的参数一起组成一个布尔表达式，取值为 TRUE 或者 FALSE。如果规定了多于一个的属性关系式，则所有评价的结果根据‘选择逻辑’参数组合起来，产生总的 TRUE 值或者 FALSE 值。如果总的评价值为 TRUE，则响应的 BACnet 用户返回关于这个对象的由‘**属性引用列表**’参数指定的属性值。

15.6.3.1.2.1 属性标识符

这个参数是 **BACnet 属性标识符**类型，标识在对象选择过程中要使用的属性。这个参数值可能不是特定属性标识符 ALL，REQUIRED，或者 OPTIONAL 之一。

15.6.3.1.2.2 属性数组索引

如果按照上述方法标识的属性是数据类型的数组，这个条件的无符号整型类型的存在，指明本服务引用属性的元素的数组索引。否则，省略这个参数。不允许 0 值的‘**属性数组索引**’的参数存在。

15.6.3.1.2.3 关系说明符

这个参数代表下列的 6 个布尔操作符之一：

＝ （相等） ＞ （大于）

≠ （不等） ≤ （小于等于）

＜ （小于） ≥ （大于等于）

可以不对每个对象的每个属性都使用布尔操作符,有些属性值可能只允许进行等于(＝)和不等于(≠)的比较。表 15-8 示出每一种数据类型可以运用的操作符。

表 15-8 对于 BACnet 数据类型合法的布尔操作符

数据类型	允许的操作符
空（NULL）	＝ ≠
布尔类型（BOOLEAN）	＝ ≠
无符号整型（Unsigned）	＝ ≠ ＜ ＞ ≤ ≥
整型（INTEGER）	＝ ≠ ＜ ＞ ≤ ≥

实型（REAL）	＝ ≠ ＜ ＞ ≤ ≥
比特流（BIT STRING）	＝ ≠
BACnet 对象标识符（BACnetObjectIdentifier）	＝ ≠ ＜ ＞ ≤ ≥
字符串（CharacterString）	＝ ≠ ＜ ＞ ≤ ≥
字节串（OCTET STRING）	＝ ≠
枚举类型（ENUMERATED）	＝ ≠
日期类型（Date）	＝ ≠ ＜ ＞ ≤ ≥
时间类型（Time）	＝ ≠ ＜ ＞ ≤ ≥
其它类型（Others）	＝ ≠

15.6.3.1.2.4 比较值

这个参数是一个具有与被比较的属性相同数据类型的常数。如果使用**比较值**选择**字符串**类型的属性，则可以使用通配符“?”和“*”表示＝或者≠操作符。

“?”字符用来精确匹配字符串中某个位置上的字符。比较值可以包含多个“?”的发生。因此，“??1”可以匹配“AC1”，“CP1”和“SF1”，但是不能匹配“C1”，“P1”或者“PUMP1”。

“*”字符用来匹配从这个字符在字符串中的位置开始向右的 0 个或者多个字符。每个比较值只使用一个“*”通配符。因此，“C*1”匹配“C1”，“CP1”和“Chiller1”。

当使用比较值选择对象标识符时，认为对象标识符是一个 32 位的无符号整型，其高位字节是对象标识符的第一个字节。

当使用比较值选择字符串时，比较对大小写字母区分。当使用不等号（＜，＞，≤，≥＝比较字符串时，其比较是根据字符的数字编码，从左向右进行。例如：“ABC” ＜“ABCD” ＜“DE” ＜“DEF”。

当使用比较值选择字符串时，比较对大小写字母区分。当使用不等号（＜，＞，≤，≥＝比较字符串时，其比较是根据字符的数字编码，从左向右进行。例如：“ABC” ＜“ABCD” ＜“DE” ＜“DEF”。

当属性的类型是日期类型或者时间类型，并且进行不相等的比较时，比较按照日期顺序进行。例如：

1-Jan-1991 ＜ 2-Feb-1992，

7:00:00.00 ＜ 23:00:00.00

对于日期或者时间类型中的子部分，例如年份，可以使用通配符表示这些部分具有“不规定”值的情况。对于具有“不规定”值的日期或者时间的子部分，在比较时，不对这个部分作比较运算，并且认为这些部分是相等的。存在三种情况，一种情况是在比较值中有具有“不规定”值的子部分，第二种情况是在属性值中有具有“不规定”值的子部分，第三种情况是在比较值和属性值中都有“不规定”值部分，在任何一种情况下，都要应用“不规定”值的匹配规则。例如（使用*表示“不规定”值）：

比较值		属性值	原因		
- Sun	=	29-Jan-1995 Sun	Sun	=	Sun
1-Jan-*	<	2-Feb-1995	1-Jan	<	2-Feb
-Aug-	>	10-Mar-1995	Aug	>	Mar
*:59:00.00	>	6:23:99.0	59:00.00	>	23:99.00
6:*.*.*	=	6:25:00.00	6	=	6

具有空、布尔、比特流、字节流、枚举类型的属性，或者任何具有复杂结构的原语类型的属性，只能够进行等于和不等于的比较运算。

15.6.3.2 读访问结果

表 15-9 表示‘**读访问结果**’参数，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-9 ‘读访问结果’参数结构

参数名称	Req	Cnf	数据类型
对象标识符(Object Identifier)	M	M(=)	BACnet 对象标识符
结果列表(List of Results)	U	U(=)	
属性标识符(Property Identifier)	M	M(=)	BACnet 属性标识符
属性数组索引(Property Array Index)	U	U(=)	无符号整型
属性值(Property Value)	S	S(=)	任何值
属性访问错误(Property Access Error)	S	S(=)	错误类型

15.6.3.2.1 对象标识符

这个参数是 BACnet **对象标识符**类型，标识其属性要被返回给服务请求的对象。

15.6.3.1.2 结果列表

读访问一个给定属性的结果，要么是这个属性值，要么是一个指明访问失败的错误代码。注意，如果在原始请求中没有给出对‘**属性引用列表**’参数的说明，则省略‘**结果列表**’参数。这个特性提供了一种方法，可以在不接收对象的任何属性值的情况下，获得满足‘**对象选择最准则**’的所有对象的标识符。

15.6.3.2.2.1 属性标识符

这个参数是 BACnet **属性标识符**类型，标识要返回其值的属性。

15.6.3.2.2.2 属性数组索引

如果按照上述方法标识的属性是数据类型的数组，并且在请求中规定了‘**属性数组索引**’，则存在这个无符号整型类型的参数，它指明本服务引用属性的元素的数组索引。否则，

省略这个参数。

15.6.3.2.2.3 属性值

如果对选择的对象的某个属性的读访问成功，则返回这个参数，它的类型与指定的属性值的类型相同，并且包含请求的属性的值。

15.6.3.2.2.4 属性访问错误

如果响应的 BACnet 用户不能访问特定对象的特定属性，则返回这个参数，其中包含一个指明访问失败原因的值。这个参数包含两个部分：(1) 一个‘错误类’，和(2) 一个‘错误代码’。参见 18 节。注意，这个参数只是关于对特定对象的特定属性访问失败的问题，而在‘**Result(-)**’原语中返回的‘**错误类型**’参数，则是关于整个**条件读属性**服务请求的失败的问题。

15.7 读多个属性（ReadPropertyMultiple）服务

一个客户端的BACnet用户使用**读多个属性**服务请求一个或者多个BACnet对象的一个或者多个特定属性的值。本服务允许对任何对象的任何属性进行访问，不论是否 BACnet 定义的对象。用户可以读单个对象的单个属性，单个对象的一个属性列表，或者，任意多个对象的任意多个属性。可以使用具有属性标识符**全部**的‘读访问规范’参数，了解一个对象的属性和它们的值得实现方式。

15.7.1 结构

表 15-10 表示**读多个属性**服务原语的结构,表中所用的术语和符号在 5.6 节中给出了解释。

表 15-10　读多个属性服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
读访问规范列表(List of Read Access Specifications)	M	M(=)		
Result(+)			S	S(=)
读访问结果列表(List of Read Access Results)			M	M(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

15.7.1.1 参数

这个参数为**读多个属性**有证实服务请求传送参数值。

15.7.1.1.1 读访问规范列表

这个参数包括一个或者多个‘**读访问规范**’的列表，每个规范包括两个参数：(1) 一个‘对象标识符’，(2) 一个‘属性引用列表’。参见 15.7.3.1 节。

15.7.1.2 Result(+)

参数‘**Result(+)**’指明服务请求已经成功。一个成功的结果包含有下列参数：

15.7.1.2.1 读访问结果列表

这个参数指明对每个指定的属性的访问是否成功。在 15.7.3.2 节中定义每个**读访问结果**的内容。

15.7.1.3 Result(-)

参数‘**Result(-)**’指明服务请求完全失败。失败的原因在‘**错误类型**’参数中说明。

15.7.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

15.7.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，开始进行访问特定对象的特点属性的操作，并且构造一个按照请求中规定的顺序的‘**读访问结果列表**’。如果‘**读访问规范列表**’参数中的‘**属性引用列表**’部分包含属性标识符**全部**、**必需的**、或者**可选的**，则按照每个被返回的属性已经被明确地引用的方法构造‘**读访问结果列表**’（参见 15.7.3.1.2 节）。虽然没有要求请求要被自动执行，但是响应的 BACnet 用户应该保证，除了服从优先级准则之外，要在尽可能最短的时间内读出所有的属性值。请求被连续执行下去，直到所有的选择的对象的指定属性都被访问完成。如果没有搜寻到满足准则的对象，或者如果指定对象的指定属性不能被访问，则发送一个‘**Result(-)**’原语。如果指定对象的指定属性中只要有一个能够被访问，则发送一个‘**Result(+)**’原语，其中，返回所有的访问值，以及对于不能访问的属性返回错误代码。

15.7.3 读多个属性服务引用的参数

以下是在**读多个属性**服务原语中出现的参数：

15.7.3.1 读访问规范参数

表 15-11 表示‘**读访问规范**’参数，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-11 ‘读访问规范’参数结构

参数名称	Req Ind	Rsp Cnf	数据类型
对象标识符(Object Identifier)	M	M(=)	BACnet 对象标识符
属性引用列表(List of Property References)	M	M(=)	BACnet 属性引用列表

15.7.3.1.1 对象标识符

这个参数是**BACnet 对象标识符**类型，提供一种方法，标识将要读出其属性值并且返回给服务请求的对象。

15.7.3.1.2 属性引用列表

这个参数是一个或者多个**BACnet 属性引用**类型的列表，每一个列表直接对应上述标识的对象的一个特定的属性。属性标识符 ALL 表示对象的所有定义的属性都可以被访问，包括专有属性。属性标识符 REQUIRED 表示只返回特征为“R”或者“W”的属性。属性表标识符 OPTIONAL 表示只返回特征为“O”的属性。参见 12 节中关于某个对象类型的规范。

15.7.3.2 读访问结果

表 15-12 表示‘**读访问结果**’参数，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-12 ‘读访问结果’参数结构

参数名称	Req	Cnf	数据类型
对象标识符(Object Identifier)	M	M(=)	BACnet 对象标识符
结果列表(List of Results)	M	M(=)	
属性标识符(Property Identifier)	M	M(=)	BACnet 属性标识符
属性数组索引(Property Array Index)	U	U(=)	无符号整型
属性值(Property Value)	S	S(=)	任何值
属性访问错误(Property Access Error)	S	S(=)	错误类型

15.7.3.2.1 对象标识符

这个参数是**BACnet 对象标识符**类型，标识其属性要被返回给服务请求的对象。

15.7.3.1.2 结果列表

读访问一个给定属性的结果，要么是这个属性值，要么是一个指明访问失败的错误代码。‘**结果列表**’中的每个元素都包含有一个‘**属性标识符**’，一个条件的‘**属性数组索引**’，一个‘**属性值**’或者‘**属性访问错误**’参数。

15.7.3.2.2.1 属性标识符

这个参数是**BACnet 属性标识符**类型，标识要返回其值的属性。

15.7.3.2.2.2 属性数组索引

如果按照上述方法标识的属性是数据类型的数组，并且在请求中规定了‘**属性数组索引**’，则存在这个无符号整型类型的参数，它指明本服务引用属性的元素的数组索引。否则，省略这个参数。

15.7.3.2.2.3 属性值

如果对选择的对象的某个属性的读访问成功，则返回这个参数，它的类型与请求的属性值的类型相同，并且包含请求的属性的值。

15.7.3.2.2.4 属性访问错误

如果响应的 BACnet 用户不能访问特定对象的特定属性，则返回这个参数，其中包含一个指明访问失败原因的值。这个参数包含两个部分：(1) 一个‘**错误类**’，和(2) 一个‘**错误代码**’。参见 18 节。注意，这个参数只是关于对特定对象的特定属性访问失败的问题，而在‘Result(-)’原语中返回的‘**错误类型**’参数，则是关于整个**条件读属性**服务请求的失败的问题。

15.8 写属性（WriteProperty）服务

一个客户端的 BACnet 用户使用**写属性**服务修改一个 BACnet 对象的一个属性值。本服务本质上允许写访问任何一个对象的任何属性，不论对象是否 BACnet 的标准对象。然而，有些系统希望禁止对某些对象的某些属性的写访问，在这种情况下，任何对禁止的属性进行写操作都将导致返回一个‘**错误类**’为 PROPERTY 和‘**错误代码**’为 WRITE_ACCESS_DENIED 的错误值。注意，这些被禁止的属性可以通过使用**虚拟终端（Virtual Terminal）**服务或者其它的方法进行访问。

15.8.1 结构

表 15-13 表示**写属性**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-13　写属性服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
对象标识符(Object Identifier)	M	M(=)		
属性标识符(Property Identifier)	M	M(=)		
属性数组索引(Property Array Index)	U	U(=)		
属性值(Property Value)	M	M(=)		

优先级(Priority)	C	C(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

15.8.1.1 参数

这个参数为**写属性**有证实服务请求传送参数值。

15.8.1.1.1 对象标识符

这个参数是 BACnet **对象标识符**类型，提供标识一个对象的方法，这个对象的属性将要被本服务修改。

15.8.1.1.2 属性标识符

这个参数是 BACnet **属性标识符**类型，提供唯一标识被本服务写入属性的方法。因为本服务用于写入单个对象的单个属性的值，所以，这个参数值不会是**全部，必需的，可选的**这样的属性标识符。

15.8.1.1.3 属性数组索引

如果被按上述方法标识的属性是一个数据类型的数组，则可以使用这个可选的无符号整型参数，指明被本服务引用的属性的元素的数组索引。如果省略这个参数，表示整个数组都被引用。如果被按上述方法标识的属性不是一个数据类型的数组，则省略这个参数。

15.8.1.1.4 属性值

如果对特定对象的特定属性的访问成功，则使用这个参数在访问时刻置换访问的属性值。这个参数的数据类型是任何对修改的属性有效的数据类型。

15.8.1.1.5 优先级

这个参数是取值范围在 1-16 之间的整数，表示分配给这个写操作的优先级。如果向一个可命令的属性进行没有说明优先级的写操作，则缺省的优先级设定为 16（最低的优先级）。如果向一个非可命令的属性进行规定优先级的写操作，则忽略优先级。参见第 19 节。

15.8.1.2 Result(+)

参数‘**Result(+)**’指明服务请求已经完全成功。

15.8.1.3 Result(-)

参数‘**Result(-)**’指明服务请求完全失败。失败的原因在‘**错误类型**’参数中说明。

15.8.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

15.8.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，使用‘**属性值**’参数提供的值执行对特定对象的特定属性的修改操作。如果修改操作成功，发送一个‘**Result(+)**’原语。如果修

改操作失败，发送一个‘**Result(-)**’原语，指明失败的原因。在第 19 节中定义条件的优先级参数的意义。

15.9 写多个属性（WritePropertyMultiple）服务

一个客户端的BACnet用户使用**写多个属性**服务修改一个或者多个BACnet对象的一个或者多个特定属性的值。本服务在本质上允许对任何对象的任何属性进行访问，不论是否BACnet 定义的对象。

本服务按照‘写访问规范列表’参数说规定的顺序对属性进行修改操作，这个操作连续执行，直到所有的指定属性都被写操作完成，或者直到遇到一个属性因为某种原因而不能按照请求的要求被修改为止。

有些系统希望禁止对某些对象的某些属性的写访问，在这种情况下，任何对禁止的属性进行写操作都将导致返回一个‘**错误类**’为 PROPERTY 和‘**错误代码**’为 WRITE_ACCESS_DENIED 的错误值。注意，这些被禁止的属性可以通过使用**虚拟终端**服务或者其它的方法进行访问。

15.9.1 结构

表 15-14 表示**写多个属性**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-14　写多个属性服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
写访问规范列表(List of Write Access Specifications)	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)
第一个失败写操作(First Failed Write Attempt)			M	M(=)

15.9.1.1 参数

这个参数为**写多个属性**有证实服务请求传送参数值。

15.9.1.1.1 写访问规范列表

每个‘**写访问规范**’传送为了执行对一个 BACnet 对象的一个或者多个属性进行修改所需要的信息。这个规范包括 5 个参数：(1) 一个‘**对象标识符**’；一个属性列表，其中每个属性包括(2) 一个‘**属性标识符**’；(3) 一个可选的‘**属性数组索引**’；(4) 一个‘**优先级值**’；和(5) 一个可选的‘**优先级**’。

15.9.1.2 Result(+)

参数‘Result(+)’指明服务请求已经完全成功，所有指定的属性都被正确地修改。

15.9.1.3 Result(-)

参数‘Result(-)’指明至少有一个指定的属性不能按照请求的要求被修改。失败的原因在‘**错误类型**’参数中说明，并且同时传送第一次遇到的不能被正确写入的属性的‘**对象标识符**’，‘**属性标识符**’，和‘**属性数组索引**’。

15.9.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

15.9.1.3.2 第一个失败写操作

这个参数是 BACnet 对象属性引用类型，传送在‘**写访问规范列表**’中的第一个写操作失败元素的‘**对象标识符**’，‘**属性标识符**’，和‘**属性数组索引**’。

15.9.2 服务过程

响应的 BACnet 用户对于包含在‘**写访问规范列表**’中的每一个‘**写访问规范**’，检验指定的对象是否在本地设备中存在、引用的属性是否是可修改的、以及每个指定的属性值是否具有正确的数据类型和处于有效的范围。

如果检验都正确，使用‘**写访问规范**’中提供的属性值置换指定的属性值，并且发送一个‘Result(+)’原语，指明服务请求已经正确执行完毕。第 19 节中给出了条件的优先级参数的意义。

如果在按照‘**写访问规范列表**’规定的顺序进行对指定的属性修改操作时，遇到了一个不能进行修改的属性，则发送一个‘Result(-)’响应原语，指明失败的原因。这个服务的结果，要么是所有的属性都被正确修改，要么是只对‘**第一个失败写操作**’参数中指明的属性之前的属性（不包括这个属性）进行了成功修改。

15.9.3 写多个属性服务引用的参数

表 15-15 表示‘**写访问规范**’参数，表中所用的术语和符号在 5.6 节中给出了解释。

表 15-15 ‘写访问规范’参数结构

参数名称	Req Ind	Rsp Cnf	数据类型
对象标识符(Object Identifier)	M	M(=)	BACnet 对象标识符
结果列表(List of Results)	M	M(=)	
属性标识符(Property Identifier)	M	M(=)	BACnet 属性标识符
属性数组索引(Property Array Index)	U	U(=)	无符号整型
属性值(Property Value)	M	M(=)	任何值
优先级(Priority)	C	C(=)	无符号整型(1 … 16)

15.9.3.1 对象标识符

这个参数是 **BACnet 对象标识符**类型，标识其属性要被修改的对象。

15.9.3.2 属性列表

这个参数表示一个或者多个列表，其中内容是上述标识符标识的对象的属性、写入到每个属性的值、和写操作所分配的优先级。列表中的每个元素规范下列参数。

15.9.3.2.1 属性标识符

这个参数是**BACnet 属性标识符**类型，提供唯一标识本服务要写入的属性的方法。这个参数值不会是特定属性标识符**全部**，**必需的**，和**可选的**中的一种。

15.9.3.2.2 属性数组索引

如果按照上述方法标识的属性是数据类型的数组，则这个可选的无符号整型类型的参数指明本服务引用属性的元素的数组索引。如果对于某个数组省略了这个参数，表示整个数组都被引用。

15.9.3.2.3 属性值

如果对选择的对象的某个属性的读访问成功，则在访问的时刻，使用这个参数置换访问的属性值。这个参数的数据类型是一个与被修改属性相一致的数据类型。

15.9.3.2.4 优先级

这个参数是取值范围在 1-16 之间的整数，表示分配给这个写操作的优先级。如果向一个可命令的属性进行没有说明优先级的写操作，则缺省的优先级设定为 16（最低的优先级）。如果向一个非可命令的属性进行规定优先级的写操作，则忽略优先级。参见第 19 节。

16. 远程设备管理服务

16.1 设备通信控制（DeviceCommunicationControl）服务

一个客户端的 BACnet 用户使用**设备通信控制**服务向一个远程设备发送指令，指示这个设备在一个规定的时间内停止起始和响应除了**设备通信控制**或者**重新初始化设备（ReinitializeDevice）**之外的所有的 APDU。本服务主要由操作者用来进行设备诊断。在客户端可以要求使用密码，时间期限可以设定为“不确定”，表示必须使用**设备通信控制**服务或者**重新初始化设备**服务重新使能设备的通信。

16.1.1 结构

表 16-1 表示**设备通信控制**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 16-1　设备通信控制服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
时间期限(Time Duration)	U	U(=)		
使能/禁止(Enable/Disable)	M	M(=)		
密码(Password)	U	U(=)		

Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

16.1.1.1 参数

这个参数为**设备通信控制**有证实服务请求传送参数值。

16.1.1.1.1 时间期限

这个可选的参数是 16 位无符号整型类型，表示为一个时间分钟的数值，说明要求远程设备忽略除了**设备通信控制**服务 APUD 和**重新初始化设备**服务 APDU 之外的所有 APDU 的时间期间。如果本参数不存在，则认为时间期限是不确定的，意味着只有一个明确的**设备通信控制**服务 APDU 或者**重新初始化设备**服务 APDU 可以重新使能设备的通信。如果‘**使能/禁止**’参数具有 ENABLE 值，则忽略‘**时间期限**’参数，并且认为期间期限是不确定的。

16.1.1.1.2 使能/禁止

这个参数是枚举类型，取值为 ENABLE 或者 DISABLE。这个参数用来表示响应的 BACnet 用户是使能还是禁止网络接口的通信。如果这个参数值是 ENABLE，则通信将被使能。如果这个参数值是 DISABLE，则通信将被禁止。当网络通信被禁止时，设备只能处理**设备通信控制**服务和**重新初始化设备**服务 APDU，而不会起始任何报文。

16.1.1.1.3 密码

这个可选的参数是一个不超过 20 个字符的字符串。对于那些要求进行密码检验的设备，如果没有密码或者密码不正确，则不能执行本服务。对于那些不要求密码的设备，忽略这个参数。

16.1.1.2 Result(+)

参数‘**Result(+)**’指明服务请求已经成功。

16.1.1.3 Result(-)

参数‘**Result(-)**’指明服务请求失败。失败的原因在‘**错误类型**’参数中说明。

16.1.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

16.1.2 服务过程

响应的 BACnet 用户检验了包括密码在内的请求的合法性之后，响应一个‘**Result(+)**’服务原语，如果‘**使能/禁止**’参数是 DISABLE，停止对除了**设备通信控制**报文和**重新初始化设备**报文之外的任何报文的响应，也停止起始报文。通信被禁止的状态一直延续到下列两种情况之一发生时为止：‘**时间期限**’参数值规定的时间到期，或者收到一个具有‘**使能/禁止**’= ENABLE 的**设备通信控制**报文，或者一个**重新初始化设备**报文。如果响应的 BACnet 用户没有时钟并且‘**时间期限**’参数没有设置为“不确定”，则忽略这个 APDU，并且发送一

个‘Result(-)’服务原语。如果要求使用密码，但是密码不合法，或者密码不存在，则忽略这个 APDU，并且发送一个‘Result(-)’响应原语。

16.2 有证实专有传输（ConfirmedPrivateTransfer）服务

一个客户端的 BACnet 用户使用**有证实专有传输**服务调用远程设备中的专有的或者非标准的服务。本标准不定义给定设备可能提供的专有服务，**专有传输**服务提供一种机制，使用标准化的方法说明某个专有服务。对于这些服务唯一要求的参数是生产商的标识代码和一个服务标号。如果需要，每种服务都可以增加参数，本标准不定义增加的参数的形式和内容。生产商的标识代码和服务标号一起可以明确地标识后续的 APDU 要求传送信息的目标，或者标识远程设备基于后续 APDU 中参数所完成的服务。

生产商的标识代码是由 ASHRAE 分配给生产商的唯一代码。对于这个代码的管理机制不在本标准中定义。参见第 23 节。

16.2.1 有证实专有传输服务结构

表 16-2 表示**有证实专有传输**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 16-2　有证实专有传输服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
生产商 ID(Vendor ID)	M	M(=)		
服务标号(Service Number)	M	M(=)		
服务参数(Parameters)	U	U(=)		
Result(+)			S	S(=)
生产商 ID(Vendor ID)			M	M(=)
服务标号(Service Number)			M	M(=)
附加结果(Result Block)			C	C(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)
生产商 ID(Vendor ID)			M	M(=)
服务标号(Service Number)			M	M(=)
错误参数(Error Parameters)			M	M(=)

16.2.1.1 参数

这个参数为**有证实专有传输**有证实服务请求传送参数值。

16.2.1.1.1 生产商 ID

这个参数是无符号整型类型，表示一个唯一的生产商标识代码，这个代码是要执行的生产商专有服务的类型所需要的。

16.2.1.1.2 服务标号

这个参数是无符号整型类型，表示要执行的服务。

16.2.1.1.3 服务参数

这个可选的参数传送附加的参数，这些参数是由‘**生产商 ID**’和‘**服务标号**’指定的服务所需要的。这些参数的数据类型和意义由生产商自行确定。

16.2.1.2 Result(+)

参数‘Result(+)’指明服务请求已经成功。一个成功的结果指明请求的 APDU 已经收到，并且能够执行指定的专有服务。

16.2.1.2.1 生产商 ID

这个参数是无符号整型类型，表示一个唯一的生产商标识代码，这个代码是为得出这个结果的生产商专有服务所设定的。

16.2.1.1.2 服务标号

这个参数是无符号整型类型，表示得出这个结果的专有服务。

16.2.1.2.3 附加结果

这个条件的参数是一个任何类型的列表，传送执行请求的服务得到的附加的结果。这些结果的意义由生产商自行确定。

16.2.1.3 Result(-)

参数‘Result(-)’指明服务请求失败。失败的原因在‘**错误类型**’参数中说明。

16.2.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

16.2.1.3.2 生产商 ID

这个参数是无符号整型类型，表示一个唯一的生产商标识代码，这个代码是为得出这个错误结果的生产商专有服务所设定的。

16.2.1.3.3 服务标号

这个参数是无符号整型类型，表示得出这个错误结果的专有服务。

16.2.1.3.4 错误参数

这个可选的参数传送执行请求的服务得到的附加的错误结果。这些结果的意义由生产商自行确定。

16.2.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，进行执行指定的专有服务请求的操作。如果操作成功，发送一个‘Result(+)’响应原语。如果请求失败，发送一个‘Result(-)’

响应原语。

16.3 无证实专有传输（UnconfirmedPrivateTransfer）服务

一个客户端的 BACnet 用户使用**无证实专有传输**服务调用远程设备中的专有的或者非标准的服务。本标准不定义给定设备可能提供的专有服务，**专有传输**服务提供一种机制，使用标准化的方法说明某个专有服务。对于这些服务唯一要求的参数是生产商的标识代码和一个服务标号。如果需要，每种服务都可以增加参数，本标准不定义增加的参数的形式和内容。生产商的标识代码和服务标号一起可以明确地标识后续的 APDU 要求传送信息的目标，或者标识远程设备基于后续 APDU 中参数所完成的服务。

生产商的标识代码是由 ASHRAE 分配给生产商的唯一代码。对于这个代码的管理机制不在本标准中定义。参见第 23 节。

16.3.1 无证实专有传输服务结构

表 16-3 表示**无证实专有传输**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 16-3　无证实专有传输服务原语结构

参数名称	Req	Ind
参数(Argument)	M	M(=)
生产商 ID(Vendor ID)	M	M(=)
服务标号(Service Number)	M	M(=)
服务参数(Parameters)	U	U(=)

16.3.1.1 参数

这个参数为**无证实专有传输**服务请求传送参数值。

16.3.1.1.1 生产商 ID

这个参数是无符号整型类型，表示一个唯一的生产商标识代码，这个代码是要执行的生产商专有服务的类型所需要的。

16.3.1.1.2 服务标号

这个参数是无符号整型类型，表示要执行的服务。

16.3.1.1.3 服务参数

这个可选的参数传送附加的参数，这些参数是由‘**生产商 ID**’和‘**服务标号**’指定的服务所需要的。这些参数的数据类型和意义由生产商自行确定。

16.3.2 服务过程

因为这是一个无确认的服务，所以没有响应原语。生产商自行确定如何对这个服务请求进行响应。

16.4 重新初始化设备（ReinitializeDevice）服务

一个客户端的 BACnet 用户使用**重新初始化设备**服务向一个远程设备发送指令，指示这个设备重新冷启动，或者重新热启动到某个预设的初始状态。本服务主要由操作者用来进行设备诊断。因为这个服务的性质，可以要求在响应的 BACnet 用户执行这个服务之前，输入密码。

16.4.1 结构

表 16-4 表示**重新初始化设备**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 16-4　重新初始化设备服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
设备重新初始化的状态(Reinitialized State of Device)	M	M(=)		
密码(Password)	U	U(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

16.4.1.1 参数

这个参数为**重新初始化设备**有证实服务请求传送参数值。

16.4.1.1.1 设备重新初始化的状态

这个参数允许客户端用户指定设备在重新初始化之后所希望的状态。这个参数的值是下列二者之一：**冷启动（COLDSTART）**或者**热启动（WARMSTART）**。热启动是指重启动设备并开始工作之后，所有数据和程序正常保留。冷启动的准确意义由生产商自行定义。

16.4.1.1.2 密码

这个可选的参数是一个不超过 20 个字符的字符串。对于那些要求进行密码检验的设备，如果没有密码或者密码不正确，则不能执行本服务。对于那些不要求密码的设备，忽略这个参数。

16.4.1.2 Result(+)

参数‘**Result(+)**’指明服务请求已经成功。

16.4.1.3 Result(-)

参数‘**Result(-)**’指明服务请求失败。失败的原因在‘**错误类型**’参数中说明。

16.4.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

16.4.2 服务过程

响应的 BACnet 用户检验了包括密码在内的请求的合法性之后，占先在所有的未完成请求之前，响应一个‘**Result(+)**’服务原语，并且立即执行停止系统运行的应用程序，然后按照在请求中由请求的 BACnet 用户所规定的方式重新初始化设备。如果服务请求要求热启动，而设备由于自己的初始特征记录正在进行中而没有准备好进行热启动，则发送一个‘**Result(-)**’服务原语。如果要求使用密码，但是密码不合法，或者密码不存在，则发送一个‘**Result(-)**’响应原语。

16.5 有证实文本报文（ConfirmedTextMessage）服务

一个客户端的BACnet用户使用**有证实文本报文**服务向另一个BACnet设备发送一个文本报文。这个服务不是一个广播服务或者多播服务。这个服务用于收到的文本报文要求有证实的情况。这个证实并不保证操作者已经看到了这个报文。报文可以被分为正常报文和紧急报文，从而可以确定优先级。另外，可以可选的为文本报文划分数字类别代码或者类别标识串。这个分类可以被接收的 BACnet 设备用来确定如何处理文本报文。例如，报文类可以指定某个输出设备打印文本，或者指定接收文本之后的一组操作。如何进行分类由生产商自行确定。

16.5.1 有证实文本报文服务结构

表 16-5 表示**有证实文本报文**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 16-5　有证实文本报文服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
文本报文源设备(Text Message Source Device)	M	M(=)		
报文类(Message Class)	U	U(=)		
报文优先级(Message Priority)	M	M(=)		
报文(Message)	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

16.5.1.1 参数

这个参数为**有证实文本报文**服务请求传送参数值。

16.5.1.1.1 文本报文源设备

这个参数是 **BACnet 对象标识符**类型，传送起始这个文本报文的设备的**设备**对象的**对象**

标识符属性值。

16.5.1.1.2 报文类

如果这个参数存在，它表示接收的报文的类别。这个参数的数据类型可以选择为是无符号整型或者字符串。关于这个参数选用的数值以及它的意义，由生产商自行确定。

16.5.1.1.3 报文优先级

这个参数是枚举类型，指明关于报文处理的优先级：

{**正常（NORMAL），紧急（URGENT）**}。

16.5.1.1.4 报文

这个参数是字符串类型，用于传送文本报文。

16.5.1.2 Result(+)

参数‘Result(+)’指明服务请求已经成功。

16.5.1.3 Result(-)

参数‘Result(-)’指明服务请求失败。失败的原因在‘**错误类型**’参数中说明。

16.5.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

16.5.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，进行本地分派给指定的‘**报文类**’的报文处理操作，并且发送一个‘Result(+)’服务原语。如果服务请求不能被执行，则发送一个‘Result(-)’服务请求，指明遇到的错误。

除了要求进行关于成功或者失败的响应之外，采用什么样的操作来响应这个通知由生产商自行确定。然而，通常，接收报文的设备要取出‘**报文**’参数中的文本，并且显示、打印、或者根据‘**报文类**’参数指定的类别归档。如果‘报文类’参数被省略，则可以假设报文为普通类别。如果‘**报文优先级**’是**紧急**，则认为这个报文比已有的正在等待处理的**正常**报文要重要。

16.6 无证实文本报文（UnconfirmedTextMessage）服务

一个客户端的BACnet用户使用**无证实文本报文**服务向一个或者多个BACnet设备发送一个文本报文。这个服务可以是广播服务和多播服务，也可以只用来向单个接收者发送报文。这个服务用于收到的文本报文不要求确认的情况。报文可以被分为正常报文和紧急报文，从而可以确定优先级。另外，可以可选的为文本报文划分数字类别代码或者类别标识串。这个分类可以被接收的 BACnet 设备用来确定如何处理文本报文。例如，报文类可以指定某个输出设备打印文本，或者指定接收文本之后的一组操作。如何进行分类由生产商自行确定。

16.6.1 无证实文本报文服务结构

表 16-6 表示**无证实文本报文**服务原语的结构，表中所用的术语和符号在 5.6 节中给出

了解释。

表 16-6　无证实文本报文服务原语结构

参数名称	Req	Ind
参数(Argument)	M	M(=)
文本报文源设备(Text Message Source Device)	M	M(=)
报文类(Message Class)	U	U(=)
报文优先级(Message Priority)	M	M(=)
报文(Message)	M	M(=)

16.6.1.1 参数

这个参数为**无证实文本报文**服务请求传送参数值。

16.6.1.1.1 文本报文源设备

这个参数是 BACnet **对象标识符**类型，传送起始这个文本报文的设备的**设备**对象的**对象标识符**属性值。

16.6.1.1.2 报文类

如果这个参数存在，它表示接收的报文的类别。这个参数的数据类型可以选择为是无符号整型或者字符串。关于这个参数选用的数值以及它的意义，由生产商自行确定。

16.6.1.1.3 报文优先级

这个参数是枚举类型，指明关于报文处理的优先级：

{**正常**，**紧急**}。

16.6.1.1.4 报文

这个参数是字符串类型，用于传送文本报文。

16.6.2 服务过程

因为这是一个无证实的服务，没有响应原语。采用什么样的操作来响应这个通知由生产商自行确定。然而，通常，接收报文的设备要取出‘**报文**’参数中的文本，并且显示、打印、或者根据‘**报文类**’参数指定的类别归档。如果‘**报文类**’参数被省略，则可以假设报文为普通类别。如果‘**报文优先级**’是**紧急**，则认为这个报文比已有的正在等待处理的**正常**报文要重要。

16.7 时间同步（TimeSynchronization）服务

一个请求的 BACnet 用户使用**时间同步**服务向一个远程 BACnet 设备通告正确的当前时间。这个服务可以是广播服务和多播服务，也可以只用来向单个接收者发送通告。这个服务的目的是向接收者通告正确的当前时间,因此,设备可以将自己的内部时钟与其它设备同步。

16.7.1 时间同步服务结构

表 16-7 表示**时间同步**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 16-7　时间同步服务原语结构

参数名称	Req	Ind
参数(Argument)	M	M(=)
时间(Time)	M	M(=)

16.7.1.1 参数

这个参数为**时间同步**服务请求传送参数值。

16.7.1.1.1 时间

这个参数是 **BACnet 日期时间**类型，传送在发送这个服务请求设备中的时钟的当前日期和时间。

16.7.2 服务过程

因为这是一个无证实的服务，没有响应原语。收到**时间同步**服务指示的设备更新本地的时间表达式，并且在**设备**对象的**本地时间**属性和**本地日期**属性中反映出更新改变。

当操作者在请求中调用这个服务完成时间同步的功能时，不存在使用上的任何限制。但是在其它的情况下，这个服务的使用受**设备**对象中**时间同步接收者**属性值的控制。如果**时间同步接收者**列表长度为 0，则设备不能自动发送**时间同步**请求。如果**时间同步接收者**列表长度为 1 或者多个，设备可以自动发送**时间同步**请求，但是只发送给**时间同步接收者**列表中已有的设备。

16.8 Who-Has 和 I-Have 服务

一个发送的 BACnet 用户使用 **Who-Has** 服务确定一些其它 BACnet 设备的**设备**对象标识符和网络地址，这些设备的本地数据库中包含有具有给定的**对象名称**属性或者给定的**对象标识符**属性的对象。设备使用 **I-Hava** 服务响应 **Who-Has** 服务请求，或者通告自己有一个具有给定的**对象名称**属性或者**对象标识符**属性的对象。可以在任何时候发送 **I-Hava** 服务请求，并不要求一定要在接收到 **Who-Has** 服务请求之后才能够使用。**Who-Has** 服务和 **I-Hava** 服务是无证实的服务。

16.8.1 Who-Has 服务结构

表 16-8 表示 **Who-Has** 服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 16-8　Who-Has 服务原语结构

参数名称	Req	Ind
参数(Argument)	M	M(=)
设备实例低阈值范围(Device Instance Range Low Limit)	U	U(=)
设备实例高阈值范围(Device Instance Range High Limit)	U	U(=)

对象标识符(Object Identifier)	S	S(=)
对象名称(Object Name)	S	S(=)

16.8.1.1 参数

这个参数为 Who-Has 无证实服务请求传送参数值。

16.8.1.1.1 设备实例低阈值范围

这个参数是无符号整型类型，取值范围是 0－4,194,303。这个参数与‘**设备实例高阈值范围**’参数一起定义有资格使用 I-Hava 服务响应本服务的设备，这些设备还满足 16.8.1.3 节和 16.8.1.4 节规定的‘**对象标识符**’准则或者‘**对象名称**’准则。如果‘**设备实例低阈值范围**’参数存在，则‘**设备实例高阈值范围**’参数也必须存在。只有那些设备**对象标识符**属性值实例在范围(‘**设备实例低阈值范围**’ ≤ 对象标识符属性值实例 ≤ ‘**设备实例高阈值范围**’)之内的设备有资格进行响应。‘**设备实例低阈值范围**’参数值小于等于‘**设备实例高阈值范围**’参数值。如果省略‘**设备实例低阈值范围**’和‘**设备实例高阈值范围**’参数，则所有接收到这个报文的设备都有资格使用 I-Hava 服务进行响应。

16.8.1.1.2 设备实例高阈值范围

这个参数是无符号整型类型，取值范围是 0－4,194,303。这个参数与‘**设备实例低阈值范围**’参数一起定义有资格使用 I-Hava 服务响应本服务的设备，这些设备还满足 16.8.1.3 节和 16.8.1.4 节规定的‘**对象标识符**’准则或者‘**对象名称**’准则。如果‘**设备实例高阈值范围**’参数存在，则‘**设备实例低阈值范围**’参数也必须存在。只有那些设备**对象标识符**属性值实例在范围(‘**设备实例低阈值范围**’ ≤ 对象标识符属性值实例 ≤ ‘**设备实例高阈值范围**’)之内的设备有资格进行响应。‘**设备实例高阈值范围**’参数值大于等于‘**设备实例低阈值范围**’参数值。如果省略‘**设备实例低阈值范围**’和‘**设备实例高阈值范围**’参数，则所有接收到这个报文的设备都有资格使用 I-Hava 服务进行响应。

16.8.1.1.3 对象标识符

这个参数是 BACnet **对象标识符**类型，传送本服务正在定位的对象的**对象标识符**属性值。如果省略这个参数，则必须存在‘**对象名称**’参数。如果这个参数存在，则只有那些包含有其**对象标识符**属性值与这个参数值匹配的对象的设备，同时又满足 16.8.1.1 节和 16.8.1.2 节规定的资格要求，可以使用 I-Hava 服务进行响应。

16.8.1.1.4 对象名称

这个参数是字符串类型，传送本服务正在定位的对象的**对象名称**属性值。如果省略这个参数，则必须存在‘**对象标识符**’参数。如果这个参数存在，则只有那些包含有其**对象名称**属性值与这个参数值匹配的对象的设备，同时又满足 16.8.1.1 节和 16.8.1.2 节规定的资格要求，可以使用 I-Hava 服务进行响应。

16.8.2 Who-Has 服务过程

发送方 BACnet 用户通常使用广播地址发送 Who-Has 服务请求。如果‘**设备实例低阈值范围**’和‘**设备实例高阈值范围**’参数存在，则那些接收到这个报文的 BACnet 用户设备中，

设备的**对象标识符**属性值实例在范围(‘**设备实例低阈值范围**’ ≤ 对象标识符属性值实例 ≤ ‘**设备实例高阈值范围**’)之内的，有资格进行响应。如果‘**对象名称**’参数存在，则只有那些有资格的设备，同时又包含有其**对象名称**属性值与‘**对象名称**’参数值匹配的对象的设备，使用 I-Hava 服务请求进行响应。如果‘**对象标识符**’参数存在，则只有那些有资格的设备，同时又包含有其**对象标识符**属性值与‘**对象标识符**’参数值匹配的对象的设备，使用 I-Hava 服务请求进行响应。

16.8.3 I-Have 服务结构

表 16-9 表示 I-Have 服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 16-9 I-Have 服务原语结构

参数名称	Req	Ind
参数(Argument)	M	M(=)
设备标识符(Device Identifier)	M	M(=)
对象标识符(Object Identifier)	M	M(=)
对象名称(Object Name)	M	M(=)

16.8.3.1 参数

这个参数为 I-Have 无证实服务请求传送参数值。

16.8.3.1.1 设备标识符

这个参数是 BACnet **对象标识符**类型，是起始 I-Have 服务请求的设备的**设备对象标识符**。

16.8.3.1.2 对象标识符

这个参数是 BACnet **对象标识符**类型，传送在起始设备中作为定位而通告的对象的**对象标识符**。这个标识符对应被通告对象的**对象标识符**属性值。

16.8.3.1.3 对象名称

这个参数是字符串类型，传送在起始设备中作为定位而通告的对象的**对象名称**。这个标识符对应被通告对象的**对象名称**属性值。

16.8.4 I-Have 服务过程

发送方 BACnet 用户广播 I-Have 服务请求。根据应用的需要，这个广播可以是在本地网络范围内的广播，也可以是在一个远程网络范围内的广播，或者是在所有网络中的全局广播。如果正在被发送的 I-Have 服务请求是对前面接收到的一个 Who-Has 服务的响应，则要按照发送 Who-Has 服务的 BACnet 用户应该接收到的 I-Have 服务结果的方式发送这个 I-Have 服务请求。因为这个请求是无证实的，所以不要求进一步的操作。一个 BACnet 用户可以在任何时候发送一个 I-Have 服务请求。

16.9 Who-Is 和 I-Am 服务

一个发送方 BACnet 用户使用 **Who-Is** 服务确定在同一个互联网上的其它 BACnet 设备的**设备**对象标识符和网络地址，**Who-Is** 服务是一个无证实服务，这个服务可以用于下列两种情况：(1) 确定在网络上的所有设备的设这对象标识符和网络地址，或者(2) 确定某个已知其设备对象标识符但是不知其网络地址的设备的网络地址。**I-Am** 服务也是一个无证实的服务，这个服务用于响应 **Who-Is** 服务请求。**I-Am** 服务也可以在任何时候被发送，并不一定要求只有接收到 **Who-Is** 服务请求之后才能够发送 **I-Am** 服务请求。特别是，当一个设备刚启动之后，可能需要广播一个 **I-Am** 服务请求。如果发送 **I-Am** 服务请求的设备与接收这个报文的设备处于同一个本地网络，则可以从 **I-Am** 服务请求报文中的 MAC 地址推导出发送设备的网络地址。如果发送设备位于一个远程网络，则可以从报文的 NPCI 域中推导出设备的网络地址。

16.9.1 Who-Is 服务结构

表 16-10 表示 **Who-Is** 服务原语的结构,表中所用的术语和符号在 5.6 节中给出了解释。

表 16-10　Who-Is 服务原语结构

参数名称	Req	Ind
参数(Argument)	M	M(=)
设备实例低阈值范围(Device Instance Range Low Limit)	U	U(=)
设备实例高阈值范围(Device Instance Range High Limit)	U	U(=)

16.9.1.1 参数

这个参数为 **Who-Is** 无证实服务请求传送参数值。

16.9.1.1.1 设备实例低阈值范围

这个参数是无符号整型类型，取值范围是 0－4,194,303。这个参数与‘**设备实例高阈值范围**’参数一起定义有资格使用 **I-Am** 服务响应本服务的设备。如果‘**设备实例低阈值范围**’参数存在，则‘**设备实例高阈值范围**’参数也必须存在。只有那些设备**对象标识符**属性值实例在范围(‘**设备实例低阈值范围**’ ≤ 对象标识符属性值实例 ≤ ‘**设备实例高阈值范围**’)之内的设备有资格进行响应。‘**设备实例低阈值范围**’参数值小于等于‘**设备实例高阈值范围**’参数值。如果省略‘**设备实例低阈值范围**’和‘**设备实例高阈值范围**’参数，则所有接收到这个报文的设备都有资格使用 **I-Am** 服务进行响应。

16.9.1.1.2 设备实例高阈值范围

这个参数是无符号整型类型，取值范围是 0－4,194,303。这个参数与‘**设备实例低阈值范围**’参数一起定义有资格使用 **I-Am** 服务响应本服务的设备。如果‘**设备实例高阈值范围**’参数存在，则‘**设备实例低阈值范围**’参数也必须存在。只有那些设备**对象标识符**属性值实例在范围(‘**设备实例低阈值范围**’ ≤ 对象标识符属性值实例 ≤ ‘**设备实例高阈值范围**’)之内的设备有资格进行响应。‘**设备实例高阈值范围**’参数值大于等于‘**设备实例低阈值范围**’参数值。如果省略‘**设备实例低阈值范围**’和‘**设备实例高阈值范围**’参数，则所有接收到这个报文的设备都有资格使用 **I-Am** 服务进行响应。

16.9.2 Who-Is 服务过程

发送方 BACnet 用户通常使用广播地址发送 **Who-Is** 服务请求。如果省略了‘**设备实例低阈值范围**’和‘**设备实例高阈值范围**’参数，则所有这个报文的 BACnet 用户设备都使用 **I-Am** 服务单独响应自己的设备对象标识符。如果‘**设备实例低阈值范围**’和‘**设备实例高阈值范围**’参数存在，则那些接收到这个报文的 BACnet 用户设备中，设备的**对象标识符**属性值实例在范围(‘**设备实例低阈值范围**’ ≤ 对象标识符属性值实例 ≤ ‘**设备实例高阈值范围**’)之内的设备，使用 **I-Am** 服务请求响应它们的设备对象标识符。

16.9.3 I-Am 服务结构

表 16-11 表示 **I-Am** 服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 16-11 I-Am 服务原语结构

参数名称	Req	Ind
参数(Argument)	M	M(=)
I-Am 设备标识符(I-Am Device Identifier)	M	M(=)
最大 APDU 长度支持(Max APDU Length Accepted)	M	M(=)
分段支持(Segmentation Supported)	M	M(=)
生产商标识符(Vendor Identifier)	M	M(=)

16.9.3.1 参数

这个参数为 **I-Am** 无证实服务请求传送参数值。

16.9.3.1.1 I-Am 设备标识符

这个参数是 **BACnet 对象标识符**类型，是起始 **I-Am** 服务请求的设备的**设备对象标识符**。

16.9.3.1.2 最大 APDU 长度支持

这个参数是无符号整型类型，传送可以包含在一个单独的、不可分割的 APDU 中的最大的字节数量。这个参数值应该与**设备**对象的**最大 APDU 长度支持**属性值相同。参见 12.9.17 节。

16.9.3.1.3 分段支持

这个参数是 **BACnet 分段**类型，传送起始 **I-Am** 服务请求的设备处理分段报文的能力。这个参数值应该与**设备**对象的**分段支持**属性值相同。参见 12.9.18 节。

16.9.3.1.4 生产商标识符

这个参数是 16 位无符号整型类型，传送起始 **I-Am** 服务请求的设备的生产商的标识符。这个参数值应该与**设备**对象的**生产商标识符**属性值相同。参见 12.9.6 节和 23 节。

16.9.4 I-Am 服务过程

发送方 BACnet 用户广播 **I-Am** 服务请求。根据应用的需要，这个广播可以是在本地网络范围内的广播，也可以是在一个远程网络范围内的广播，或者是在所有网络中的全局广播。如果正在被广播的 **I-Am** 服务请求是对前面接收到的一个 **Who-Is** 服务的响应，则要按照发送 **Who-Is** 服务的 BACnet 用户应该接收到的 **I-Am** 服务结果的方式发送这个 **I-Am** 服务请求。因

为这个请求是无证实的，所以不要求进一步的操作。一个 BACnet 用户可以在任何时候发送一个 **I-Am** 服务请求。

17. 虚拟终端服务

一个客户端 BACnet 用户使用**虚拟终端**（以下将**虚拟终端**简称为 VT--译者注）服务建立一个与另一个 BACnet 设备上的应用程序服务器的连接。建立这个连接的目的是为了实现数据的面向字符双工交换。**VT** 服务通常被用来使一个 BACnet 设备上的应用程序作为一个“终端模拟器”，与另一个 BACnet 设备上的“操作员接口”应用程序交互。

这种类型的连接称之为 VT 会话（VT-session）。一旦建立了一个 VT 会话，客户端应用程序和服务器端应用程序都称之为 VT 用户（VT-user）。

VT 会话向 VT 用户提供下面一些特征和服务：

(a) 提供在两个对等 VT 用户之间建立能够进行虚拟终端信息交换的 VT 会话的手段；

(b) 提供在不同的 VT 类型之间进行选择的方法，包括对字符清单和编码的选择；

(c) 提供控制通信完整性的方法；

(d) 提供单方终止 VT 会话的方法；

(e) 提供交换虚拟终端数据的方法。

17.1 虚拟终端模型（Virtual Terminal Model）

每个 VT 会话都是建立在两个对等应用进程之间的双工连接。一旦在两个对等实体之间建立了一个会话，就可以通过使用**虚拟终端队列（Virtual Terminal Queues, VTQ）**进行数据交换。**VTQ** 是先进先出（FIFO）队列。两对等进程之间的数据流采用先进先出队列模型的目的，是为了使 VT 会话的任意一方上的应用进程的实现独立于另一方。通常，使用 BACnet 设备的操作者请求操作员接口应用程序建立一个与另一个 BACnet 设备上的操作员接口应用程序之间的 VT 会话。**VTQ** 模型使用两个先进先出队列，以使操作员接口，或者其它能提供双工数据交换的应用程序，通过 BACnet 协议进行工作。

图 17-1 表示一个操作员接口应用程序与一个诸如 CRT 终端的物理设备之间的典型关系。

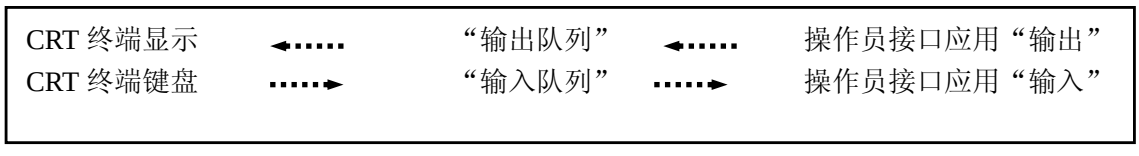


图 17-1 操作员接口程序和物理设备之间关系

图 17-2 表示这种模型是如何延伸使用 VT 队列概念的。

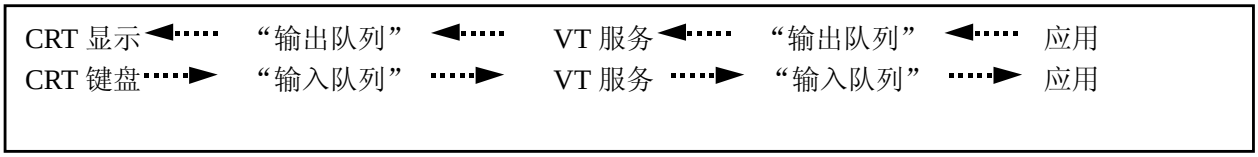


图 17-2 延伸 VT 模型以容纳队列

VT 会话每一端上的对等进程可以不是诸如 CRT 终端这样的物理设备的实际代理。**VTQ** 允许在每个 BACnet 设备中以灵活的方式实现。实际上，在每一个 BACnet 设备内可以有几个不同的进程，这些进程协调它们对 VT 服务的使用。例如，在多窗口操作员接口程序中，可以有几个窗口，每个窗口都和其它某个 BACnet 设备进行 VT 会话。所以，每个 VT 会话都是建立在两个唯一的进程之间的，而不是建立在两个 BACnet 设备之间。每一个会话由两个进程组成，这两个进程作为它们各自应用程序和各自 **VTQ** 的代理。这些代理进程和它们的 **VTQ** 一起称之为 VT 用户。图 17-3 表示这个数据流的模型。

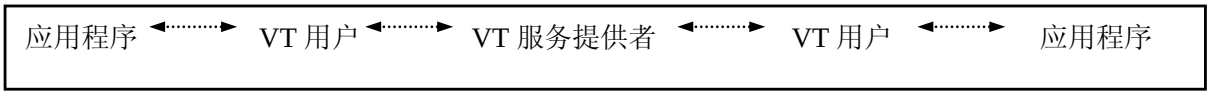


图 17-3 VT 数据流

一旦建立了 VT 会话，两个对等进程就可以通过各自的 **VTQ** 进行字符数据交换。通常，操作员所键入的字符将直接传送到输入队列，这个输入队列继续把字符传输到对等进程的输入队列。在这个过程中与操作员连接的本地设备不产生任何回送信号。对等应用程序在收到来自于它自己的输入队列的字符后，把字符放入它的输出队列，这样，字符可以发回到操作员设备的输出队列中，如此继续。特别地，本地操作员接口应用程序将负责产生全“屏幕”输出，包括回车控制和适当的字符回送。

虽然 VT 服务模型提供了一种如 17-3 图所示的对等体之间的连接，但一般来说，操作员总是使用一个 BACnet 设备建立与第二个 BACnet 设备之间的虚拟终端连接。第二个 BACnet 设备包含一个“操作员接口”应用程序，操作员的按键无过滤地送到这个程序。操作员接口应用程序的输出，通过 VT 服务被传输，最后显示给操作员。通常，与操作员真正相连的 BACnet 设备可以识别一些本地的“结束 VT 会话”的信息，但是除此之外，不会过滤按键操作结果。

17.1.1 基本服务

有三种基本服务：**VT 开启（VT-Open）**、**VT 关闭（VT-Close）**、**VT 数据（VT-Data）**。**VT 开启**服务用于在两个对等进程之间建立一个 VT 会话。**VT 关闭**服务用来释放已建立的 VT 会话。**VT 数据**服务用于在两个对等进程之间进行数据交换。

17.1.2 VT 类型

通过检测对等 BACnet **设备**对象，可以确定一个对等 VT 服务中 VT 的类型。BACnet **设备**对象有一个**虚拟终端类型支持**属性，使用**读属性**服务、**读多个属性**服务，或者**条件读属性**服务可以读取这个属性值，从而确定设备中所具有的有效 VT 类型。

17.1.3 活动 VT 会话

通过检测对等设备中的 BACnet **设备**对象，可以确定对等 VT 服务中是否存在活动的 VT 会话。BACnet **设备**对象有一个**活动虚拟终端会话**属性，使用**读属性**服务、**读多个属性**服务，或者**条件读属性**服务可以读取这个属性值，从而确定在这个设备中正在使用的 VT 会话的 ID。

17.1.4 VT 开启、VT 关闭和 VT 数据的状态图

在一个 VT 会话过程中，有三个操作状态：**空闲**（IDLE）状态、**数据交换**（DATA EXCHANGE）状态和**等待**（HOLD）状态。在**空闲**状态，不存在 VT 会话。如果使用 **VT 开启**服务创建了一个 VT 会话，则 VT 会话过程就进入**数据交换**状态。如果发生下列两个事件之一，则 VT 过程离开**数据交换**状态，这两个事件是：

(a) 成功地执行了 **VT 关闭**服务，释放了 VT 会话；

(b) **VT 数据**请求返回一个‘Result(-)’。

当一个 **VT 数据**请求还没有被确认时，会话进入**等待**状态。图 17-4 表示各状态之间的关系。

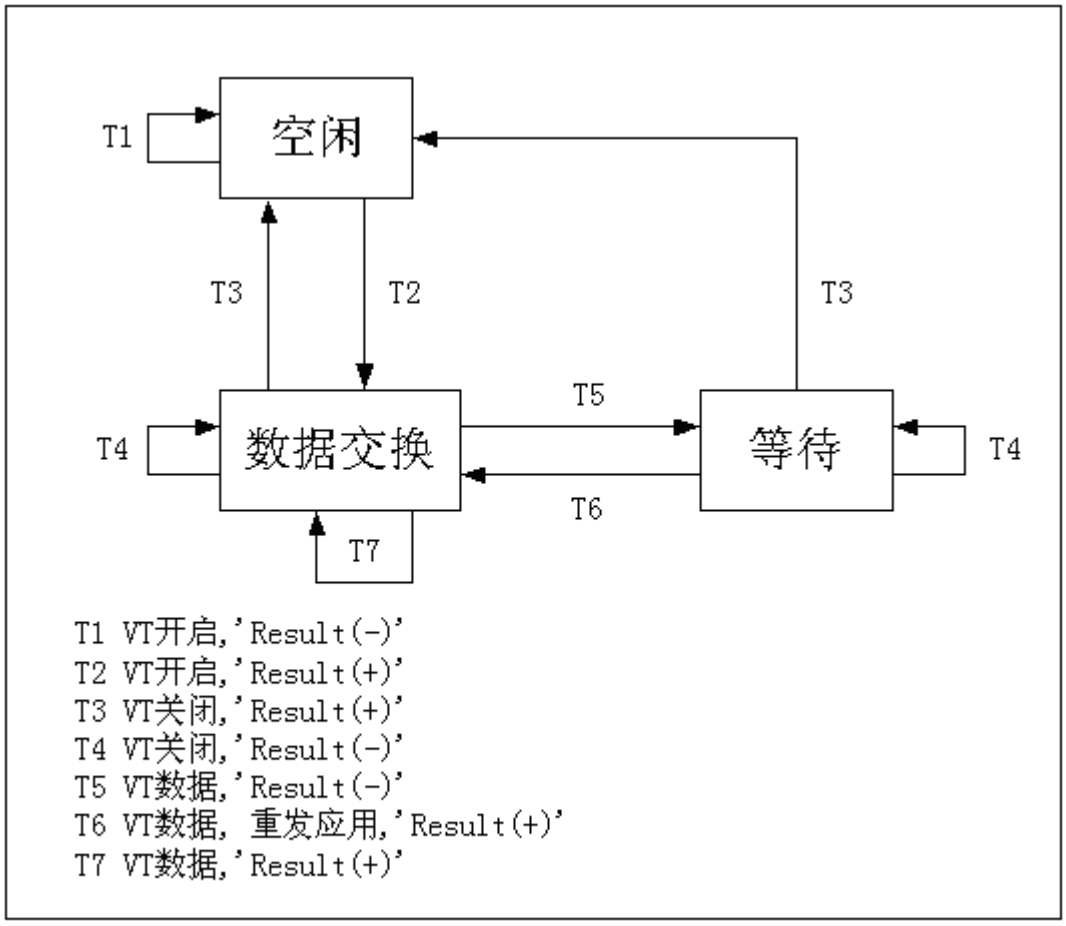


图17-4 虚拟终端服务状态图

17.1.5 VT 会话同步

参与VT会话的每一个对等体都有两个用于同步VT会话的VT数据标志(VT-data flags)。一个标志表示设备作为发送方 BACnet 用户的角色，是一个序列号，交替取 0 和 1 值，代表下一个要发送的**VT 数据**请求。当收到对于一个**VT 数据**请求的‘Result(+)’响应时，序列号以 2 为模加 1 变化。当 VT 会话建立时，这个标志被初始化为 0（第一个**VT 数据**请求使用序列号 0)。

另一个标志表示设备作为接收方 BACnet 用户的角色，是一个序列号，代表最后一个正确接收的 **VT 数据**请求。当 VT 会话建立时，这个标志被初始化为 0（下一个期待接收的 **VT 数据**请求指示应该具有序列号 0)。当收到一个其指示具有期待的序列号的 **VT 数据**请求，并且成功处理之后，序列号以 2 为模加 1 变化。

接收方 VT 会话过程还应该保存上一次收到的‘**全部收到的新数据（All New Data Accepted)**’和‘**接收到的字节数（Accepted Octet Count)**’参数。如果对 **VT 数据**指示的‘Result(+)’响应发生丢失，就需要这些数据。如果收到一个 **VT 数据**指示，检查其中的‘**VT 数据标志（ VT-Data Flag)**’参数和先前已收到并记录在这个 VT 会话过程中的‘**VT 数据标志**’是否相同，从而判断响应是否丢失。

17.1.6 VT 会话标识符

每个 VT 会话有两个会话标识符，一个是‘**本地 VT 会话标识符（Local VT Session Identifier)**’，另一个是‘**远程 VT 会话标识符（Remote VT Session Identifier)**’。这两个标识符提供了一种方法，能够将某个 **VT 数据**请求的数据同处理它的进程相联系。**VT 开启**服务的一个功能就是建立这两个会话标识符的值。每一个设备选择自己的‘**本地 VT 会话标识符**’，不论设备在会话中的角色是客户还是服务器，这个标识符在设备的所有活动 VT 会话中是唯一的。通过 **VT 开启**服务的参数可以获得合适的‘**远程 VT 会话标识符**’。在 **VT 数据**被传送给一个远程设备时，这个设备的‘**远程 VT 会话标识符**’一起被传送到远程设备上，远程设备使用这个会话标识符来标识正确的 VT 会话。

17.2 VT 开启服务

VT 开启服务可以用来在两个对等 VT 用户之间建立一个 VT 会话。**VT 开启**服务请求包括一个 VT 类型，这个类型标识一组在会话中要使用的有关字符指令表和编码。

17.2.1 结构

表 17-1 表示 **VT 开启**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 17-1　VT 开启服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
VT 类型(VT-class)	M	M(=)		

本地 VT 会话标识符(Local VT Session Identifier)	M	M(=)		
Result(+)			S	S(=)
远程 VT 会话标识符(Remote VT Session Identifier)			M	M(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

17.2.1.1 参数

这个参数为 **VT 开启**确认服务请求传送参数值。

17.2.1.1.1 VT 类型

VT 类型参数标识希望建立的会话类别，标准的类别枚举值如下：

DEFAULT-TERMINAL

ANSI-X3.64

DEC-VT52

DEC-VT100

DEC-VT220

HP-700/94

IBM-3130

枚举值 DEFAULT-TERMINAL 指特性由 17.5 节所定义的终端。所有的 VT 用户都必须至少支持 DEFAULT-TERMINAL 这个 VT 类别。一个给定的 VT 用户也可以支持其它的 VT 类别。通过读取**设备**对象的**虚拟终端类型支持**属性值，可以知道用户支持的 VT 类别。

17.2.1.1.2 本地 VT 会话标识符

这个参数是无符号整型类型，取值范围是 0－255，表示在请求的 VT 用户中的一个 VT 会话。这个标识符可以用于从开启的 VT 接收输出的 **VT 数据**。这个标识符对于响应的 VT 用户来说是‘**远程 VT 会话标识符**’。

17.2.1.2 Result(+)

参数‘**Result(+)**’指明服务请求已经成功。一个成功的结果包含下列参数。

17.2.1.2.1 远程 VT 会话标识符

这个参数是无符号整型类型，取值范围是 0－255，表示在响应的 VT 用户中的一个 VT 会话。对于响应的 VT 用户来说，这个参数是‘**本地 VT 会话标识符**’。

17.2.1.3 Result(-)

参数‘**Result(-)**’指明服务请求完全失败。失败的原因在‘**错误类型**’参数中说明。

17.2.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

17.2.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，分配必要的资源建立一个 VT 会话。如果没有 VT 用户能提供期望的 VT 类别，则返回一个‘**Result(-)**’。如果有一个 VT 用户能提供期望的 VT 类别，则响应方 BACnet 用户就建立一个新的 VT 会话。如果 VT 用户没有足够的资源建立另一个 VT 会话，则返回一个‘**Result(-)**’。如果 VT 用户有足够的资源，则建立一个新的 VT 会话。本地 VT 数据标志按照 17.1.5 节中的规范进行初始化，新的 VT 会话标识符在‘**Result(+)**’响应中返回。

17.3 VT 关闭服务

VT 关闭服务用来释放在两个对等 VT 用户之间已经建立的一个 VT 会话。**VT 关闭**服务请求说明要释放的某个 VT 会话，或者一个 VT 会话列表。

17.3.1 结构

表 17-2 表示 **VT 关闭**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 17-2　VT 关闭服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
远程 VT 会话标识符列表 (List of Remote VT Session Identifier)	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)
VT 会话标识符列表(List of VT Session Identifier)			C	C(=)

17.3.1.1 参数

这个参数为 **VT 关闭**确认服务请求传送参数值。

17.3.1.1.1 远程 VT 会话标识符列表

这个参数是一个列表，其中有一个或者多个‘**远程 VT 会话标识符**’。每一个‘**远程 VT 会话标识符**’指明一个要释放连接的 VT 会话。

17.3.1.2 Result(+)

参数‘**Result(+)**’指明服务请求已经成功。

17.3.1.3 Result(-)

参数‘**Result(-)**’指明服务请求完全失败。失败的原因在‘**错误类型**’参数中说明。

17.3.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

17.3.1.3.2 VT 会话标识符列表

如果‘**错误类型**’参数返回一个 VT_SESSION_TERMINATION_FAILURE 的‘**错误代码**’，则必须包含本参数。如果‘**错误类型**’参数指明是某些其它的错误，则省略本参数。‘**VT 会话标识符列表**’参数是一个列表，其中包含有一个或者多个‘**VT 会话标识符**’。每一个‘**VT 会话标识符**’指明一个不能被释放连接的 VT 会话。返回的**会话标识符**相对于接收‘Result(-)’原语的设备（也就是请求方 VT 用户）来说，是本地的标识符。

17.3.2 服务过程

响应的 BACnet 用户检验了请求的合法性之后，执行释放‘**VT 会话标识符列表**’参数中指定的每个 VT 会话的操作。从响应方 BACnet 用户的观点来看，这些标识符都是‘**本地 VT 会话标识符**’。如果因为某种原因，有一个或者多个 VT 会话不能被释放，则释放所有能够被释放的会话，然后返回一个‘Result(-)’响应。如果所有的指定的 VT 会话都被成功释放，则返回一个‘Result(+)’响应。

17.4 VT 数据服务

VT 数据服务用来在两个已经建立了 VT 会话的对等 VT 用户之间交换数据。发送方 BACnet 用户向对等 VT 用户提供新输入数据，VT 用户可以接收这些新数据，也可以拒绝这些新数据。如果向数据被拒绝，则要等待发送方 BACnet 用户重新发送请求。

17.4.1 结构

表 17-3 表示 **VT 数据**服务原语的结构，表中所用的术语和符号在 5.6 节中给出了解释。

表 17-3　VT 数据服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
VT 会话标识符(VT-session Identifier)	M	M(=)		
VT 新数据(VT-new Data)	M	M(=)		
VT 数据标志(VT-data Flag)	M	M(=)		
Result(+)			S	S(=)
全部收到的新数据(All new Data Accepted)			M	M(=)
接收的字节数(Accepted Octet Count)			C	C(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

17.4.1.1 参数

这个参数为 **VT 数据**确认服务请求传送参数值。

17.4.1.1.1 VT 会话标识符

这个参数标识一个 VT 会话，其中，数据将要发送给这个 VT 会话。从请求方的 BACnet 用户的观点来看，这个参数是‘**远程 VT 会话标识符**’。从响应方 BACnet 用户的观点来看，这个参数是‘**本地 VT 会话标识符**’。

17.4.1.1.2 VT 新数据

这个参数规定将要发送给对等 VT 用户的新数据的字节。

17.4.1.1.3 VT 数据标志

这个参数是无符号整型类型，指明 **VT 数据**请求的期待序列号。这个参数的值应该是 0 或者 1。在同一个 VT 会话中，对于每一个新的 **VT 数据**请求，这个参数值交替改变。

17.4.1.2 Result(+)

参数‘**Result(+)**’指明服务请求已经成功。一个成功的结果包含有下列参数。

17.4.1.2.1 全部收到的新数据

这个参数是布尔类型，如果所有的‘**VT 新数据**’的字节都被对等 VT 用户所接收，这个参数等于 TRUE。在这种情况下，认为服务被完成。如果本参数等于 FALSE，则有一些‘**VT 新数据**’的字节不能够被对等 VT 用户所接收。通常，这种情况发生的原因是对等 VT 用户的资源有限。在这种情况下，等待 BACnet 用户重新发送 **VT 数据**请求。即使仅仅只有那些不能被接收的字节要传送，也需要等待新的请求。

17.4.1.2.2 接收的字节数

只有在‘**全部收到的新数据**’参数等于 FALSE 的情况下，本参数才存在。在这种情况下，本参数指明从服务请求的‘**VT 新数据**’参数中实际接收的字节的数目。如果‘**全部收到的新数据**’参数等于 TRUE，则省略本参数。

17.4.1.3 Result(-)

参数‘**Result(-)**’指明服务请求完全失败。失败的原因在‘**错误类型**’参数中说明。

17.4.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。参见 18 节。

17.4.2 服务过程

发送方 BACnet 用户使用一个 0 值‘**VT 数据标志**’发送初始的 **VT 数据**请求。对同一个会话来说，除了重发之外，其后的 **VT 数据**请求的‘**VT 数据标志**’在 0 和 1 之间交替取值。只有在收到了一个对先前 **VT 数据**请求的‘**Result(+)**’响应之后，才能发送取交替序列号的新 **VT 数据**。

接收方 BACnet 用户检验了请求的合法性之后，就搜寻由‘**VT 会话标识符**’参数所指定的会话。如果不能搜寻到指定的 VT 会话，则返回‘**Result(-)**’。如果能够找到指定的 VT 会话，而且会话接收到的‘**VT 数据标志**’和上一次收到的‘**VT 数据标志**’相同，则认为这是一个重复的 **VT 数据**请求。接收方 BACnet 用户丢弃这个数据，同时返回一个‘**Result(+)**’响应，其中包含从前一个 **VT 数据**请求中得到、并且保存这个 VT 会话过程中的‘**全部收到的**

新数据’参数值和‘**接收的字节数**’参数值。

如果能够找到指定的 VT 会话，而且会话接收到的‘**VT 数据标志**’和上一次收到的‘**VT 数据标志**’不相同，则认为这是一个新的 **VT 数据**请求。如果所有的数据都能加入会话的输入队列，则将数据加入排队队列，而且返回一个‘**Result(+)**’响应，其中‘**全部收到的新数据**’参数等于 TRUE，‘**接收的字节数**’参数不存在。如果所有的数据不能都进入会话的输入队列，则将数据把输入队列排满，丢弃剩余的数据，返回一个‘**Result(+)**’响应，其中的‘**全部收到的新数据**’参数等于 FALSE，‘**接收的字节数**’参数等于能够加入到队列的字节数。在每一种情况下，都要把‘**全部收到的新数据**’参数值和‘**接收的字节数**’参数值保存在这个 VT 会话过程中。

17.5 缺省终端的指标

每一个 VT 用户至少要实现一种 VT 类型，这就是 DEFAULT_TERMINAL（缺省终端）类型。缺省终端取通常使用的交互式终端设备功能的一个有限集。缺省终端的指标包括一个字符清单，控制功能，页尺寸设定。除此之外，再没有对 VT 用户的其它要求。

17.5.1 缺省终端字符清单

缺省终端字符清单定义为在单个字节值与它们的意义之间的一个特定映射关系。缺省终端字符清单包含三个项目，分别是：

(a) 一个特定的符号（SYMBOL），例如：“A”；

(b) 一个特定的隐含的控制功能（CONTROL），例如：回车；

(c) 一个空（NUL），例如：不用，忽略。

在缺省终端字符清单中规定为空的那些字节没有设定的意义，如果 VT 用户接收到这些字符，就忽略它们。缺省终端字符清单是 ASSCII 码的一个子集。表 17-4 给出缺省终端字符清单中每个字符的字节编码。“字节值”域以十进制表示字节编码，取值范围是 0－255。字节值的范围用两个十进制数表示，例如，000-006。

表 17-4 缺省终端字符清单

字节值	类型	意义
000-006	NUL	空
007	CONTROL	声音提示（BEL）
008	CONTROL	不删除后退（BS）
009	CONTROL	水平制表符（TAB）
010	CONTROL	下移（LF）
011-012	NUL	空
013	CONTROL	回车（CR）
014-031	NUL	空

032	SYMBOL	空格
033	SYMBOL	! 感叹号
034	SYMBOL	“ 双引号
035	SYMBOL	# 英镑符号
036	SYMBOL	$ 美元符号
037	SYMBOL	% 百分号
038	SYMBOL	& “与”符号
039	SYMBOL	‘ 省略符号
040	SYMBOL	(左圆括号
041	SYMBOL	) 右圆括号
042	SYMBOL	* 星号
043	SYMBOL	+ 加号
044	SYMBOL	, 逗号
045	SYMBOL	- 减号
046	SYMBOL	. 句号
047	SYMBOL	/ 正斜线
048	SYMBOL	0
049	SYMBOL	1
050	SYMBOL	2
051	SYMBOL	3
052	SYMBOL	4
053	SYMBOL	5
054	SYMBOL	6
055	SYMBOL	7
056	SYMBOL	8
057	SYMBOL	9
058	SYMBOL	: 冒号
059	SYMBOL	; 分号
060	SYMBOL	< 左尖括号（小于号）
061	SYMBOL	= 等号
062	SYMBOL	> 右尖括号（大于号）
063	SYMBOL	? 问号
064	SYMBOL	@
065	SYMBOL	A
066	SYMBOL	B

067	SYMBOL	C
068	SYMBOL	D
069	SYMBOL	E
070	SYMBOL	F
071	SYMBOL	G
072	SYMBOL	H
073	SYMBOL	I
074	SYMBOL	J
075	SYMBOL	K
076	SYMBOL	L
077	SYMBOL	M
078	SYMBOL	N
079	SYMBOL	O
080	SYMBOL	P
081	SYMBOL	Q
082	SYMBOL	R
083	SYMBOL	S
084	SYMBOL	T
085	SYMBOL	U
086	SYMBOL	V
087	SYMBOL	W
088	SYMBOL	X
089	SYMBOL	Y
090	SYMBOL	Z
091	SYMBOL	[左方括号
092	SYMBOL	\ 反斜线
093	SYMBOL	] 右方括号
094	SYMBOL	^ 脱字符
095	SYMBOL	_ 下划线
096	SYMBOL	` 重音标记符
097	SYMBOL	a
098	SYMBOL	b
099	SYMBOL	c
100	SYMBOL	d
101	SYMBOL	e

102	SYMBOL	f
103	SYMBOL	g
104	SYMBOL	h
105	SYMBOL	i
106	SYMBOL	j
107	SYMBOL	k
108	SYMBOL	l
109	SYMBOL	m
110	SYMBOL	n
111	SYMBOL	o
112	SYMBOL	p
113	SYMBOL	q
114	SYMBOL	r
115	SYMBOL	s
116	SYMBOL	t
117	SYMBOL	u
118	SYMBOL	v
119	SYMBOL	w
120	SYMBOL	x
121	SYMBOL	y
122	SYMBOL	z
123	SYMBOL	{ 左花括号
124	SYMBOL	| 垂直线段
125	SYMBOL	} 右花括号
126	SYMBOL	~ 代字号
127	CONTROL	不删除后退（DEL）
128-255	NUL	空

17.5.2 控制功能

在缺省终端字符清单中，有 6 个字节代码执行控制功能。在本协议中，控制功能表示，接收到这种控制代码的设备，要执行这个代码所隐含的操作。

17.5.2.1 字节代码 007

字节代码 007 表示一个声音提示（BEL）。通常，这个声音是发声器产生的一个声调音或者响铃信号。

17.5.2.2 字节代码 008 和 127

字节代码 008 表示一个不删除后退操作（BS）。这个操作使得虚拟“输出设备”的光标

向左移动一个字符位。如果光标位置处于左端顶位置或者一个线的开始位置，则 BS 操作不产生任何效果。这个操作只改变光标的当前位置，并不改写或者改变光标向左移动位置原来的字符。字节代码 127 的功能与 008 相同。

17.5.2.3 字节代码 013

字节代码 013 表示回车（CR）操作。这个操作使得光标重新定位在当前“线”的开始。在这个操作之后，接收到的字符将改写位于当前“线”上的已有的字符。

17.5.2.4 字节代码 010

字节代码 010 表示下移（LF）操作。这个操作使得光标移动到下一行线上，但是不改变光标在线内的位置。通常，这个代码与 CR 一起使用。

17.5.2.5 字节代码 009

字节代码 009 表示水平前移到下一个制表位（TAB）操作。这个操作使得光标移动到当前行的下一个制表位上。这个操作只改变光标的位置，并不会改写或者改变光标移动位置上原来的字符。每进行一次制表位操作，光标向右移动 8 个字符位置，如图 17-5 所示。

```
      1      2      3      4      5
12345678901234567890123456789012345678901234567890
      T      T      T      T      T      T
```

图17-5. VT制表位置

17.5.3 页尺寸设定

在缺省终端的规范中，只有两个对页尺寸的设定。第一个设定是，设定一页有 80 列。每个接收的 SYMBOL 字符占据一列。空字符对列位置不产生任何影响。CONTROL 字符对列位置的影响各不相同，在 17.5.2 节中已有规定。第二个设定是，每一页的长度没有限制。这个设定等价与一个有连续形式纸的打印机。

18. 错误、拒绝和中止代码

本节列出在 BACnet 协议中涉及到的所有错误，这些错误按照“**错误类**”进行分类。在每一个错误类别中，进一步按照“**错误代码**”进行分类。这样分类，使得应用程序可能按照两个尺度，对错误采取不同的补救方式。

18.1 错误类-设备（Error Class - DEVICE）

这种**错误类**涉及影响整个 BACnet 设备功能的环境。出现这种类别的错误，通常表示整个服务请求已经失败。

18.1.1 设备忙（DEVICE_BUSY ）

一个服务请求被暂时拒绝，因为它所寻址的 BACnet 设备希望处理更高优先级的进程，

而这所需要的时间超出了正常请求/确认的超时等待时间。

18.1.2 正在进行配置（CONFIGURATION_IN_PROGRESS）

一个服务请求被暂时拒绝，因为它所寻址的 BACnet 设备正在进行配置，这种对设备的配置可能是通过本地设备进行，也可能是通过其它的协议服务进行。

18.1.3 运行问题（OPERATIONAL_PROBLEM）

一个服务请求被暂时拒绝，因为它所寻址的 BACnet 设备检测到了阻碍执行请求的服务的运行问题。

18.1.4 其它（OTHER）

如果出现的设备错误不是上面所提到的，则返回这个错误代码。

18.2 错误类-对象（Error Class - OBJECT）

这种**错误类**涉及在验证、访问和操作 BACnet 对象时出现的错误，不论这些对象是否是 BACnet 定义的标准对象。因为这种错误一般与单个对象特性相关，所以不必发送整个服务请求已失败的信号。

18.2.1 不支持动态创建（DYNAMIC_CREATION_NOT_SUPPORTED）

这种错误是指试图用一种不能动态创建的对象类型创建一个对象。

18.2.2 无指定类型的对象（NO_OBJECT_OF_SPECIFIED_TYPE）

这种错误是指在 BACnet 设备中的对象数据库不能找到服务请求指定的对象类型的对象。

18.2.3 不支持对象删除（OBJECT_DELETION_NOT_PERMITED）

这种错误是指试图删除一个不能被删除，或者当前被保护的对象。

18.2.4 对象标识符已存在（OBJECT_INDENTIFIER_ALREADY_EXISTS）

这种错误是指试图用已经在使用的对象标识符来定义一个新的对象。

18.2.5 读访问拒绝（READ_ACCESS_DENIED）

这种错误是指试图用 BACnet 协议读服务对一个已被定义为不可访问的对象的属性进行读操作。

18.2.6 未知对象（UNKNOWN_OBJECT）

这种错误是指所寻址的 BACnet 设备对象数据库中不存在对象标识符所表示的对象。

18.2.7 不支持的对象类型（UNSUPPORTED_OBJECT_TYPE）

这种错误是指所寻址的 BACnet 设备不支持或不知道服务参数所指定的对象类型。

18.2.8 其它（OTHER）

当出现的对象错误不是以上所提到的，则返回这个错误代码。

18.3 错误类-属性（Error Class - PROPERTY）

这种**错误类**涉及在对 BACnet 对象的属性进行验证、访问和操作时出现的错误，不论这些对象属性是否是 BACnet 定义的标准对象属性。因为这种错误一般与单个属性有关，所以不必发送整个服务请求已失败的信息。

18.3.1 不支持字符集（CHARACTER_SET_NOT_SUPPORTED）

这种错误是指遇到了不支持的字符集中的字符串值。

18.3.2 选择准则不一致（INCONSISTENT_SELECTION_CRITERION）

这种错误是指属性的数据类型和‘**对象选择准则**’服务参数中指定的‘**比较值**’不一致。例如，如果一个模拟属性和一个布尔常数相比较，就会发生这样的错误，反之亦然。

18.3.3 无效数组索引（INVALID_ARRAY_INDEX）

这个错误是指试图使用超出范围的数组索引访问一个数组属性。

18.3.4 无效数据类型（INVALID_DATATYPE）

这个错误是指在一个服务参数中指定的属性值的类型与**属性标识符**指定的引用属性的类型不匹配。

18.3.5 读访问拒绝（READ_ACCESS_DENIED）

这种错误是指试图用 BACnet 协议读服务对一个已被定义为不可访问的属性进行读操作。

18.3.6 未知属性（UNKNOWN_PROPERTY）

这种错误是指所寻址的 BACnet 设备中的对象所支持的属性不包含服务参数中指定的**属性标识符**。

18.3.7 写访问拒绝（WRITE_ACCESS_DENIED）

这种错误是指试图用 BACnet 协议写服务向一个已被定义为不可访问的属性进行写操作。

18.3.8 值超出阈值（VALUE_OUT_OF_RANGE）

这种错误是指试图用超出阈值的值给属性赋值。

18.3.9 其它（OTHER）

如果出现的属性错误不是以上所提到的，则返回这个错误代码。

18.4 错误类-资源（Error Class - RESOURCES）

这种**错误类**涉及影响 BACnet 设备执行协议服务请求能力的设备资源方面的错误。

18.4.1 无空间创建对象（NO_SPACE_FOR_OBJECT）

这种错误是指在寻址的 BACnet 设备内没有足够的动态存储空间来创建一个对象。

18.4.2 无空间添加列表元素（NO_SPACE_TO_ADD_LIST_ELEMENT）

这种错误是指在寻址的 BACnet 设备内没有足够的动态存储空间用来进行在一个列表中增加一个元素的操作。

18.4.3 无空间写属性（NO_SPACE_TO_WRITE_PROPERTY）

这种错误是指在寻址的 BACnet 设备内没有足够的动态存储空间用来进行写属性值的操作。

18.4.4 其它（OTHER）

当出现的资源错误不是以上所提到的，则返回这个错误代码。

18.5 错误类-安全（Error Class - SECURITY）

这种**错误类**涉及安全性服务执行方面的错误。毫无例外，这种类型错误表示响应方 BACnet 用户完全不能执行要求的服务。这种错误是“致命”错误。

18.5.1 认证失败（AUTHENTICATION_FAILED）

这个错误是指服务提供者没有生成报文的认证信息。

18.5.2 不支持字符集（CHARACTER_SET_NOT_SUPPORTED）

这种错误是指遇到了不被支持的字符集中的字符串值。

18.5.3 不兼容的安全等级（INCOMPATIBLE_SECURPITY_LEVELS）

这种错误是指两客户的安全权限不相同。

18.5.4 无效的操作员名字（INVALID_OPERATOR_NAME）

这种错误是指‘操作员姓名’和已知道的操作员名字不符合。

18.5.5 密钥生成错误（KEY_GENERATION_ERROR）

这种错误是指密钥服务器不能生成一个**会话密钥**（Session Key, SK）。

18.5.6 密码错误（PASSWORD_FAILURE）

这种错误是指‘操作员姓名’和‘操作员口令’不匹配。

18.5.7 不支持安全性（SECURITY_NOT_SUPPORTED）

这种错误是指远程客户不支持任何 BACnet 安全机制。

18.5.8 超时（TIMEOUT）

这种错误是指在等待时间耗尽之前，还没有收到带有期待调用 ID（Invoke ID）的 APDU。

18.5.9 其它（OTHER）

当出现的安全错误不是以上所提到的，则返回这个错误代码。

18.6 错误类-服务（Error Class - SERVICES）

这种**错误类**涉及协议服务请求执行方面的错误，不论服务请求是否是 BACnet 定义的标准服务。毫无例外，这种类型错误表示响应方 BACnet 用户完全不能执行要求的服务。这种错误是“致命”错误。

18.6.1 不支持字符集（CHARACTER_SET_NOT_SUPPORTED）

这种错误是指遇到了不被支持的字符集中的字符串值。

18.6.2 文件访问拒绝（FILE_ACCESS_DENIED）

这种错误是指，当使用**基本读文件**或者**基本写文件**服务请求访问一个当前被写保护或者不能被访问的文件时，产生在响应中的错误指示。

18.6.3 参数不一致（INCONSISTENT_PARAMETERS）

这种错误是指服务请求中的两个或者多个参数互相矛盾。

18.6.4 无效的文件访问方法（INVALID_FILE_ACCESS_METHOD）

这个代码产生在对一个**基本读文件**或者**基本写文件**服务请求的响应中，指出这个**基本读文件**或者**基本写文件**服务请求中指定的‘**文件访问方法**’对于指定的文件无效。

18.6.5 无效的文件开始位置（INVALID_FILE_START_POSITION）

这个代码产生在对一个**基本读文件**或者**基本写文件**服务请求的响应中,指出这个**基本读文件**或者**基本写文件**服务请求指定了一个无效的‘**文件开始位置**’或者‘**文件开始记录**’参数。

18.6.6 无效的参数类型（INVALID_PARAMETER_DATATYPE）

这种错误是指服务参数指定的数据类型不适合这个参数。

18.6.7 无效的时间戳（INVALID_TIME_STAMP）

这种错误是指由**确认报警**服务请求传送的‘**时间戳**’参数和正在被确认的事件的最近发生时间不匹配。

18.6.8 属性非列表元素（PROPETY_IS_NOT_A_LIST）

这种错误是指试图使用**添加列表元素**或者**删除列表元素**服务访问一个非列表数据类型的属性。

18.6.9 缺少必要的参数（MISSING_REQUIRED_PARAMETER）

这种错误是指没有提供执行服务请求所需的参数。

18.6.10 服务请求拒绝（SERVICE_REQUST_DENIED）

这种错误是指请求方 BACnet 设备没有权限要求执行这个服务请求。

18.6.11 其它（OTHER）

如果出现的服务错误不是以上所提到的，则返回这个错误代码。

18.7 错误类-虚拟终端（Error Class - VT）

这种**错误类**涉及执行虚拟终端服务时出现的错误。

18.7.1 未知 VT 类别（UNKNOWN_VT_CLASS）

这种错误是指目标设备不能识别 **VT 开启**服务请求指定的‘VT 类别’。

18.7.2 未知 VT 会话（UNKNOWN_VT_SESSION）

这种错误是指目标设备不能识别 **VT 关闭**服务请求或者 **VT 数据**服务请求指定的‘**VT 会话 ID**’。

18.7.3 无可用的 VT 会话（NO_VT_SESSION_AVAILABLE）

这种错误是指因为资源的限制，目标设备不能完成一个 **VT 开启**请求。

18.7.4 VT 会话已经关闭（VT_SESSION_ALREADY_CLOSED）

这种错误是指试图关闭一个已经关闭了的 **VT 会话**。

18.7.5 释放 VT 会话失败（VT_SESSION_TERMINATION_FAILURE）

这种错误是指由于实现方面的原因，不能释放 **VT 关闭**服务请求指定的一个‘VT 会话’。

18.7.6 其它（OTHER）

如果出现的 VT 错误不是以上所提到的，则返回这个错误代码。

18.8 拒绝原因（Reject Reason）

只有确认请求 PDU 能够被拒绝，拒绝的原因有下面几种。

18.8.1 缓冲器溢出（BUFFER_OVERFLOW）

这个拒绝的原因是指超出了输入缓冲器容量。

18.8.2 参数不一致（INCONSISTENT_PARAMETERS）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 省略了一个应该存在的条件服务参数或者包含了一个不应该存在的条件服务参数。

18.8.3 无效参数类型（INVALID_PARAMETER_DATA_TYPE）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 中一个或多个服务参数的编码不符合正确的类型规范。

18.8.4 无效标志（INVALID_TAG）

这个拒绝的原因是指在分析一个报文时，遇到一个无效的标志。

18.8.5 缺少必要的参数（MISSING_REQUIRED_PARAMETER）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 至少缺少一个必需的服务参数。

18.8.6 参数超出阈值（PARAMETER_OUT_OF_RANGE）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 传送的参数值超出了服务所规定的范围。

18.8.7 过多参数（TOO_MANY_PARAMETER）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 中的服务参数的总数多于服务所规定的数量。

18.8.8 未定义枚举（UNDEFINED_ENUMERATION）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 中有一个或者多个服务参数解码得到的枚举是没有被该参数类型规范所定义的。

18.8.9 不支持的服务（UNRECOGNIZED_SERVICE）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 的**服务选择**域中所指定的服务是一个未知或者不被支持的服务。

18.8.10 其它（OTHER）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 包含一个非上述原因的语法错误。

18.9 中止原因（Abort Reason）

18.9.1 缓冲器溢出（BUFFER_OVERFLOW）

指出中止的原因是超出了输入缓冲器的容量。

18.9.2 在本状态下无效的 APDU（INVALID_APDU_IN_THIS_STATE）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 不是事务处理状态机当前状态的期待状态。

18.9.3 高优先级任务抢占（PREEMPTED_BY_HIGHER_PRIORITY_TASK）

这个中止的原因是因为要中止这个事务处理而运行更高优先级的进程。

18.9.4 不支持分段（SEGMENTATION_NOT_SUPPORTED）

这个代码产生在对一个有确认请求 APDU 的响应中，指出这个请求 APDU 中的分段位设置为 TRUE，但是接收设备不支持分段。

18.9.5 其它（OTHER）

如果中止是非上述原因所引起的，则返回这个错误代码。

19. 命令优先级机制

在楼宇自动控制系统中，一个对象可能被多个实体操作。例如，一个**二进制输出**对象的当前值可能被多个应用程序设置，诸如查询计量读数，优化启/停操作等等。每一个这样的应用程序都有一个明确的功能需要执行。当有两个或者多个应用程序都要操作一个属性值而发生冲突时，就需要在这些应用程序之间进行协调。协调处理的目的是要保证被多个程序（或者非程序）实体操作的对象达到期望的状态。例如，一个启/停程序可能规定某个**二进制输出**对象应该为 ON，而查询计量读数的程序可能规定同一个**二进制输出**对象为 OFF。在这种情况下，OFF 应该具有优先权。操作者可以不顾查询计量读数程序，强行设置这个**二进制输出**对象为 ON，在这种情况下，ON 就具有优先权。

在 BACnet 协议中，采用一个优先级机制提供这种协调处理方法。这个优先级机制根据整个系统要求，给命令实体分配不同的优先级的级别。每个包含有一个可命令属性的对象负责按照命令的优先级顺序执行相应的操作。任何这样的机制都必须在复杂性和稳定性之间进行折中，本协议确定优先级机制应该是简单有效，并且可以应用到非常简单的 BACnet 设备中。

下列的属性类型涉及优先级机制：

(a) **可命令**属性：每一个支持命令优先级的对象都有一个或者多个称之为“**可命令属性（Commandable Properties）**”的属性。这些属性的值可以被命令优先级机制控制。

(b) **优先值数组**属性：这个属性是一个只读数组，其中包含按优先级别递减排列的具有优先级的命令或者空。具有非空值的最高优先级（最低的数组索引）是活动命令。

(c) **释放缺省**属性：这个属性与**可命令**属性具有相同的数据类型（和工程单位）。当**优先值数组**属性的所有表目都为空值时，**可命令**属性的值取**释放缺省**属性规定的值。

虽然可以使用**命令**对象向一组对象属性写入值，但是只能对可命令的属性写入命令优先级。

19.1 优先级机制

对于 BACnet 对象，命令的优先级依据分配给发送命令的实体的优先值确定。一个区分先后顺序的命令是一个直接对一个对象的一个可命令属性进行作用的命令，使用**写属性**服务请求或者**写多个属性**服务请求执行这个命令。**写属性**服务请求或者**写多个属性**服务请求原语包含有一个条件的‘**优先级**’参数，取值为 1－16。一个对象的每个可命令属性都有一个相关的优先级表，其中表示**优先值数组**属性。**优先值数组**属性包含有一个按递减优先级顺序排列的命令值的数组。这个数组的第一个值对应优先级 1（最高级），第二个值对应优先级 2，依此类推，直到第十六个值对应优先级 16（最低级）。

在**优先值数组**属性的表目中，可以是一个命令值，也可以是一个空值。空值表示不存在这个优先值的命令。对象顺序检查优先级表中的所有表目，搜寻出具有最高优先级的非空值，设置可命令属性为这个值。

一个发布命令的实体（应用程序，操作员，等等）可以发送一个命令对一个对象的可命令属性进行写操作，或者指示放弃以前发送的命令。使用类似于命令本身的写操作执行放弃命令操作。在可命令属性为空值的情况下，放弃命令操作将空值放置在**优先值数组**属性中对应于合适的优先值的位置。这个优先级机制应用于改变可命令属性值的本地操作，也应用于通过 BACnet 服务进行的写操作。

如果要对一个可命令属性进行没有明确规定优先值的写操作，则假定这个操作使用缺省的 16 级优先级（最低优先级）。如果要对一个不可命令的属性进行有明确规定优先值的写操作，则忽略优先级。优先值数组属性是只读属性，只能通过对这个可命令属性自身进行写操作才能间接改变这个属性的值。

19.1.1 可命令属性（Commandable Properties）

优先级机制只能应用于某些对象的某些属性。本标准中规定的可命令属性和对象如下：

对　　象	可命令属性
模拟输出	当前值
二进制输出	当前值
多态输出	当前值
模拟值	当前值
二进制值	当前值

这些对象的指定属性定义为标准的可命令属性，另外生产商也可以决定对任何一个生产商定义的属性应用优先级机制。这些附加的可命令属性也以适当的名字具有相应的**优先值数组**属性和**释放缺省**属性。参见 23.3 节。

19.1.2 优先级命令（Prioritized Command）

所谓优先级命令是指直接作用于可命令属性的命令，这个命令是**写属性**服务请求或者**写多个属性**服务请求中的一个。在每一种情况下，请求原语都包含下列参数：

属性标识符：　　**可命令属性**

属性值：　　**希望的值**

优先级：　　**优先级**

一个成功写操作的最后结果是将一个希望的值放置在优先级表中适当优先值的位置。如果在这个优先值位置处已经存在一个另外的值，则新值将覆盖旧值，不论是否与先前的命令实体相符。

19.1.3 释放命令（Relinquish Commands）

如果一个命令实体不再希望控制一个可命令属性，则发送一个释放命令。释放命令也是释放命令也是**写属性**服务请求或者**写多个属性**服务请求中的一个。在每一种情况下，请求原语都包含下列参数：

属性标识符：　　**可命令属性**

属性值：　　**空值**

优先级：　　**优先级**

优先级表中的元素的值为空值表示在这个优先值处不存在任何命令。如果优先级表中的所有元素都是空值，则可命令属性认为取对象的**释放缺省**属性中定义的值。

存在着一个应用实体释放不是它自己的优先级的可能性，这将导致无法预料的结果。如果有多个应用被分配为相同的优先级级别，则有可能一个应用实体释放了来自其它应用实体的命令，从而导致无法预料的操作。为了减少这种可能性，在系统内不允多个命令实体具有相同的优先级级别就格外重要。

19.1.4 命令源标识符（Command Source ID）

本标准没有规定把维护命令源标识符作为优先级表的一部分。对命令源标识符的任何实现都具有生产商专用特征。

19.1.5 命令改写（Command Overwrite）

只要向可命令属性发送一个命令，就在优先值数组中的适当优先值位置放置了一个值，不论该位置当前是什么，新的命令将改写原来的命令值，而且这种改写操作并不导致向原命令实体发送任何通知。

19.2 应用优先级的分配（Application Priority Assignments）

命令实体被分配 16 个可能的优先级级别中的一个。多数优先级的分配与站点有关，表示站点管理的客观需要。表 19-1 列出了标准优先级的级别。另外一些需要分配优先级的应用有温度超值（Temperature Override）、查询限制（Demand Limiting）、最佳启/停、任务循环（Duty Cycling）和时间表。这些应用的相关优先级在不同的站点有不同的等级，没有一个标准。为了在任何站点都可以互相操作，只需所有的设备都按照相同的优先级方案实现相应设置。表中标记为**可利用（Available）**的位置的优先级级别可以分配给 DDC、EMS 等程序。至于什么条件构成**手动操作的人身安全**或者**自动操作的人生安全**的判别准则，由生产商自行确定。

表 19-1. 标准的命令优先级级别

优先级级别 PRIORITY LEVEL	应　用 APPLICATION
1	手动操作的人身安全 Manual-Life Safety
2	自动操作的人身安全 Automatic-Life Safety
3	可利用 Available
4	可利用 Available
5	关键仪器设备的控制 Critical Equipment Control
6	活动/非活动最小值 Minimum On/Off
7	可利用 Available
8	人工操作者 Manual Operator
9	可利用 Available
10	可利用 Available
11	可利用 Available
12	可利用 Available
13	可利用 Available
14	可利用 Available
15	可利用 Available
16	可利用 Available

19.3 活动最小值和非活动最小值（Minimum-On-Time and Minimum-Off-Time）

如果可命令属性是**二进制输出**对象或者**二进制值**对象的**当前值**属性，且对象具有可选的**活动最小值**和**非活动最小值**属性，则最小活动时间和最小非活动时间将遵循下列算法的规律。命令优先级的级别 6 专门为这种算法保留，任何对象都不能用作它用。

(a) 如果**当前值**是 ACTIVE，最近一次**当前值**的改变到现在的时间小于**活动最小值**，则**优先级数组**的元素 6 包含的值为 ACTIVE。

(b) 如果**当前值**是 INACTIVE，最近一次**当前值**的改变到现在的时间小于**非活动最小值**，则**优先级数组**的元素 6 包含的值为 INACTIVE。

(c) 如果(a)、(b)都不成立，则**优先级数组**的元素 6 包含的值为 NULL。

这些规则规定了根据写入的**当前值**和活动时间的状态应该执行的操作。如何实现这些操作由生产商自行确定，只要求所产生的效果符合这些规则。

在向任何优先级的可命令属性进行写入赋值时，不论**活动最小值**或者**非活动最小值**如何，总是直接向优先级表中的适当位置写入指定值或者空值。

本地优先级维护实体检验**优先级数组**，确定具有非零值的最高优先级。如果这个值与写入操作前的**当前值**不同，则发生状态改变。如果这样的状态改变发生，则将新值写入**优先级数组**的优先级 6 的位置，且记录改变的时间。如何进行计时操作由生产商自行确定。

如果由优先级级别 6 中的非零值表示的最小活动时间或者最小非活动时间已经到达，则本地最小时间维护实体向优先级级别 6 写入空值，并重新检查**优先级数组**，确定新的**当前值**。如果新的**当前值**指示状态改变，则按照上述的方法进行适当的操作。

如果优先级级别 6 的非空值的作用是对任何低优先级（大的优先级号码）进行写操作，这些操作都不导致状态的改变。这样，就可以达到对这些优先级相关的最小活动时间或者最小非活动时间的保护。

对任何一个高于级别 6（更小的优先值号码）的优先级进行写操作，都将导致状态改变，不论**活动最小值**或者**非活动最小值**如何。因此，这些优先级都只作为紧急应用。还必须注意到，向这些高级别的优先级进行写操作所导致的状态改变，又将导致向级别 6 进行写操作，并且产生上述的其它操作。因此，如果一个空值接着写入高优先级中，而最小时间又处于有效时间内，则可以观察到在较低优先级的值改变所引起的状态改变之前的这段时间。

对于最小活动时间和最小非活动时间处理的进一步的讨论参见附录 I 。

19.4 命令对象的优先级机制（Prioritization for Command Objects）

一个**命令**对象能够像其它命令发布实体一样发布命令。可以将一个**命令**对象与一个具有任何优先级的应用相关联。**命令**对象的行为属性包含所有的向可命令属性进行写操作所必需的参数。

19.5 环对象的优先级机制（Prioritization for Loop Objects）

一般来说，**环**对象不使用 BACnet 服务进行命令方面的操作，但是还是有可能需要与具有可命令属性的对象相互作用。为此，每个**环**对象都有一个**写入优先级**属性，它指定了与可命令属性相关的控制环的优先级。参见 12.13.28 节。

19.6 时间表对象的优先级机制（Prioritization for Schedule Objects）

一般来说，**时间表**对象不使用 BACnet 服务进行命令方面的操作，但是还是有可能需要与具有可命令属性的对象相互作用。为此，每个**时间表**对象都有一个**写入优先级**属性，它指定了与可命令属性相关的控制环的优先级。参见 12.13.10 节。

20. BACnet 协议数据单元编码

在 BACnet 中，使用**应用层协议数据单元（APDU）**传送包含在应用服务原语和相关参数中的信息。

ISO 标准 8824 是关于**抽象句法结构符号 1（Abstract Syntax Notation One, ASN.1）**的规范说明，这个规范被选定作为表述 BACnet 服务中的数据内容的方法。第 21 节给出本标准中定义的所有服务的 ASN.1 定义。ASN.1 提供了一个抽象的句法结构。然而，一个 APDU 可以有多种比特格式，这些取决于所采用的编码规则。

在 OSI 模式中，由表示层使用一个协商程序选定所用的编码规则。协同工作的系统使用这个协商程序，不仅确定诸如 ISO 8825、ASN.1 的**基本编码规则**规范说明这样的基本编码规

则，而且确定 APDU 是否要遵守的诸如数据压缩、加密、字符代码转换这样的处理规则。

因为 BACnet 采用简化的 OSI 体系结构，不包含任何表示层的功能，必须预先定义好通信设备一致遵守的编码规则。BACnet 的编码规则设计考虑到了楼宇自动控制体系对简单化、紧凑化的要求。因此，这个编码规则在一些方面与 ISO 8825 有所不同。但是仍然允许使用 ASN.1 对 BACnet 的 APDU 进行编码。这意味着在将来的全 OSI 网络中，应用实体仍然可以通过增加表示层的功能而使用 BACnet 服务及其程序，这些表示层的功能是协调本标准的编码规则与以后可能使用的编码规则的一致性。

在 ISO 8825 中规范的 ASN.1 编码对 PDU 中的所有数据元素都适用。每个数据元素由三个部分组成：1. 标识符字节，2. 长度字节，3. 内容字节。由于每个数据元素都有明确的标识，这就能够开发出解析器，在不用已知 PDU 的格式或者语义内容的情况下，对任何 PDU 进行解码。另外一种替代的方法是*隐含地***标识数据元素**，这要求共同遵守统一的数据格式和元素在 PDU 中的位置。前一种方法一般会有更大的开销，而后一种方法节省开销，但是限制了将来的扩展性。

在 BACnet 中采用的是一种折中的方法。每个 APDU 分为固定部分和可变部分。固定部分包含协议控制信息，采用隐含式的编码方法，在 20.1 节详细规范可变部分包含具体的服务信息，采用明确表示的编码方式，在 20.2 节中详细规范。这种方案显著地减少了开支而又保持了将来较容易地增加新服务的特征。

20.1 BACnet APDU 的固定部分的编码

BACnet APDU 由**协议控制信息**和**用户数据**所组成。

“协议控制信息”（PCI）包含进行应用层协议操作所需的数据，这些数据包括 APDU 的类型、匹配服务请求和服务响应的信息以及重组分段报文的信息。这些信息包含在 APDU 的“头部”，也称之为固定部分。

“用户数据”包含单个服务请求和服务响应的具体信息，APDU 的这一部分称之为‘可变部分’。

因为每个 BACnet 的 APDU 都具有 PCI 域，尽管在 ASN.1 中规定可以使用标记（tag）进行 APDU 的句法结构描述，但是在 BACnet 中，对 PCI 编码中不使用标记或者长度信息。而对内容可变的用户数据进行编码时，则使用标记，20.2 节详细规范。有选择地使用标记可以大大减少开销。本节后面的内容对 APDU 类型的格式进行详细说明。

20.1.1 BACnet PDU 选择标记的编码

所有的被 ANS.1 定义的 BACnet 报文称之为 BACnetPDU。参见 21 节。在 BACnet 中，一共有 **8 种** BACnet APDU 的 ANS.1 编码类型，每一种类型就形成一个 BACnetPDU。对所有的 BACnet APDU，头部的第一个字节（8 位）中的第 4－7 位被编码成为一个 4 比特的二进制数，其中第 7 位是高位，表示 BACnet APDU 的类型。这 4 个比特编码表示标记的值（0－7），代表 APDU 的类型选择。下面给出每一种 APDU 类型的编码例程。

20.1.2 BACnet 有证实请求 PDU

BACnet 有证实请求 PDU 用于传送包含在有证实服务请求原语中的信息。

```
BACnet-Confirmed-Request-PDU ::=SEQUENCE{
pdu-type                    [0] Unsigned (0..15),--0 for this PDU type
segmented message           [1] BOOLEAN【报文分段】
more follows                [2] BOOLEAN【后继】
segmented-response-accepted [3] BOOLEAN【收到的分段响应】
reserved                    [4] Unsigned (0..31),--must be set to zero【】
max-APDU-lenth-accepted     [5] Unsigned (0..15)    【接收的最大 APDU 长度】
invokeID                    [6] Unsigned (0..255)   【调用 ID】
sequence number             [7] Unsigned (0..255) OPTIONAL,only if segmented msg
                                                    【序列号】
proposed-window-size        [8] Unsigned (0..127) OPTIONAL,only if segmented msg
                                                    【预设窗口尺寸】
service-choice              [9]BACnetConfirmedServiceChoice 【服务选择】
service-request             [10]BACnet-Confirmed-Service-Request【服务请求】
-- 在头部编码中不使用上下文特定标记 0..10
}
```

BACnet 有证实请求 PDU 的参数意义如下所述。

20.1.2.1 报文分段（segmented-message）参数

这个参数包含在当前 PDU 中，指明有证实服务请求是完整的还是仅是一部分。如果当前 PDU 是一个整体，则‘segmented-message’参数值应为 FALSE。如果当前 PDU 仅包含了请求的一段，则该参数值应为 TRUE

20.1.2.2 后继（more-follows）参数

只有‘segmented-message’参数值为 TRUE 时，‘more-follows’参数才有意义。如果‘segmented-message’为 TRUE，则除了最后一段之外的所有组成有证实服务请求的段的‘more-follows’参数值都是 TRUE ，最后一段的‘more-follows’的参数值为 FALSE。如果‘segmented-message’参数值为 FALSE ，则‘more-follows’参数被编码器置为 FALSE，而解码器忽略这个参数。

20.1.2.3 收到的分段响应（segmented-response-accepted）参数

如果发送有证实请求的设备要求收到分段的复杂确认作为响应，则应该将这个参数值设置为 TRUE。否则将其设置为 FALSE。这个参数包含在有证实请求中以便响应设备可以确定如何传送响应。

20.1.2.4 接收的最大 APDU 长度（max-APDU-length-accepted）参数

这个参数说明发送设备**能接收**的单个 APDU 的最大尺寸。这个参数包含在有证实请求中以便响应设备可以确定如何传送响应。这个参数的编码如下：

注：octets 表示 八位位组,八位字节(=byte)

B‘0000’ Up to MinimumMessageSize (50 octets)

B‘0001’ Up to 128 octets

B‘0010’ Up to 206 octets(fits in a LonTalk frame)

B‘0011’ Up to 480 octets(fits in an ARCNET frame)

B‘0100’ Up to 1024 octets

B‘0101’ Up to 1476 octets(fits in a ISO 8802-3 frame)

B‘0110’ reserved by ASHRAE【保留的】

B‘0111’ reserved by ASHRAE

B‘1000’ reserved by ASHRAE

B‘1001’ reserved by ASHRAE

B‘1010’ reserved by ASHRAE

B‘1011’ reserved by ASHRAE

B‘1100’ reserved by ASHRAE

B‘1101’ reserved by ASHRAE

B‘1110’ reserved by ASHRAE

B‘1111’ reserved by ASHRAE

20.1.2.5 调用 ID（invokeID）参数

这个参数是一个整数，取值范围是 0-255，由服务请求者分配。这个参数用于把对于有证实服务请求的响应和原始请求联系起来。在没有错误发生的情况下，服务提供者使用一个 **BACnet 简单确认** PDU 或者一个 **BACnet 复杂确认** PDU 返回‘**invokeID**’参数。在有错误发生的情况下，服务提供者使用一个 **BACnet 错误** PDU、一个 **BACnet 拒绝** PDU 或者一个 **BACnet 中止** PDU 返回‘**invokeID**’参数。

发送服务请求的设备产生参数‘**invokeID**’。此参数在所有的由该设备产生的还没有被确认的 APDU 中是唯一的。一个分段服务请求的所有分段都有一个相同的‘**invokeID**’。一旦一个‘**invokeID**’分配给了一个 APDU，则这个‘**invokeID**’将保留在设备中，直到收到一个具有相同的‘**invokeID**’的响应 APDU，或者设备的计时器到时仍没有收到任何响应。不论是收到了正确响应，还是计时器到时，都要将‘**invokeID**’释放，以便重新分配。如何维护当前没有被使用的‘**invokeID**’值，以及如何挑选一个‘**invokeID**’值分配给新的 APDU，由生产商自行确定。如何在请求设备和响应设备中维护正在使用的‘**invokeID**’值也由生产商自行确定。请求设备可以给所有的有证实 APDU 分配一个‘**invokeID**’空间，也可以给每一个目标设备地址都分配一个‘**invokeID**’空间。因为‘**invokeID**’值只是源设备唯一，所以响应设备只将请求设备地址和‘**invokeID**’值保留到发送响应为止。在发送了响应之后，响应设备就可以丢弃‘**invokeID**’信息。

20.1.2.6 序列号（sequence-number）参数

如果‘segmented-message’参数为 TRUE，则存在可选参数‘sequence-number’，这个参数是无符号整型类型，取值范围是 0－256，连续递增变化。这个参数标示分段请求的每一个分段，响应设备根据收到的‘sequence-number’值确定这个分段是否已收到。分段请求的第一分段的‘sequence-number’值为 0。

20.1.2.7 预设窗口尺寸（proposed-window-size）参数

如果‘segmented-message’参数为 TRUE，则存在可选参数‘proposed-window-size’，这个参数是无符号二进制整数，取值范围是 1－127，指定包含‘invokeID’的报文分段的最大数目，这个值是发送设备在等待分段确认 PDU 之前能够发送的最大报文分段的数目。(参见 5.2 节和 5.3 节)。

20.1.2.8 服务选择（service-choice）参数

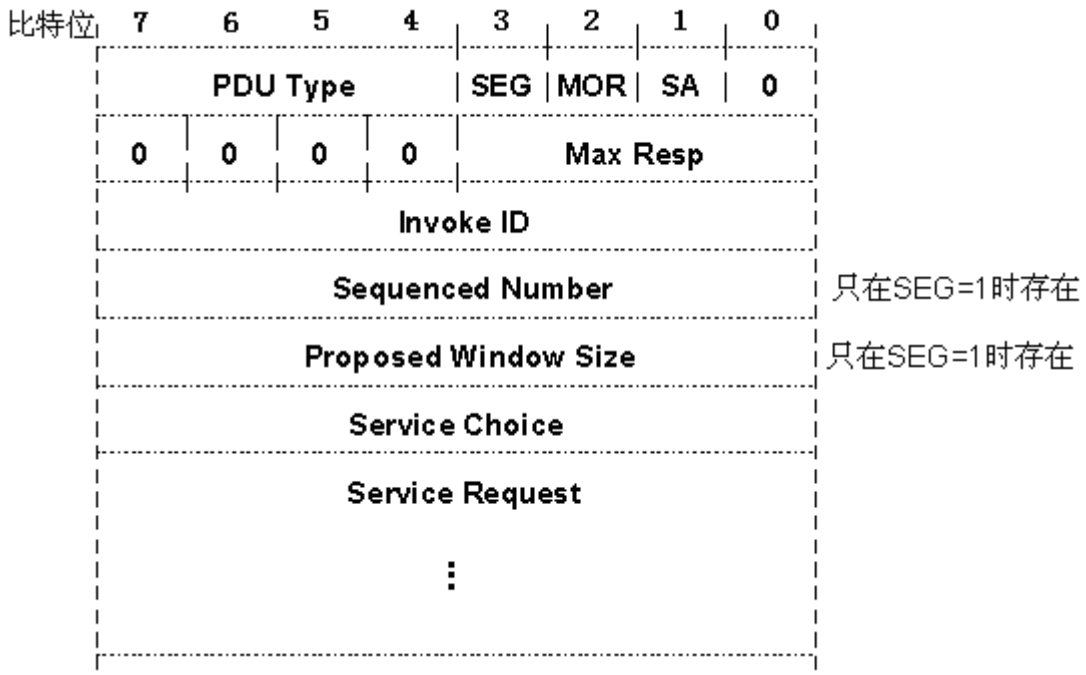
这个参数包含 BACnet 有证实服务选择（BACnetConfirmedServiceChoice）的值。参见 21 节。

20.1.2.9 服务请求（service-request）参数

这个参数包含被请求的特定服务的参数，它的编码是根据 20.2 节中的规则进行的。这些参数在本标准的每个服务规范中定义，在 21 节，使用 ASN.1 规则对它们进行了详细规定。

20.1.2.10 BACnet 有证实请求 PDU 的格式

BACnet-Confirmed-Request-PDU 的格式如下：



PDU 域的值如下：

PDU Type =0 (BACnet-Confirmed-Request-PDU)

SEG =0 (Unsegmented Request)

=1 (segmented Request)

MOR =0 (No More Segments Follow)

	=1 (No More Segments Follow)
SA	=0 (Segmented Response not accepted)
	=1 (Segmented Response accepted)
Max Resp	=(0..5)(Size of Maximum APDU accepted)
Invoke ID	=(0..255)
Sequence Number	=(0..255) 只有当 SEG=1 时才存在
Proposed Window Size	=(0..127) 只有当 SEG=1 时才存在
Service Choice	=BACnetConfirmedServiceChoice
Service Request	=Variable Encoding per 20.2

上面列表中的‘0’位设置为零值。这些位目前不用，保留为 ASHRAE 所用。

20.1.3 BACnet 无证实请求 PDU

BACnet 无证实请求 PDU 用于传送包含在无证实服务请求原语中的信息。

```
BACnet- Unconfirmed-Request-PDU ::＝SEQUENCE{
pdu-type                [0] Unsigned (0..15), --1 for this PDU type
reserved                [1] Unsigned (0..15),--must be set to zero
service-choice          [2] BACnetUnConfirmedServiceChoice     【服务选择】
service-request         [3] BACnet-UnConfirmed-Service-Request【服务请求】
--Context specific tags 0..3 are NOT used in header encoding
}
```

BACnet 无证实请求 PDU 的参数的意义如下所述。

20.1.3.1 服务选择（service-choice）参数

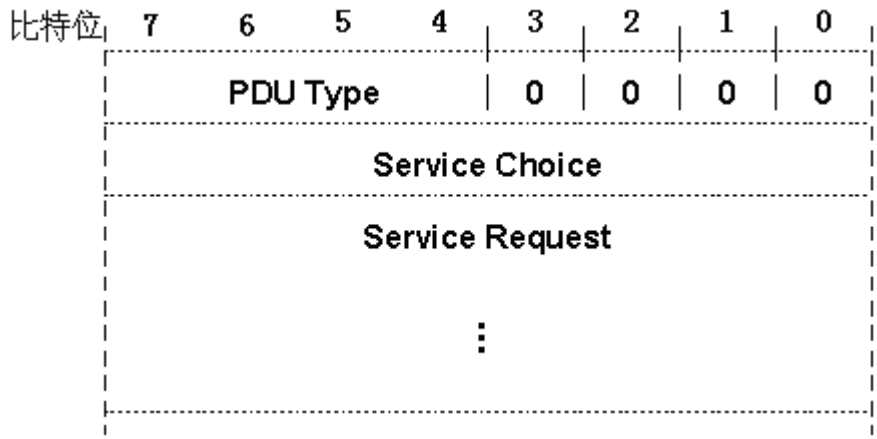
这个参数包含 BACnet 无证实服务选择的值。参见 21 节。

20.1.3.2 服务请求（service-request）参数

这个参数包含被请求的特定服务的参数，它的编码是根据 20.2 节中的规则进行的。这些参数在本标准的每个服务规范中定义，在 21 节，使用 ASN.1 规则对它们进行了详细规定。

20.1.3.3 BACnet 无证实请求 PDU 的格式

BACnet-Unconfirmed-Request-PDU 的格式如下：



PDU 域的值如下：

PDU Type =1(BACnet-UNConfirmed-Request-PDU)

Service Choice = BACnetConfirmedServiceChoice

Service Request = Variable Encoding per 20.2

上面列表中的‘0’位设置为零值。这些位目前不用，保留为 ASHRAE 所用。

20.1.4 BACnet 简单确认 PDU

BACnet 简单确认 PDU 用于传送包含在一个服务响应原语（‘Result(+)’）中的信息，这个信息是服务请求已经成功执行。

```
BACnet-SimpleACK-PDU ::=SEQUENCE{
pdu-type                [0] Unsigned (0..15), --2 for this PDU type
reserved                [1] Unsigned (0..15),--must be set to zero
original-invokeID       [2] Unsigned (0..225)          【初始调用 ID】
service-ACK-choice      [3] BACnetConfirmedServiceRequest【服务确认选择】
--Context specific tags 0..3 are NOT used in header encoding
}
```

BACnet 简单确认 PDU 的参数意义如下所述。

20.1.4.1 初始调用 ID（original-invokeID）参数

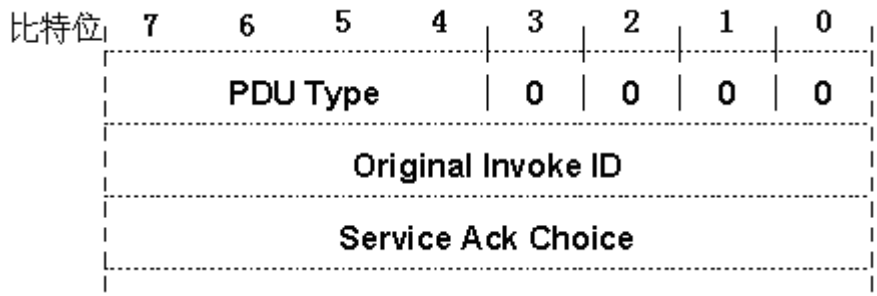
这个参数是包含在正在被确认的有证实服务请求中的‘invokeID’。

20.1.4.2 服务确认选择（service-ACK-choice）参数

这个参数包含 BACnet 有证实服务选择的值，这里的 BACnet 有证实服务选择是对包含在先前的、导致这个确认的 BACnet 有证实服务请求中的服务而言的。

20.1.4.3 BACnet 简单确认 PDU 的格式

BACnet-SimlieACK-PDU 的格式如下：



这个 APDU 总是具有三个字节。

PDU 域的值如下：

PDU Type =2(BACnet-SimpleACK-PDU)

Original Invoke ID =(0 .. 255)

Service ACK Choice = BACnetConfirmedService

上面列表中的‘0’位设置为零值。这些位目前不用，保留为 ASHRAE 所用。

20.1.5 BACnet 复杂确认 PDU

BACnet 复杂确认 PDU 用于传送包含在一个服务响应原语（‘**Result(+)**’）中的信息，这个信息除了包含服务请求已经成功执行之外，还有其它一些信息。

```
BACnet-ComplexACK-PDU ::=SEQUENCE{
pdu-type              [0] Unsigned (0..15),--3 for this PDU type
segmented message     [1] BOOLEAN【报文分段】
more follows          [2] BOOLEAN【后继】
reserved              [3] Unsigned (0..3),--must be set to zero
original-invokeID     [4] Unsigned (0..225)【初始的调用 ID】
sequence number       [5] Unsigned (0..255) OPTIONAL,--only if segment【序列号】
proposed-window-size  [6] Unsigned (0..127) OPTIONAL, --only if segment
                                                    【预设窗口尺寸】
service-ACK-choice    [7]BACnetConfirmedServiceChoice,【服务确认选择】
service-ACK           [8]BACnet -Confirmed-Service-ACK【服务确认】
--在头部编码中不使用上下文特定标记 0..8
}
```

BACnet 复杂确认 PDU 的参数的意义如下所述。

20.1.5.1 报文分段（segmented-message）参数

这个参数包含在当前 PDU 中，指明有证实服务响应是完整的还是仅是一部分。如果当前的响应 PDU 是一个整体，则‘**segmented-message**’参数值应为 FALSE。如果当前 PDU 仅包含了响应的一段，则该参数值应为 TRUE

20.1.5.2 后继（more-follows）参数

只有‘**segmented-message**’参数值为 TRUE 时，‘**more-follows**’参数才有意义。如果‘**segmented-message**’为 TRUE，则除了最后一段之外的所有组成有证实服务响应的段的‘**more-follows**’参数值都是 TRUE ，最后一段的‘**more-follows**’的参数值为 FALSE。如果‘**segmented-message**’参数值为 FALSE ，则‘**more-follows**’参数被编码器置为 FALSE，而解码器忽略这个参数。

20.1.5.3 初始的调用 ID（original-invokeID）参数

这个参数是包含在正在被确认的有证实服务请求中的‘invokeID’参数。一个分段确认的所有分段都使用一个相同的‘original-invokeID’参数。

20.1.5.4 序列号（sequence-number）参数

如果‘segmented-message’参数为 TRUE，则存在可选参数‘sequence-number’，这个参数是无符号整型类型，取值范围是 0－256，连续递增变化，模 256。这个参数标示分段响应的每一个分段，请求设备根据收到的‘sequence-number’值确定这个分段是否已收到。分段响应的第一分段的‘sequence-number’值为 0。

20.1.5.5 预设窗口尺寸（proposed-window-size）参数

如果‘segmented-message’参数为 TRUE，则存在可选参数‘proposed-window-size’，这个参数是无符号二进制整数，取值范围是 1－127，指定包含‘original-invokeID’的报文分段的最大数目，这个值是发送设备在等待分段确认 PDU 之前能够发送的最大报文分段的数目。（参见 5.2 节和 5.3 节）。

20.1.5.6 服务确认选择（service-ACK-choice）参数

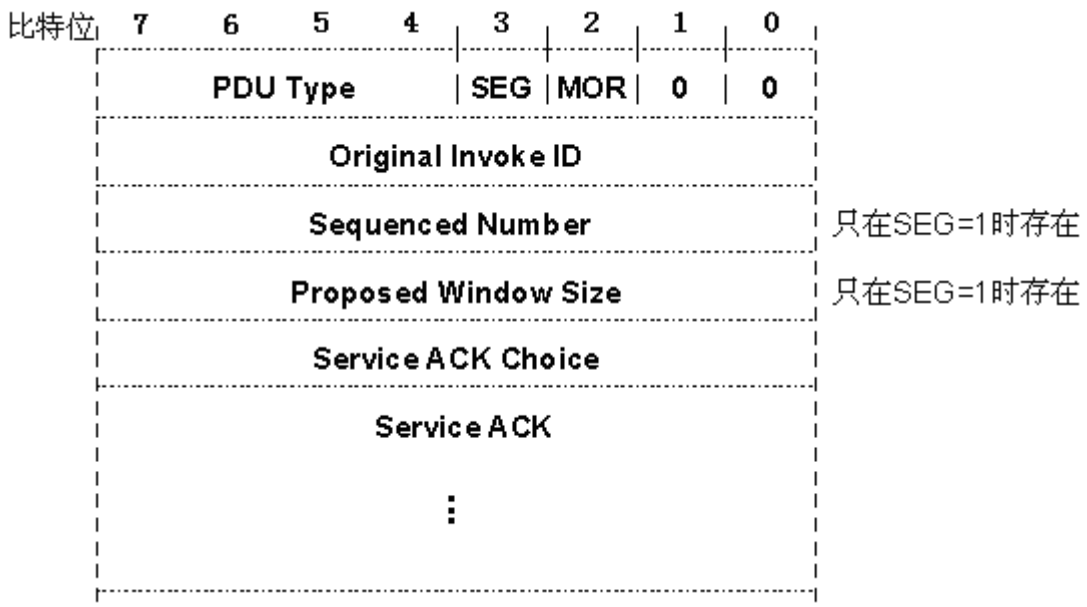
这个参数包含 BACnet 有证实服务选择的值，这里的 BACne 有证实服务选择是对包含在先前的、导致这个确认的 BACnet 有证实服务请求中的服务而言的。

20.1.5.7 服务确认（service-ACK）参数

这个参数包含特定服务确认的参数，它的编码是根据 20.2 节中的规则进行的。这些参数在本标准的每个服务规范中定义，在 21 节，使用 ASN.1 规则对它们进行了详细规定。

20.1.5.8 BACnet 复杂确认 PDU 的格式

BACnet-ComplexACK-PDU 的格式如下：



PDU 域的值如下：

PDU Type =3 (BACnet-ComplexACK-PDU)

SEG =0 (Unsegmented Request)

=1 (segmented Request)

MOR =0 (No More Segments Follow)

=1 (More Segments Follow)

Original Invoke ID = (0..255)

Sequence Number = (0..255) 只有当 SEG=1 时才存在

Proposed Window Size = (0..127) 只有当 SEG=1 时才存在

Service ACK Choice = BACnetConfirmedServiceChoice

Service ACK = Variable Encoding per 20.2

上面列表中的‘0’位设置为零值。这些位目前不用，保留为 ASHRAE 所用。

20.1.6 BACnet 分段确认 PDU

BACnet 分段确认 PDU 用于对收到一个或者多个 PDU 进行确认，这些 PDU 包含一个分段报文的分段。**BACnet 分段确认 PDU** 也用于对分段报文的下一个或者几个分段的请求。

BACnet-SegmentACK-PDU ::＝SEQUENCE{

pdu-type [0] Unsigned (0..15), --4 for this PDU type

reserved [1] Unsigned (0..15),--must be set to zero

negative-ACK [2] BOOLEAN**【否定确认】**

server [3] BOOLEAN**【服务器】**

original-invokeID [4] Unsigned (0..225)**【初始的调用 ID】**

```
sequence number              [5] Unsigned (0..255)【序列号】
actual-window-size           [6] Unsigned (0..127)【实际窗口尺寸】
-- 在头部编码中不使用上下文特定标记 0..6
}
```

BACnet 分段确认 PDU 的参数的意义如下所述。

20.1.6.1 否定确认（negative-ACK）参数

如果正在被发送的 BACnet 分段确认 PDU 指示收到的分段乱序，则这个参数为 TURE，否则为 FALSE。

20.1.6.2 服务器（server）参数

如果 BACnet 分段确认 PDU 是一个服务器所发送的，即 BACnet 分段确认 PDU 是对一个有证实请求 PDU 的一个或者几个分段的确认，则这个参数为 TURE。

如果 BACnet 分段确认 PDU 是一个客户所发送的，即 BACnet 分段确认 PDU 是对一个复杂确认 PDU 的一个或者几个分段的确认，则这个参数为 FALSE。

20.1.6.3 初始的调用 ID（original-invokeID）参数

这个参数是包含在正在被确认的分段中的‘invokeID’参数。

20.1.6.4 序列号（sequence-number）参数

这个参数是先前收到的报文分段的‘sequence-number’参数。这个参数用于对本参数标明的报文分段以及这个分段之前的所有分段的正确接收的确认。

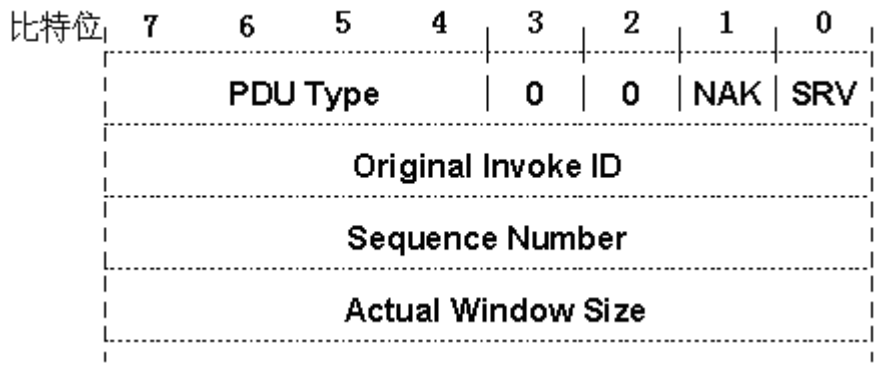
如果接收的报文分段的‘more-follows’参数为 TRUE，则‘sequence-number’参数也表示请求接收其‘sequence-number’参数值为(1+本参数值，模 256)开始的后续报文分段。

20.1.6.5 实际窗口尺寸（actual-window-size）参数

这个参数是无符号二进制整数，取值范围是 1－127，指定包含‘original-invokeID’的报文分段的最大数目,这个值是发送设备在发送另一个分段确认 PDU 之前能够接收的最大报文分段的数目。(参见 5.3 节)。

20.1.6.6 BACnet 分段确认 PDU 的格式

BACnet-SegmentACK-PDU 的格式如下：



这个 APDU 总是具有四个字节。

PDU 域的值如下：

PDU Type =4 (BACnet-segmentACK-PDU)

NAK =0 (Normal Acknowledgement)

=1 (Negative Acknowledgement, Segment Out Order)

SRV =0 (Sent by Client)

=1(Sent by Server)

Original Invoke ID = (0..255)

Sequence Number = (0..255)

Actual Window Size = (0..127)

上面列表中的‘0’位设置为零值。这些位目前不用，保留为 ASHRAE 所用。

20.1.7 BACnet 差错 PDU

BACnet 差错 PDU 用于传送包含在一个服务响应原语（‘Result(+)’）中的信息，这个信息指出前一个服务请求完全失败的原因。

```
BACnet-Error-PDU ::＝SEQUENCE{
pdu-type                [0] Unsigned (0..15), --5 for this PDU type
reserved                [1] Unsigned (0..15), --must be set to zero
original-invokeID       [2] Unsigned (0..225),
error-choice                [3]BACnetConfirmedServiceChoice
error                       [4] BACnet-Error
-- 在头部编码中不使用上下文特定标记 0..4
}
```

BACnet 差错 PDU 的参数的意义如下所述。

20.1.7.1 初始调用 ID（original- invokeID）参数

这个参数是包含在这个 PDU 进行差错响应的有证实服务请求中的‘invokeID’。

20.1.7.2 差错选择（error-choice）参数

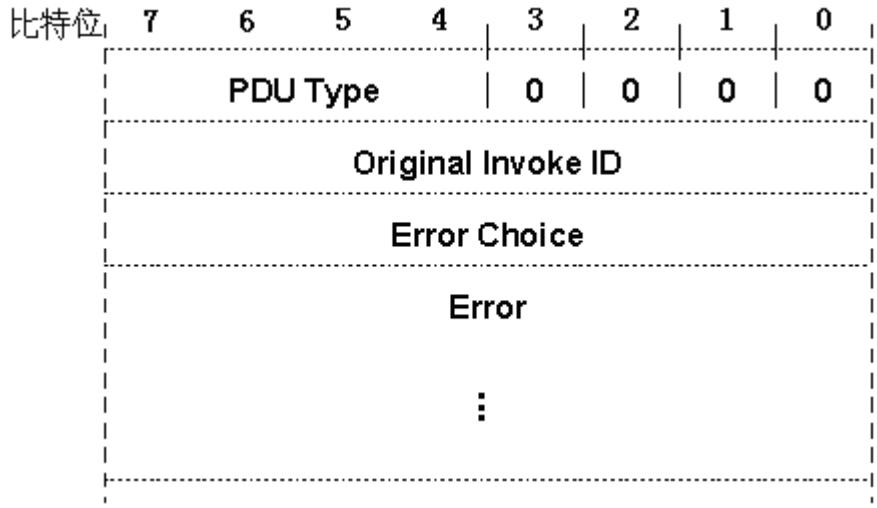
这个参数是 BACnet 有证实服务选择（BACnetConfirmedServiceChoice）类型，包含 BACnet 差错（BACnet-Error）选择的标记值。参见 21 节。

20.1.7.3 差错（error）参数

这个参数是 BACnet 差错（BACnet-Error）类型，指出指定的服务请求不能被执行的原因。此参数采用 20.2 中的规则进行编码。

20.1.7.4 BACnet 差错 PDU 的格式

BACnet-Error-PDU 的格式如下：



PDU 域的值如下：

PDU Type =5 (BACnet-Error-PDU)
Original Invoke ID = (0..255)
Error-choice = BACnetConfirmedServiceChoice
Error = Variable Encoding per 20.2

上面列表中的‘0’位设置为零值。这些位目前不用，保留为 ASHRAE 所用。

20.1.8 BACnet 拒绝 PDU

BACnet 拒绝 PDU 用于对一个有证实请求 PDU 的拒绝接收，其原因是这个被拒绝的 PDU 具有句法结构错误或者其它的协议错误，使得不能对这个 PDU 进行解读，或者不能够提供请

求的服务。只能对有证实请求 PDU 进行拒绝（参见 18.8 节）。

```
BACnet-Reject-PDU ::＝SEQUENCE{
pdu-type              [0] Unsigned (0..15), --6 for this PDU type
reserved              [1] Unsigned (0..15), --must be set to zero
original- invokeID    [2] Unsigned (0..225),
reject reason         [3]BACnetRejectReason
-- 在头部编码中不使用上下文特定标记 0..3
}
```

BACnet 拒绝 PDU 的参数的意义如下所述。

20.1.8.1 初始调用 ID（original- invokeID）参数

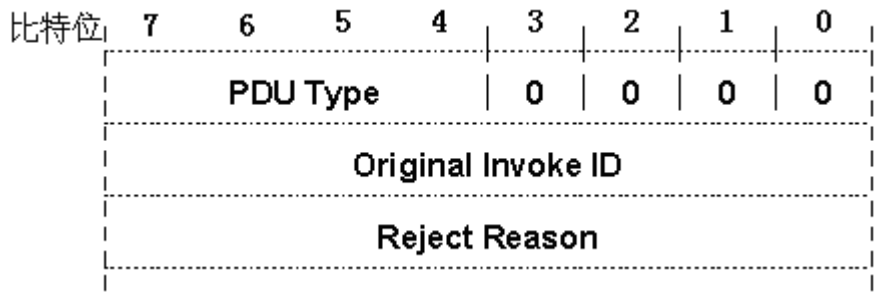
这个参数是正在被拒绝的 PDU 的‘**invokeID**’。

20.1.8.2 拒绝原因（reject-reason）参数

这个参数是 **BACnet 拒绝原因（BACnetRejectReason）**类型，包含正在被拒绝的具有指定的‘**invokeID**’的 PDU 被拒绝的原因。

20.1.8.3 BACnet 拒绝 PDU 的格式

BACnet-Reject-PDU 的格式如下：



这个 PDU 总是具有三个字节。

PDU 域的值如下：

```
PDU Type          =6 (BACnet-Reject-PDU)
Original Invoke ID   =(0..255)
Reject Reason       =One octet containing the reject reason enumeration
```

上面列表中的‘0’位设置为零值。这些位目前不用，保留为 ASHRAE 所用。

20.1.9 BACnet 中止 PDU

BACnet 中止 PDU 用于结束两个对等实体之间的事务处理。

```
BACnet-Abort-PDU:: ＝SEQUENCE{
pdu-type                    [0] Unsigned (0..15), --7 for this PDU type
reserved                    [1] Unsigned (0..7), --must be set to zero
server                      [2] BOOLEAN,
original-invokeID           [3] Unsigned (0..225),
abort reason                [4] BACnetAbortReason
-- 在头部编码中不使用上下文特定标记 0..4
}
```

BACnet 中止 PDU 的参数的意义如下所述。

20.1.9.1 服务器（server）参数

如果 BACnet 中止 PDU 是服务器发送的，则这个参数为 TURE。如果 BACnet 中止 PDU 是客户发送的，这个参数为 FALSE。

20.1.9.2 初始的调用 ID（original-invokeID）参数

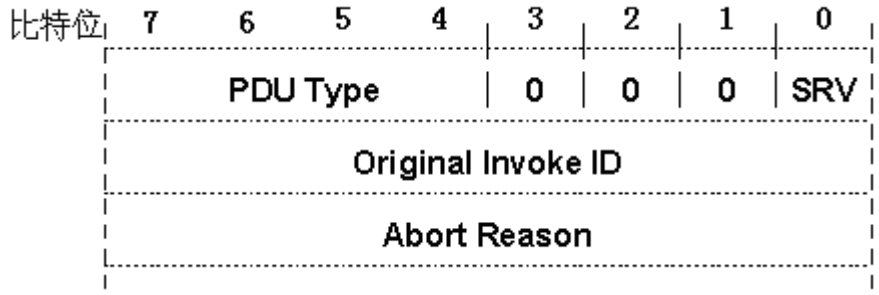
这个参数是正在被中止的事务的‘invokeID’。

20.1.9.3 中止原因（abort-reason）参数

这个参数是 BACnet 中止原因（BACnetAbortReason）类型，包含具有指定‘invokeID’的事务处理被中止的原因。

20.1.9.4 BACnet 中止 PDU 的格式

BACnet-Abort-PDU 的格式如下：



这个 PDU 总是具有三个字节。

PDU 域的值如下：

PDU Type　　=7 (BACnet-Abort-PDU)

SRV　　=0 (Sent by Client)

=1 (Sent by Server)

Original Invoke ID = (0..255)

上面列表中的‘0’位设置为零值。这些位目前不用，保留为 ASHRAE 所用。

20.2 BACnet APDU 的可变部分的编码（首先参考 20.2.12 日期值的编码，容易理解）

在 20.1 节中已经对 BACnet APDU 的头部编码进行了详细规范，本节详细规范 BACnet APDU 可变部分的编码。可变部分也称之为“服务参数（service parameters）”，这些参数的类型为：**BACnet 有证实服务请求**类型， BACnet **无证实服务请求**类型，**BACnet 有证实服务确认**类型，和 **BACnet 差错**类型。在 21 节中使用 ASN.1 的表述方法对每一个参数进行了明确的定义。

服务参数中所有的数据元素都使用称之为“标记（tag）”构件进行标识。每一个标记都对应着一个唯一的参数或子参数。

BACnet 编码使用两类标记。第一类标记标识本标准定义和使用的**基本数据类型**，例如**布尔**类型、**无符号整型**、**字符串**类型、**日期**类型、**时间**类型、或者 **BACnet 对象标识符**类型。如果数据类型用大写字母表示，其语义与对应的 ANS.1 中相同名称的通用数据类型相同，这些在 21 节中已经指出。这样的标记称之为“应用”标记，在 20.2.1.4 节中对标记的值进行详细说明。

第二类标记用于标识这样的数据元素，它们的数据类型可以从它们所处位置的上下文推导出来。这种标记称为“上下文特定（context specific）”标记。

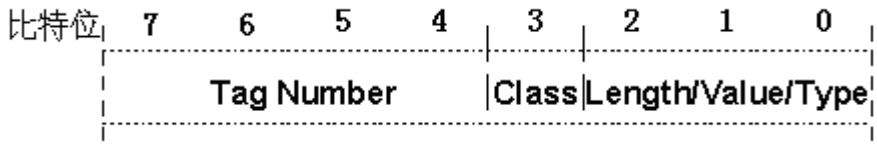
ANS.1 定义另外两种类型的标记，称之为“通用的”和“私有的”。 BACnet 所采用的编码方案不允许使用这种标记，BACnet ASN.1 的编码表述也不要求使用这种类型的标记。

在有些实例中，某个参数的数据类型不能从它所处位置的上下文推导出来。在这种情况下，就需要一个上下文标记和一个或者多个应用标记一起使用。这种情况在 ANS.1 表述中使用服务参数进行说明，而这些服务参数的数据类型由关键词 ABSTRACT-SYNTAX.&TYPE(ANY)，CHOICE，SEQUENCE，或者 SEQUENCE OF 标明。

下面给出每种标记元素的标识、长度规定以及值的编码。

20.2.1 BACnet 标记编码的一般规则

BACnet 标记的编码是一个主字节，后续 0 个或者多个条件的字节。主字节定义如下：



其中：

Tag Number = the tag number with the class【标记编号】

Class = the class of tag (application or context specific)【类别】

Length/Value/Type = where the data following the tag is primitive or constructed and specifies the length or value of primitive data.【长度/值/类型】

20.2.1.1 类别（Class）

如果是应用标记，则类别比特位的值是 0。如果是上下文特定标记，则类别比特位的值是 1。

20.2.1.2 标记编号（Tag Number）

对于取值为 0 到 14（包括 14）的标记编号，在主字节中表示，4—7 四个比特位是标记编号域，其中第 7 比特位是高位。

对于取值为 15 到 254（包括 254）的标记编号，**则在主字节的标记编号域取值为 B’1111’，在主字节其后紧跟一个字节，8 个比特位全部用来表示标记编号，其中第 7 位是高位。**

编码不允许、BACnet 也不要求标记编号大于 254。在后续字节中，值 B’11111111’保留为 ASHRAE 所用。

20.2.1.3 长度/值/类型（Length/Value/Type）

主字节的长度/值/类型域的内容用于区别**原语编码**和**构件编码**，并且标明原语数据的值或者长度。原语编码是数据中不包含其它标记编码的编码，构件编码是数据中包含有其它标记编码的编码。

20.2.1.3.1 原语数据（Primitive Data）

如果被编码数据是应用类别的布尔数据，则当布尔值为 FALSE 时，设置主字节的长度/值/类型域为 B’000’；当布尔值为 TRUE，设置主字节的长度/值/类型域为 B’001’。通过这种方式，完成对布尔值的原语数据的编码，长度/值/类型域表示的是值。

如果被编码数据是原语编码（也就是非构件编码），而且不是应用类别布尔数据，则数据的值按照 20.2.2 节到 20.2.14 节中所述的规则进行编码。主标记字节的长度/值/类型域标明原语数据的长度，规则如下：

如果数据长度为 0 到 4（包括 4）个字节，则在主字节的长度/值/类型域内使用 3 个比特位表示的二进制整数进行编码，其中第 2 比特位是高位。

如果数据长度为 5 到 253（包括 253）个字节，则将主字节的长度/值/类型域的值设置为 B’101’，随后紧跟一个字节，全部 8 个比特位用来表示一个二进制整数，表示数

据的长度，其中第 7 比特位为高位。**如果标记编号的取值为 15 到 254（包括 254），则紧跟主字节之后的是一个标记编号扩展字节（参见 20.2.1.2 节），所以，表示数据长度的字节紧跟在标记编号扩展字节之后。**

如果数据长度为 254 到 65535（包括 65535）个字节，则将主字节的长度/值/类型域的值设置为 B’101’，随后紧跟一个字节，其值取为 D’254’，其后再附加 2 个字节，全部 16 个比特位用来表示一个二进制整数，表示数据的长度，其中第 1 个字节为高位字节。如果标记编号的取值为 15 到 254（包括 254），则紧跟主字节之后的是一个标记编号扩展字节（参见 20.2.1.2 节），所以，表示数据长度的字节紧跟在标记编号扩展字节之后。

如果数据长度为 65536 到 $2^{32}-1$（包括 $2^{32}-1$）个字节，则将主字节的长度/值/类型域的值设置为 B’101’，随后紧跟一个字节，其值取为 D’255’，其后再附加 4 个字节，全部 32 个比特位用来表示一个二进制整数，表示数据的长度，其中第 1 个字节为高位字节。如果标记编号的取值为 15 到 254（包括 254），则紧跟主字节之后的是一个标记编号扩展字节（参见 20.2.1.2 节），所以，表示数据长度的字节紧跟在标记编号扩展字节之后。

大于 $2^{32}-1$ 的数据不能用原语标记编码。

注：除了 8 字节的 IEEE-754 双精度浮点数和特定的比特、字节及字节串之外，BACnet 应用标记原语的长度适合于标记字节，不需要扩展。

20.2.1.3.2 构件数据（Constructed Data）

如果被编码的数据结构中包含有标记元素，则这种编码称为“构件”，它由以下几部分组成：

(a) 一个“opening”标记。这个标记的标记编号域包含标记编号的值，类别域标识为“上下文特定”，长/值/类型域的值为 B’110’；

(b) 完整的编码。这个编码具有标记和组成数据的 0 个、1 个或者多个元素；

(c) 一个“closing”标记。这个标记的类别域和标记编号域的值与“opening”标记的相同，而它的长/值/类型域的值为 B’111’。

在这种情况下，“opening”标记和“closing”标记的长度/值/类型域的值表示的是类型。

*注：一个被包含的标记元素**可能**自身就是一个构件元素，这种循环并不会导致编码混淆，因*

为每一个“opening”标记都必须有一个相应的“closing”标记，构件元素的“opening”标记和“closing”标记包含在每个外部的“opening”标记和“closing”标记之内。

20.2.1.4 应用标记（Application Tags）

编码的 BACnet 应用标记的标记编号域表示应用数据类型，参数分配如下：

标记编号：

0 = Null

1 = Boolean

2 = Unsigned Integer

3 = Signed Integer (2’ complement notation)

4 = Real (ANSI/IEEE-754 floating point)

5 = Double (ANSI/IEEE-754 double precision floating point)

6 = Octet String

7 = Character String

8 = Bit String

9 = Enumerated

10 = Data

11 = Time

12 = BACnetObjectIdentifier

13, 14, 15 = Reserved for ASHRAE

注：所有当前定义的 BACnet 应用数据类型都是原语编码。

20.2.1.5 上下文特定标记（ Context-Specific Tags）

编码的 BACnet 上下文特定标记的标记编号域包含上下文特定标记编号的值。由上下文特定标记标定的数据类型可以是原语编码，也可以是构件编码。

20.2.2 空值的编码（Encoding of a Null Value）

空值表示一个无内容字节，空值是原语编码。

例程：Application-tagged null value

ASN.1 = NULL

Application Tag = Null (Tag Number = 0)

Encoded Tag = X’00’

20.2.3 布尔值的编码（Encoding of a Boolean Value）

应用标记的布尔值在一个单字节中编码，如果编码的值是 FALSE，则设置长度/值/类型域为 B’000’；如果编码的值为 TRUE，则设置长度/值/类型域为 B’001。

例程：Application-tagged Boolean value

ASN.1 = BOOLEAN

Value = FALSE

Application Tag = Boolean (Tag Number = 1)

Encoded Tag = X’10’

上下文特定标记的布尔值原语数据包含一个内容字节。如果编码的值为 FALSE，则这个字节为 B’00000000’； 如果编码的值为 TRUE，则这个字节为 B’00000001’。

例程：Context-tagged Boolean value

ASN.1 = [2] BOOLEAN

Value = TRUE

Context Tag = 2

Encoded Tag = X’29’

Encoded Data = X’01’

注：在编码中，布尔数据类型的编码方式与其它的数据类型不同，表现在上下文特定标记的布尔值的编码和应用标记的布尔值的编码不一样。这样做的原因是，应用标记的值在一个单字节中编码，没有内容字节，而如果要在上下文特定标记的情况下使用与应用标记相同的编码值，则为了区分在长度/值/类型域中的内容是长度还是值，必须知道上下文关系，这是不希望出现的情况。参见 20.2.20 节。

20.2.4 无符号整型数值的编码（Encoding of a Unsigned Integer Value）

无符号整型数值是原语编码，具有至少一个内容字节。

无符号整型数值在内容字节中被编码成为一个取值范围为 0 到$(2^{8*L}-1)$的二进制数，其中 L 是用于对这个值编码的字节数，L 至少为 1。被编码成多个字节的值在传送时，先从高位字节传送。所有无符号整数都应该尽可能地用最少的字节数来编码。也就是说，每一个多字节编码值的第一个字节都不应该是 X’00’。

例程：Application-tagged unsigned integer

ASN.1 =Unsigned

Value = 72

Context Tag = Unsigned Integer (Tag Number = 2)

Encoded Tag = X’21’

Encoded Data = X’48’

20.2.5 有符号整型数值的编码（Encoding of a Signed Integer Value）

有符号整型数值是原语编码，具有至少一个内容字节。

有符号整型数值在内容字节中被编码成为一个取值范围为$-2^{(8*L)}$到$(2^{8*L}-1)$的、使用 2 的补码标记的二进制数，其中 L 是用于对这个值编码的字节数，L 至少为 1。被编码成多个字节的值在传送时，先从高位字节传送。所有有符号整数都应该尽可能地用最少的字节数来编码。也就是说，对于每一个多字节编码值来说，如果第二个字节的最高位比特（比特 7）是 0，则第一个字节不应该是 X’00’，如果第二个字节的最高位比特是 1，则第一个字节不应该是 X’FF’。

例程：Application-tagged signed integer

ASN.1 =INTEGER

Value = 72

Application Tag = Signed Integer (Tag Number = 3)

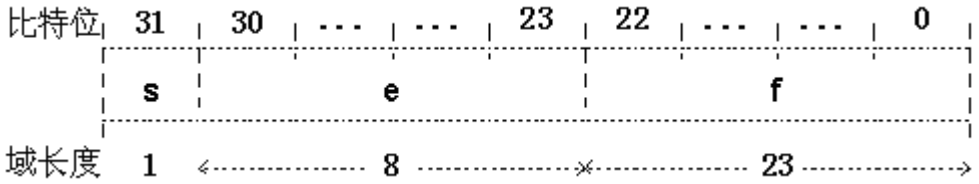
Encoded Tag = X’31’

Encoded Data = X’48’

20.2.6 实数的编码（Encoding of a Real Number Value）

实数是原语编码，具有 4 个内容字节。实数使用 ASN.1L/IEEE 标准 754-1985 中规范的方法，“IEEE Standard for Binary Floating-Point Arithmetic”，进行编码。对于细节问题应该参考这个标准。在传送这个多字节编码值时，先从最高位字节（符号和指数）传送。

单精度实数的编码格式如下：



其中，域长度的数值是比特数。使用方程式 $v=(-1)^{s}2^{e-127}(1\cdot f)$表示非零值，其中，符号·表示二进制点。在方程式中，设置 s、e 和 f 为 0，表示零。

例程：Application-tagged single precision real

ASN.1 = REAL

Value = 72.0

Application Tag = Real (Tag Number = 4)

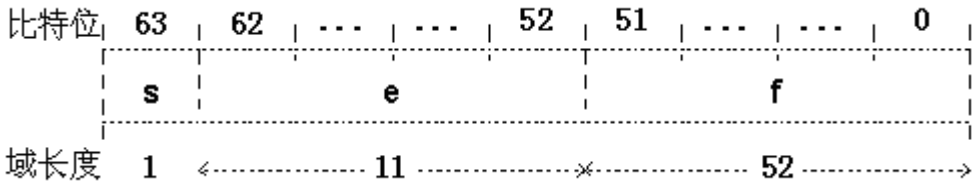
Encoded Tag = X’ 44’

Encoded Data = X’ 42900000’

20.2.7 双精度实数的编码（Encoding of a Double Precision Real Number Value）

双精度实数是原语编码，具有 8 个内容字节。双精度实数使用 ASN.1L/IEEE 标准 754-1985 中规范的方法，“IEEE Standard for Binary Floating-Point Arithmetic”，进行编码。对于细节问题应该参考这个标准。在传送这个多字节编码值时，先从最高位字节（符号和指数）传送。

双精度实数的编码格式如下：



其中，域长度的数值是比特数。使用方程式 $v=(-1)^{s}2^{e-1023}(1\cdot f)$ 表示非零值，其中，符号 · 表示二进制点。在方程式中，设置 s、e 和 f 为 0，表示零。

例程：Application-tagged double precision real

ASN.1 = Double

Value = 72.0

Application Tag = Double (Tag Number = 5)

Encoded Tag = X’ 55’

Encoded Data = X’ 4052000000000000’

BACnet 不支持 ASNI/IEEE-754 扩展精度格式。

20.2.8 字节串值的编码（Encoding of a Octet String Value）

字节串值是原语编码。字节串的编码包含 0 个、1 个或者多个内容字节，其字节数等于它们的数据值的字节数，其顺序与它们的数据值表示的顺序一样，数据值中的一个字节的最高位与内容字节中的一个字节的最高位对应。

例程：Application-tagged octet string

ASN.1 = OCTET

Value = X ‘1234FF’

Application Tag = Octet String (Tag Number = 6)

Encoded Tag = X’ 63’

Encoded Data = X’ 1234FF’

20.2.9 字符串值的编码 (Encoding of a Character String Value)

字符串是原语编码。字符串的编码包含一个主内容字节和 0 个、1 个或者多个附加内容字节，这些字节等于它们的数据值的字节，其顺序与它们的数据值表示的顺序一样，也就是说，最高位的字节在先，数据值中的一个字节的最高位与内容字节中的一个字节的最高位对应。

主字节用下面编码表示字符集：

X’ 00’ ANSI X3.4

X’ 01’ IBM / Microsoft DBCS

X’ 02’ JIS C 6226

X’ 03’ ISO 10646(UCS-4)

X’ 04’ ISO 10646(UCS-2)

X’ 05’ ISO 8859-1

主字节的其它值都被保留为 ASHRAE 所用。

例程：Application-tagged charater string

ASN.1 = Character String

Value = “This is a BACnet string” (ANSI X3.4)

Application Tag = Character String (Tag Number = 6)

Encoded Tag = X’ 75’

Length Extension= X’ 19’

Character Set = X’ 00’

Encoded Data =X’ 546869732069732061204241436E657420737472696E6721’

在 IBM / Microsoft DBCS(X’ 01’)的情况下，主字节后紧跟着两个附加字节，这两个附加字节是无符号整数，高位字节在先，它表示后面字符的设定代码页。

例程：Application-tagged character string(DBCS)

ASN.1 = Character String

Value = “This is a BACnet string!” (IBM / Microsoft DBCS,code page 850)

Application Tag = Character String (Tag Number = 7)

Encoded Tag = X’ 75’

Length Extension= X’ 1B’

Encoded Data=X’ 0103525468697320697320612042414 36E657420737472696E6721’

在 ISO 10646 UCS-2(X’04’)和 UCS-4(X’03’)情况下，字符串的每个字符分别由两个或者四个字节表示。UCS-2 的字节顺序是 Row-Cell，而 UCS-4 的字节顺序是 Group-Plane-Row-Cell。

例程：Application-tagged character string(UCS-2)

ASN.1 = Character String

Value = “This is a BACnet string!”(ISO 10646 UCS-2)

Application Tag = Character String (Tag Number = 7)

Encoded Tag = X’75’

Length Extension= X’31’

Encoded Data =X’040054006800690073002000690073002000610020004200410043006E006500740020007300740072006900 6E00670021’

20.2.10 比特串值的编码（Encoding of a Bit String Value）

比特串是原语编码。原语码的内容字节包含一个主字节和其后的 0 个或者多个字节，这些字节包含了比特位串。主字节编码成为一个无符号二进制整数，表示最后一个后续字节中未使用的比特的数目，未使用的比特的数目可以在 0 到 7 之间取值，包括 0 和 7。

本标准中定义的比特位串，例如，**状态标志**属性，按照定义的顺序编码，其被定义的第一个布尔值位于第一个后续字节的最高位比特位，即比特 7。比特位串中的比特依次从第一个后续字节的第 7 到第 0 位顺序放置，然后是第二个字节的第 7 到 0 位，第三个字节的第 7 位到第 0 位，等等，直到放置完所有比特的最后一个后续字节，都是从第 7 位开始。如果比特位串是个空位串，则没有后续字节，且主字节为 0。

例程：Application-tagged bit string

ASN.1 = BIT STRING

Value = B’10101’

Application Tag = Bit String (Tag Number = 8)

Encoded Tag = X’82’

Encoded Data = X’03A8’

20.2.11 枚举值的编码（Encoding of a Enumerated Value）

枚举值是原语编码，具有至少一个内容字节。

枚举值在内容字节中被编码成为一个取值范围为 0 到$(2^{8*L}-1)$的二进制数，其中 L 是用于对这个值编码的字节数，L 至少为 1。被编码成多个字节的值在传送时，先从高位字节传

送。所有枚举值都应该尽可能地用最少的字节数来编码。也就是说，任何一个多字节编码值的第一个字节都不应该是 X’00’。

例程：Application-tagged enumeration

ASN.1 =BACnetObjectType

Value = ANALOG-INPUT(0)

Application Tag = Enumerated (Tag Number = 9)

Encoded Tag = X’91’

Encoded Data = X’00’

20.2.12 日期值的编码（Encoding of a Date Value）

日期值是原语编码，具有 4 个内容字节。

日期值在内容字节中被编码成为四个二进制整数，第一个字节表示年份减去 1900；第二个字节表示月份，一月用 1 表示；第三个字节表示月中的日；第四个字节表示星期，星期一用 1 表示。这四个字节中只要有一个字节中的值为 X’FF’=D’255’，则表示该值超出规定的范围。如果四个字节都等于 X’FF’，则对应的日期表示“任意”或者“不必管”。

例程：Application-tagged date value

ASN.1 =Date

Value = January 24,1991(Day of week = Thursday)

Application Tag = Date (Tag Number = 10)

Encoded Tag = X’A4’

Encoded Data = X’5B011804’

20.2.13 时间值的编码（Encoding of a Time Value）

时间值是原语编码，具有 4 个内容字节。

时间值在内容字节中被编码成为四个二进制整数，第一个内容字节表示小时，这里采用 24 小时制（1 P.M = D’13’）；第二个字节表示小时内的分钟；第三个字节表示分钟内的秒钟；第四个字节表示一秒的百分之几。这四个字节中只要有一个字节的值为 X’FF’=D’255’，则表示该值超出了规定的范围。如果四个字节都等于 X’FF’，则对应的时间表示“任意”或者“不必管”。

例程：Application-tagged time value

ASN.1 =Time

Value = 5:35:45.17 P.M.=17:35:45.17

Application Tag = Time (Tag Number = 11)

Encoded Tag = X’ B4’

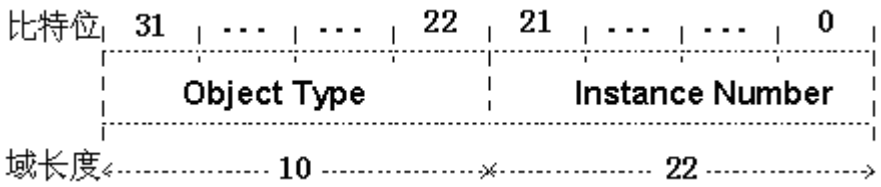
Encoded Data = X’ 11232D11’

20.2.14 对象标识符值的编码（Encoding of an Object Identifier Value）

一个 BACnet **对象标识符**的值由两部分组成：

(1) 一个 10 比特的对象类型，表示对象的 BACnet **对象类型**属性值，其中比特 9 是最高位，比特 0 是最低位。对于本标准定义的对象，这个域值由 21 节所述的 BACnet **对象类型**枚举值确定。

(2) 一个 22 比特的对象实例编号，其中比特 21 是最高位，比特 0 是最低位。



对象标识符值是原语编码，具有 4 个内容字节，表示如下：

比特 9 至比特 2 所表示的对象类型被编码成为第一个内容字节的第 7 位至第 0 位，而比特 1 至比特 0 所表示的对象类型被编码成为第二个内容字节的第 7 位至第 6 位。

比特 21 至比特 16 所表示的对象实例被编码成为第二个内容字节的第 5 位至第 0 位，比特 15 至比特 8 所表示的对象实例编码成为第三个内容字节的第 7 位至第 0 位，比特 7 至比特 0 所表示的对象实例被编码成为第四个内容字节的第 7 位至第 0 位。

例程：Application-tagged object identifier value

ASN.1 = ObjectIdentifier

Value = (Binary Input,15)

Application Tag = ObjectIdentifier (Tag Number = 12)

Encoded Tag = X’ C4’

Encoded Data = X’ 00C0000F’

20.2.15 标记值的编码（Encoding of a Tagged Value）

标记值的编码由对应的数据值的完整编码推导而得。

ISO 8824 定义了关键字“**隐式（IMPLICIT）**”和“**显式（EXPLICIT）**”，缺省为显式。第 21 节，使用“**隐式标记定义（DEFINITION IMPLICIT TAGS）**”作为开始，将缺省值改变

成为**隐式**。BACnet ASN.1 的定义都采用这种缺省方式，只是在作为强制实行时，才使用**显式**。

如果“**显式**”关键字用于类型的表述中，则编码是构件编码，内容字节是包含标记的完整基本编码。

如果“**显式**”关键字没有被用于类型的定义中，则

a) 如果基本代码是构件代码，则类型定义为构件编码，否则，是原语编码，或者

b) 如果基本代码不是原语标记应用布尔类别，则内容字节应和基本代码的内容字节相同，或者

c) 如果基本代码是原语标记应用布尔类别，则内容字节用值 B’00000000’表示布尔值为 FALSE，用值 B’00000001’表示布尔值 TRUE。

下面的例程所表示的上下文标记编号仅用来说明目的。

例程：Context-tagged null value

ASN.1 = [3] NULL

Encoded Tag = 3

Encoded Data = X’38’

例程：Context-tagged Boolean value

ASN.1 = [6]BOOLEAN

Value = FALSE

Context Tag = 6

Encoded Tag = X’69’

Encoded Data = X’00’

例程：Context-tagged unsigned integer

ASN.1 = [0] Unsigned

Value = 256

Context Tag = 0

Encoded Tag = X’0A’

Encoded Data = X’0100’

例程：Context-tagged signed integer

ASN.1 = [5] INTEGER

Value = -72

Context Tag = 5

Encoded Tag = X’59’

Encoded Data = X’ B8’

例程：Context-tagged single precision real

ASN.1 = [0] REAL

Value = -33.3

Context Tag = 0

Encoded Tag = X’ 0C’

Encoded Data = X’ C2053333’

例程：Context-tagged double precision real

ASN.1 = [1] Double

Value = -33.3

Context Tag = 0

Encoded Tag = X’ 1D’

Extended Length =X’ 08’

Encoded Data = X’ C040A66666666666’

例程：Context-tagged octet string

ASN.1 = [1] OctetString

Value = X’ 4321’

Context Tag = 1

Encoded Tag = X’ 1A’

Encoded Data = X’ 4321’

例程：Context-tagged character string

ASN.1 = [5] CharacterString

Value = “This is a BACnet string!” (ANSI X3.4)

Context Tag = 5

Encoded Tag = X’ 5D’

Length Extension = X’ 19’

Character Set =X’ 00’ (ASNI X3.4)

Encoded Data =X’ 546869732069732061204241
436E657420737472696E6721’

例程：Context-tagged bit string

ASN.1 = [0] BIT STRING

Value = B’ 10101’

Context Tag = 0

Encoded Tag = X’ 0A’

Unused Bits in Last Octet =X’ 08’

Encoded Data = X ‘A8’

例程：Context-tagged enumeration

ASN.1 = [9] BACnetObjectType

Value = ANALOG-INPUT(0)

Context Tag = 9

Encoded Tag = X’ 99’

Encoded Data = X’ 00’

例程：Context-tagged date value

ASN.1 = [9] Date

Value = January 24,1991 (Day of week = Thursday)

Context Tag = 9

Encoded Tag = X’ 9C’

Encoded Data = X’ 5B011805’

例程：Context-tagged time value

ASN.1 = [4] Time

Value = 5:35:45.17 P.M.=17:35:45.17

Context Tag = 4

Encoded Tag = X’ 4C’

Encoded Data = X’ 11232D11’

例程：Context-tagged object identifier value

ASN.1 = [4] ObjectIdentifier

Value = (Binary Input, 15)

Context Tag = 4

Encoded Tag = X’ 4C’

Encoded Data = X’ 00C0000F’

20.2.16 序列值的编码（Encoding of a Sequence Value）

序列值的编码由包含标记的完整编码组成，这个编码是由每一个列于 ASN.1 表述的序列类型中的一个数据值的编码所得来的，其顺序与出现在定义中的顺序相同，只有在引用类型时使用了关键字“**可选的**”这种情况除外。

对于使用了关键字“可选的”引用的类型，其数据值的编码可以存在，但是并不是必需的。如果编码存在，则它在编码中所处于的位置应和相应的 ASN.1 定义的类型出现的位置相同。

例程：SEQUENCE value

```
ASN.1              = BACnetDateTime
Value              = January 24,1991,5:35:45.17 P.M.
Application Tag       = Date (Tag Number = 9)
Encoded Tag        = X’94’
Encoded Data       = X’5B011805’
Application Tag       = Time (Tag Number = 10)
Encoded Tag        = X’A4’
Encoded Data       = X’11232D11’
```

例程：Context-tagged SEQUENCE value

```
ASN.1              = [0] BACnetDateTime
Value              = January 24, 1991, 5:35:45.17 P.M.
Context Tag            = 0
Encoded Tag        = X’0E’  (opening tag)
      Application Tag = Date (Tag Number = 9)
      Encoded Tag     = X’94’
      Encoded Data    = X’5B011805’
      Application Tag = Time (Tag Number = 10)
      Encoded Tag     = X’A4’
      Encoded Data    = X’11232D11’
      Encoded Tag     = X’0F’  (closing tag)
```

包含构件元素的所有序列的 ASN.1 表述都有明确的标记，这样才能进行准确的编码和解码。下面的例程表示出这种要求。

```
( incorrect usage)
Var1 :: = SEQUENCE {
```

```
        varn1 SEQUENCE {
              varn2  [1] INTEGER,
              varn3  [2] INTEGER OPTIONAL
              },
        varn4 SEQUENCE {
              varn5  [1] INTEGER OPTIONAL
              varn6  [2] INTEGER
              }
        }

  (correct usage)
  Var1 :: = SEQUENCE {
        varn1 SEQUENCE {
              varn2  [1] INTEGER,
              varn3  [2] INTEGER OPTIONAL
              },
        varn4 SEQUENCE {
              varn5  [3] INTEGER OPTIONAL
              varn6  [4] INTEGER
              }
        }
```

20.2.17 Sequence-Of 值的编码（Encoding of a Sequence-Of Value）

Sequence-Of 值的编码由 0 个、1 个或者多个包含标记的完整编码组成，这些编码是由列于 ASN.1 定义的类型中的数据值的编码所得来的。数据值的编码顺序和在被编码的 sequence-of 之中的数据值的顺序一致。

例程：SEQUENCE OF primitive data

```
ASN.1          = SEQUENCE OF INTEGER
Value          = 1,2,4
Application Tag    = Unsigned Integer (Tag Number = 2)
Encoded Tag    = X’21
Encoded Data   = X’01’
Application Tag    = Unsigned Integer (Tag Number = 2)
```

Encoded Tag = X’21’

Encoded Data = X’04’

例程：Context-tagged SEQUENCE OF primitive data

ASN.1 =[1] SEQUENCE OF INTEGER

Value = 1, 2, 4

Encoded Tag = X’1E’ (opening tag)

Application Tag =Unsigned Integer (Tag Number = 2)

Encoded Tag = X’21’

Encoded Data = X’01’

Application Tag =Unsigned Integer (Tag Number = 2)

Encoded Tag = X’21’

Encoded Data = X’01’

Encoded Tag = X’1F’ (closing tag)

例程：SEQUENCE OF constructed data

ASN.1 = SEQUENCE OF BACnetDateTime

Value = (January 24, 1991, 5:00 P.M.),

(January 24, 1991, 6:45 P.M.)

Application Tag = Date (Tag Number = 9)

Encoded Tag = X’94’

Encoded Data = X’5B011805’

Application Tag = Time (Tag Number = 10)

Encoded Tag = X’A4’

Encoded Data = X’11000000’

Application Tag = Date (Tag Number = 9)

Encoded Tag = X’94’

Encoded Data = X’5B011805’

Application Tag = Time (Tag Number = 10)

Encoded Tag = X’A4’

Encoded Data = X’122D0000’

20.2.18 选择值的编码（Encoding of a Choice Value）

选择值的编码和已选中的类型的值的编码相同，选择值可以是原语编码，也可以是构件编码，依据所选择的类型而定。

例程：CHOICE of primitive data

ASN.1 = BACnetTimeStamp

Value = 5:35:45.17 P.M.= 5:35:45.17

Context Tag = 0 (Choice for ‘time’ in BACnetTimeStamp)

Encoded Tag = X’0C’

Encoded Data = X’11232D11’

例程：CHOICE of constructed data

ASN.1 = BACnetTimeStamp

Value = January 24, 1991, 5:45.17 P.M.

Context Tag = 2 (Choice for ‘dateTime’ in BACnetTimeStamp)

Encoded Tag = X’2E’ (opening tag)

Application Tag =Date (Tag Number = 9)

Encoded Tag = X’94’

Encoded Data = X’5B011804’

Application Tag =Time (Tag Number = 10)

Encoded Tag = X’A4’

Encoded Data = X’11232D11’

Encoded Tag = X’2F’ (closing tag)

20.2.19 ANY 类型值的编码（Encoding of a Value of the ANY Type）

一个 ANY 类型值的编码是一个本标准规范的完整编码，其中用占位符 ANY 替代值类型。在 ASN.1 中的 ABSTRACT-SYNTAX .&Type 对这种编码进行了说明。

20.2.20 标记规则总结（Summary of the Tagging Rules）

虽然在不了解上下文的情况下不能够对一个 BACnet PDU 的标记部分进行解释，但是 20.2 节中对标记规则的描述，使得能够在不了解上下文的情况下，准确地解析一个具有标记的比特流。

(a) 一个比特流的第一个字节是一个标记，这个标记或者是上下文特定类别，或者是应用类别。

(b) 如果标记是应用类别，则根据 20.2 节中的定义，可以知道标记的数据的格式和内容。特别地，可以跳过数据，找出比特流中的下一个标记。

(c) 如果标记是上下文特定类别和原语编码，则它包含某种类型的原语数据（没有被标记）。标记的长度/值/类型域指定了这个数据的长度。这样，当数据的数据类型

和格式不知道时，它的长度仍然可准确知道。这样就可以跳过数据，找出比特流中的下一个标记。

(d) 如果标记是构件编码（长度/值/类型 = 6），则它是一对标记的开始标记。跟在这个标记后面的是 0 个或者多个标记的元素，然后是这一对标记的结束标记（长度/值/类型 = 7）。在开始标记和结束标记之间的标记比特流也可以根据同样的四条规则来解析，其过程是循环递减的，直到找出原语标记，或者直到再没有标记。

21. 应用层协议数据单元的正式描述

本节给出 ASN.1 模块，这个模块定义所有的 BACnet APDU，及其数据类型。在 13 节至 17 节中有许多被定义为条件的或者可选的服务参数，这些参数在 ASN.1 的表述中都标注为 OPTINAL，说明这些参数可以在 PDU 中存在，也可以不在 PDU 中存在。在 ASN.1 表述中只使用 OPTIONAL 标注，不改变在服务规范中对条件的参数的要求。

```
BACnetModule DEFINITIONS IMPLICIT TAGS :: =
BEGIN

-- ******************** APDU 定义********************

BACnetPDU :: = CHOICE {
    confirmed-request-PDU           [0] BACnet-Confirmed-Request-PDU,
    unconfirmed-request-PDU         [1] BACnet-Unconfirmed-Request-PDU,
    simpleACK-PDU                 [2] BACnet-SimpleACK-PDU,
    complexACK-PDU                [3] BACnet-ComPlexACK-PDU,
    segmentAck-PDU                 [4] BACnet-SegmenACK-PDU,
    error-PDU                       [5] BACnet-Error-PDU,
    reject-PDU                      [6] BACnet-Reject-PDU,
    abort-PDU                       [7] BACnet-Abort-PDU
    }

BACnet-Confirmed-Request-PDU:: = SEQUENCE {
    pdu-type                      [0] Unsigned (0..15), --这个 PDU 类型使用 0
    segmented-message             [1] BOOLEAN,
    more-follows                  [2] BOOLEAN,
    segmented-response-accepted    [3] BOOLEAN,
```

reserved [4] Unsigned (0..31), --必须设置为 0
max-APDU-length-accepted [5] Unsigned (0.. 15), --如 20. 1.2.4 节所分配
invokeID [6] Unsigned (0..255),
sequence-number [7] Unsigned (0..255) OPTIONAL, --只在分段报文中使用
proposed-window-size [8] Unsigned (1.. 127) OPTIONAL, --只在分段报文中使用
service-choice [9] BACnetConfirmedServiceChoice,
service-request [10]BACnet-Confirmed-Service-Request OPTIONAL
-- 在头部编码中不使用上下文特定标记 0..10
}

BACnet-Unconfirmed-Request-PDU :: = SEQUENCE {
pdu-type [0] Unsigned (0..15), --这个 PDU 类型使用 1
reserved [1] Unsigned (0.. 15), --必须设置为 0
service-choice [2] BACnetUnconfirmedServiceChoice,
service-request [3] BACnet-Unconfirmed-Service-Request
-- 在头部编码中不使用上下文特定标记 0..3
}

BACnet-SimpleACKPDU :: = SEQUENCE {
pdu-type [0] Unsigned (0.. 15), --这个 PDU 类型使用 2
reserved [1] Unsigned (0.. 15), --必须设置为 0
invokeID [2] Unsigned (0..255),
service-ACK-choice [3] BACnetConfirmedServiceChoice
-- 在头部编码中不使用上下文特定标记 0..3
}

BACnet-ComplexACK-PDU ::= SEQUENCE {
pdu-type [0] Unsigned (0..15), --这个 PDU 类型使用 3
segmented-message [1] BOOLEAN,
more-follows [2] BOOLEAN,
reserved [3] Unsigned (0..3), --必须设置为 0
invokeID [4] Unsigned (0..255),
sequence-number [5] Unsigned (0..255) OPTIONAL,--只在分段报文中

使用

proposed-window-size [6] Unsigned (1.. 127) OPTIONAL,--只在分段报文中使用

service-ACK-choice [7] BACnetConfirmedServiceChoice,

service-ACKt [8]BACnet-Confirmed-Service-ACK

-- 在头部编码中不使用上下文特定标记 0..8

}

BACnet-SegmentACK-PDU ::= SEQUENCE {

pdu-type [0] Unsigned (0..15), --这个 PDU 类型使用 4

reserved [1] Unsigned (0..3), --必须设置为 0

negative-ACK [2] BOOLEAN,

server [3] BOOLEAN,

original-invokeID [4] Unsigned (0..255),

sequence-number [5] Unsigned (0..255),

actual-window-size [6] Unsigned (1.. 127)

-- 在头部编码中不使用上下文特定标记 0..6

}

BACnet-Error-PDU :: = SEQUENCE {

pdu-type [0] Unsigned (0..15), --这个 PDU 类型使用 5

reserved [1] Unsigned (0..15), --必须设置为 0

original-invokeID [2] Unsigned (0..255),

error-choice [3] BACnetConfirmedServiceChoice,

error [4] BACnet-Error

-- 在头部编码中不使用上下文特定标记 0..4

}

BACnet-Reject-PDU :: = SEQUENCE {

pdu-type [0] Unsigned (0..15), --这个 PDU 类型使用 6

reserved [1] Unsigned (0.. 15), --必须设置为 0

original-invokeID [2] Unsigned (0..255),

reject-reason [3] BACnetRejectReason

-- 在头部编码中不使用上下文特定标记 0..4

}

```
BACnet-Abort-PDU  :: = SEQUENCE {
    pdu-type                [0] Unsigned (0..15), --这个 PDU 类型使用 7
    reserved            [1] Unsigned (0..7), --必须设置为 0
    server              [2] BOOLEAN,
    original-invokeID       [3] Unsigned (0..255),
    abort-reason         [4] BACnetAbortReason
-- 在头部编码中不使用上下文特定标记 0..4
    }
```

********** 有证实服务的表述 **********

```
BACnetconfirmedserviceChoice  :: = ENUMERATED {
-- Alarm and Event Services
    acknowledgeAlarm            (0),
    confirmedCOVNotification        (1),
    confirmedEventNotification      (2),
    getAlarmSummny              (3),
    getEnrollmentSummary            (4),
    subscribeCOV                (5),

-- File Access Services
    atomicReadFile              (6),
    atomicWriteFile                 (7),

-- Object Access Services
    addListElement              (8),
    removeListElement           (9),
    createobjeet                    (10),
    deleteObjeCt                    (11),
    readProperty                    (12),
    readPropertyCondtional          (13),
    readProPertyMultiple            (14),
    writeProperty                   (15),
```

```
    writeProPertyMultiPle           (16),

-- Remote Device Management Service
    deviceCommunicationControl   (17),
    confirmedPrivateTransfer        (18),
    confirmedTestMessage            (19),
    reinitializeDevice              (20),

-- Virtual Terminal Services
    vtOpen                       (21),
    vtClose                         (22),
    vtData                       (23),

-- Security Services
    authenticate                    (24),
    requestKey                   (25)
    }
```

--新定义的服务可以增加到上述的枚举类型中。本表述中的所有枚举值都保留为 ASHRAE 用作定义中。使用**有证实专有转换**服务或者**无证实专有转换**服务进行专有服务扩展。参见 23 节。

```
BACnet-confirmed-Service-Request  :: = CHOICE {
-- Alarm and Event Services
    acknowledgeAlarm          [0] AckowledgeAlarm-Request,
    confirmedCOVNotification      [1]ConfirmedCOVNotification-Request,
    confirmedEventNotification    [2]ConfirmedEventNotification-Request,
    -- getAlarmSummary conveys no parameters
    getEnrollmentSummary          [4] GetEnrollmentSummary - Requst,
    subscribeCOV              [5] SubscribeCOV-Request,

-- File Access Services
    atomicReadFile            [6] AtomicReadFile- Request,
    atomicWriteFile               [7] AtomicWrit File-Request,

-- Object Access Services
```

```
    addListElement              [8] AddListElement-Reqest,
    removeListElement           [9] RemoveListElement-Request,
    createObject                    [10] CreateObect-Request,
    deleteObject                    [11] DeleteObeCt-Request,
    readProperty                    [12] ReadPrOPrty-Request,
    readProPertyConditional         [13] ReadProperty Conditional-Request,
    readPropenyMultiple             [14]ReadProPertyMultiple-Request,
    writeProperty                   [15] WriteProperty-Request,
    writeProPertyMultiPle           [16] WriteProPertyMultiple - Request,

-- Remote Device Management Services
    deviceCommunicationControl  [17]DeviceCommunicationControl- Requet,
    confirmedPrivateTransfer        [18]ConfirmedPrivateTransfer-Request,
    confirmedTextMesssage       [19] ConfirmedTextMessage-Requet,
    reinitializeDevice              [20] Reinitialize Device-Request,

-- Virtual Terminal Services
    vtOpen                      [21] VT-Open-Request,
    vtClosc                         [22] VT-Close-Request,
    vtData                      [23] VT-DataRequest,

-- Security services
    authenticate                    [24] Anthenicate-Request,
    requestKey                  [25] RequestKey-Request
    }
```

-- 在编码中没有使用上下文特定标记 0..25。标记编号在**BACnet 有证实请求 PDU** 中作为服务选择参数传送。

--新定义的服务可以增加到上述的枚举类型中。本表述中的所有枚举值都保留为 ASHRAE 用作定义中。使用**有证实专有转换**服务进行专有服务扩展。参见 23 节。

BACnet-Confirmed-Service-ACK :: = CHOICE{

-- 这个表述表示为每个有证实服务所定义的‘**Result(+)**’参数，这些服务在‘**Result(+)**’中返回一个或者多个参数。

```
-- Alarm and Event Services
```

```
    getAlarmSummary          [3] GetAlarmSummary-ACK,
    getEnrollmentSummary     [41 GetEnrolImentSummary-ACK,

-- File Access Services
    atomicReadFile           [6] AtomicReadFile-ACK,
    atomicWriteFile              [7] AtomicWriteFile-ACK,

--Object Access Services
    createObject                 [10] CreateObject-ACK,
    readProperty                 [12] ReadProperty-ACK,
    readPropertyConditional      [13] ReadPropertyConditional-ACK,
    readProPertyMultiple         [14] ReadPropertyMultiPle-ACK,

-- Remote Device Management Services
    confirmedPrivateTransfer     [18] ConfirmedPrivateTransfer-ACK,

-- Virtual Terminal Services
    vtOpen                   [21] VT-Open-ACK,
    vtData                   [23] VT-Data-ACK,

-- Security Services
    authenticate                 [24] Authenticate-ACK
```

-- 在编码中没有使用上下文特定标记 3..24。标记编号在**BACnet 复杂确认 PDU** 中作为服务确认选择参数传送。

--新定义的服务可以增加到上述的枚举类型中。本表述中的所有枚举值都保留为 ASHRAE 用作定义中。使用**有证实专有转换**服务进行专有服务扩展。参见 23 节。

```
    }
```

--********** **有证实报警和事件服务**********

```
AcknowledgeAlarmRequest  :: = SEQUENCE {
    acknowledgingProcessIdentifier        [0] Unsigned,
    eventObjectIdentifier                [1] BACnetObjectIdentifier,
    evenStateAcknowledged           [2] BACnetEventState,
    timeStamp                       [3] BACnetTimeStamp,
```

```
    acknowledgmenSource                 [4] CharacterString,
    timeOfAcknowledgment            [5] BACnetTimeStamp
    }

ConfirmedCOVNotification-Request  :: = SEQUENCE {
    subscriberProcess Idenifier             [0] Unsigned,
    initiatingDeviceIdenifier           [1] BACnetObjectIdentifier,
    monitoredObjectIdenifier            [2] BACnetObectIdenifier,
    timeRemaining                   [3] Unsigned,
    listOfValues                        [4]SEQUENCE OF BACnetProperty Value
    }

ConfirmedEventNotification-Request  :: = SEQUENCE{
    processIdentifier                   [0] Unsigned,
    initiatingDeviceIdentifier          [1] BACnetObjectIdentifier,
    eventObjectIdentifier               [2] BACnetObectIdentifier,
    timeStamp                       [3] BACnetTimeStamp,
    notificationClass                   [4] Unsigned,
    priority                            [5] Unsigned8,
    eventType                       [6] BACnetEventType,
    messageText                         [7] CharacterString OPTIONAL,
    notifyType                      [8] BACnetNotifyType,
    ackRequired                         [9] BOOLEAN OPTIONAL
    fromState                       [10] BACnetEventState OPTIONAL
    toState                             [11] BACnetEVentState OPTIONAL.
    eventValues                         [12]       BACnetNotificationParameters
OPTIONAL
    }

GotAlarmSummary-ACK  :: = SEQUENCE OF SEQUENCE {
    objectIdentifier          BACnetObjectIdentifier,
    alarmState            BACnetEventState.
    acknowledgedTransitions  BACnetEventTransitionBits
    }
```

```
GotEnrollmentSummary-Request  :: = SEQUENCE {
    acknowledgementFilter        [0] ENUMERATED {
                                    all             (0),
                                    acked       (1),
                                    not-acked   (2)
                                    },
    enrollmentFilter             [1]BACnetRecipientProcess OPTIONAL,
    eventStateFilter             [2] ENUMERATED {
                                    offnormal   (0),
                                    fault       (1),
                                    normal      (2),
                                    all             (3),
                                    active      (4)
                                    } OPTIONAL,
    eventTypeFilter              [3] BACnetEvenType OPTIONAL,
    priorityFilter               [4] SEQUENCE {
                                    minPriority  [0] Unsigned8,
                                    maxPriority  [1] Unsigned8
                                    } OPTIONAL,
    notificationClassFilter          [5] Unsigned OPTIONAL
    }

GetEnrolImentSummary-ACK :: = SEQUENCE OF SEQUENCE {
    objectIdentifier        BACnetObjectIdenifier,
    eventTyPe           BACnetEvenType,
    eventState          BACnetEventState
    priority                Unsigned8,
    notificationClass       Unsigned OPTIONAL
    }

SubscribeCOV-Request  :: = SEQUENCE {
    subscriberProcessIdentifier         [0] Unsigned,
    monitoredObectIdentifier        [1] BACnetObjectIdentifier,
    issueConfirmedNotifications     [2] BOOLEAN OPTIONAL,
    lifetime                        [3] Unsigned OPTIONAL
```

```
    }

-- ****************** 有证实文件访问服务 ********************

AtomicReadfile-Request  :: = SEQUENCE {
    fileIdentifier   BACnetObectIdentifier,
    accessMethod  CHOICE {
                    streamAccess  [0] SEQUENCE {
                                      fileStartPosition     INTEGER,
                                      requestedOctetCount   Unsigned
                                      },
                    recordAccess  [1] SEQUENCE {
                                      fileStartRecord        INTEGER,
                                      requestedRecordCount Unsigned
                                      }
                    }
    }

AtomicReadfile-ACK  :: = SEQUENCE {
    endOfFile    BOOLEAN,
    accessMethod CHOICE {
                 streamAccess [0] SEQUENCE {
                                    fileStartPosition    INTEGER,
                                    fileData            OCTET STRING
                                    },
                 recordAccess [1] SEQUENCE {
                                  fileStartRecord        INTEGER,
                                  returnedRecordCount  Unsigned,
                                  fileRecordDare    SEQUENCE OF OCTET STRING
                                  }
                 }
    }

AtomicWritefIle-Request  ::= SEQUENCE {
    fileIdentifier   BACnetObjectIdenifier,
```

```
    accessMethod  CHOICE {
                    streamAccess    [0] SEQUENCE {
                                    fileStartPositinn  INTEGER,
                                    fileData        OCTET STRING
                                    },
                    recordAccess [1] SEQUENCE {
                                    fileSartRecord  INTEGER,
                                    recordCOunt       Unsigned,
                                    fileRecordData  SEQUENCE OF OCTET STRING
                                    }
                    }
    }

AtomicWriteFile-ACK  :: = CHOICE
    fileStartPosition   [0] INTEGER,
    fileStartRecord [1] INTEGER
    }

-- ********** 有证实对象访问服务 **********

AddListElement-Request  :: = SEQUENCE {
    objectIdenifier        [0] BACnetObjectIndifier,
    propertyIdenifier      [1] BACnetProPertyIdenifier,
    propertyArrayIndex   [2] Unsigned OPTIONAL, --只与数组数据类型一起使用
    listOffilements        [3] ABSTRACT-SYNTAX.&Type
    }

CreaeObect-Request  :: = SEQUENCE
    objectSpecifier     [0] CHOICE {
                          objectTyPe     [0]BACnetObjectType,
                          objectIdentifier  [1]BACnetObjectIdentifier
                          },
    listOfInitialValues     [1] SEQUENCE OF BACnetPrpertyValue OPTIONAL
    }
```

```
CreatObject-ACK  :: = BACnetObjectIdentifier

DeleteObjectRequest  :: = SEQUENCE {
    ObjectIdenifier     BACnetObjectIdenifier
    }

ReadProperty-Request  :: = SEQUENCE {
    objectIdentifier      [0] BACnetObjectIdenifier,
    propertyIdentifier    [1] BACnetProPertyIdentifier,
    propertyArrayIndex  [2] Unsigned OPTIONAL --只与数组数据类型一起使用
                            --如果数组中省略这个参数，表示引用整个数组
    }

ReadProperty-ACK  :: = SEQUENCE {
    objectIdentifier      [0] BACnetObjectIdentifier.
    proPertyIdenifier     [1] BACnetPropertyIdentifier,
    PropertyArrayIndex  [2] Unsigned OPTIONAL, --只与数组数据类型一起使用
                            --如果数组中省略这个参数，表示引用整个数组
    propertyValue       [3] ABSTRACT-SYNTAX.& Type
    }

ReadProPertyConditional-Request  :: = SEQUENCE{
    objectSelectionCriteria       [0] SEQUENCE {
            selectionLogic  [0] ENUMERATED {
                            and     (0),
                            or  (1),
                            all     (2)
                            },
            listOfSelectionCriteria     [1] SEQUENCE OF SEQUENCE{
                            propertyIdenifier   [0] BACnetPropertyIdentifier,
                            propertyArrayIndex  [1]Unsigned OPTIONAL,
                            relationSpecifier   [2] ENUMERATED {
                                    equal               (0),
                                    not-equal           (1),
                                    ess-than                (2),
```

```
                                    greater-than            (3),
                                    less-than-or-equal      (4),
                                    greater-than-or-equal   (5)
                                    }
                            comparisonValue [3] ABSTRACT-SYNTAX.&Type
                            } OPTIONAL
            },
    listOfPropertyReferences  [1]SEQUENCE OF BACnetPropertyReference OPTIONAL
    }

ReadProPertyConditional-ACK  :: = SEQUENCE {
    listOfReadAccessResults  SEQUENCE OF ReadAccessResult OPTIONAL
    }

ReadPrpertyMultiple-Request  :: = SEQUENCE {
    listOfReadAccessSpecs  SEQUENCE OF ReadAccessSpecification
    }

ReadPropertyMultiple-ACK  :: = SEQUENCE {
    listOfReadAccessResults  SEQUENCE OF ReadAccessResult
    }

RemoveListElement-Request  :: = SEQUENCE {
    objectIdenifier         [0] BACnetObjectIdenifier,
    propertyIdenifier       [1] BACnetPrOpertyIdentifier,
    propertyArrayIndex  [2] Unsigned OPTIONAL, --只与数组数据类型一起使用
    listOfElements      [3] ABSTRACT-SYNTAX.&Type
    }

WriteProperty-Request  :: = SEQUENCE {
    objectIdenifier       [0] BACnetObjectIdentifier,
    propertyIdentifier    [1] BACnetPropertyIdentifier,
    propertyArrayIndex  [2] Unsigned OPTIONAL, --只与数组数据类型一起使用
                            --如果数组中省略这个参数，表示引用整个数组
    propertyValue    [3] ABSTRACT-SYNTAX.&Type,
```

```
    priority              [4] Unsigned8 (1..16) OPTIONAL --只与命令属性一起使用
    }

WritePropertyMultiple-Request  :: = SEQUENCE{
    ListOfwriteAccessSpecifications SEQUENCE OF WriteAccessSpecification
    }

-- ************ 有证实远程设备管理服务 **********

DeviceCommunicationControl-Requet  :: = SEQUENCE {
    timeDuration    [0] Unsigned16 OPTIONAL,
    enable-disable      [1] ENUMERATED {
                        enable  (0),
                        disable     (1)
                        },
    password        [2] CharacterString (SIZE(1..20)) OPTIONAL
    }

ConfirmedPrivateTransfer-Request  :: = SEQUENCE {
    vendorID            [0] Unsigned,
    serviceNumber       [1] Unsigned,
    serviceParameters       [2] ABSTRACT-SYNTAX.&Type OPTIONAL
    }

ConfirmedPrivateTransferACK  :: = SEQUENCE {
    vendorID            [0] Unsigned,
    serviceNumber       [1] Unsigned,
    resultBlock             [2] ABSTRACT-SYNTAX.&Type OPTIONAL
    }

ConfirmedTextmessage-Request  :: = SEQUENCE {
    textMessageSourceDevice     [0] BACnetObectIdentifier,
    messageClass                [1] CHOICE {
                                numeric         [0] Unsigned,
                                character   [1] CharacterString
```

```
                                } OPTIONAL,
    messagePriority             [2] ENUMERATED {
                                normal      (0),
                                urgent      (1)
                                },
    message                     [3] CharacterString
    }

ReinitializeDevice-Request  :: = SEQUENCE {
    reinitializedStateOfDevice  [0] ENUMERATED {
                                coldstart (0),
                                warmstart (1)
                                },
    password                [1] CharacterString (SIZE (1..20)) OPTIONAL
    }

-- ********** 有证实虚拟终端服务 **********

VT-Open-Request  :: = SEQUENCE {
    vtClass                     BACnetVTClass,
    localVTSessionIdentifier    Unsigned8
    }

VT-Open-ACK  :: = SEQUENCE {
    RemoteVTSessionIdentifier   Unsigned8
    }

VT-Close-Request  :: = SEQUENCE {
    listOfRemoteVTSessionIdentifier  SEQUENCE OF Unsigned8
    }

VT-Data-Request  :: = SEQUENCE {
    vtSessionIdenifier  Unsigned8,
    vtNewData       OCTET STRING,
    vtDataFlag      Unsigned (0.. 1)
```

```
    }

VT-Data-ACK  :: = SEQUENCE {
    allNewDataAccepted  [0] BOOLEAN,
    aceeptedOctetCount     [1] Unsigned OPTIONAL --只在 allNewDataAccepted = FALSE
                                                --条件下存在
    }

-- ****************** 有证实安全服务 ******************

Authenticate-Request  :: = SEQUENCE {
    pseudoRandomNumber    [0] Unsigned32,
    exPectedInvokeID          [1] Unsigned8 OPTIONAL,
    operatorName          [2] CharacterString OPTIONAL,
    operatorPassword          [3] CharacterString (SIZE (1..20)) OPTIONAL,
    startEnciPheredSession    [4] BOOLEAN OPTIONAL
    }

Authenticate-ACK  :: = SEQUENCE {
    modifiedRandomNumber     Unsigned32
    }

RequestKey-Request  :: = SEQUENCE {
    requestingDeviceIdentifier  BACnetObjectIdenifier,
    requestingDeviceAddress     BACnetAddress,
    remoteDeviceIdenifier       BACnetObectIdentifier,
    remoteDeviceAddress         BACnetAddress
    }

--************ 无证实请求的表述 *****************

BACnetUnconfirmedServiceChoice  :: = ENUMERATED {
    i-Am                            (0),
    i-Have                          (1),
    unconfirmedCOVNotification      (2),
```

```
    unconfirmedEventNotification        (3),
    unconfirmedPrivateTransfer          (4),
    unconfirmedTextMessage          (5),
    timeSynchronizaion              (6),
    Who-Has                         (7),
    who-Is                          (8)
    }
```

--新定义的服务可以增加到上述的枚举类型中。本表述中的所有枚举值都保留为 ASHRAE 用作定义中。使用**无证实专有转换**服务进行专有服务扩展。参见 23 节。

```
BACnet-Unconfirmed-Service-Request  :: = CHOICE {
    i-Am                        [0] I-Am-Request,
    i-Have                      [1] I-Have-Request,
    unconfirmedCOVNotification    [2]UnconfirmedCOVNotification-Request,
    unconfirmedEventNotification   [3] UnconfirmedEventNotification-Reqest,
    unconfirmedPrivateTransfer      [4] UnconfirmedPrivateTransfer-Request,
    unconfirmedTextMessage      [5] UnconfirmedTextMessage-Request,
    timeSynchronizaion          [6] TimeSynchronization-Request,
    who-Has                     [7] Who-Has-Request,
    Who-Is                      [8] Who-Is-Request
    }
```

-- 在编码中没有使用上下文特定标记 0..8。标记编号在 **BACnet 无证实请求 PDU** 中作为服务选择参数传送。

--新定义的服务可以增加到上述的枚举类型中。本表述中的所有枚举值都保留为 ASHRAE 用作定义中。使用**有证实专有转换**服务或者**无证实专有转换**服务进行专有服务扩展。参见 23 节。

```
    }
```

-- ********** **无证实报警和事件服务** **********

```
UnconfirmedCOVNotificationHequest  :: = SEQUENCE {
    subscriberProcessIdentifier         [0] Unsigned,
    initiatingDeviceIdentifier      [1] BACnetObjectIdenifier,
    monitoredObjectIdentifier       [2] BACnetObjectIdentifier,
```

```
    timRemaining                [3] Unsigned,
    listOfValues                    [4] SEQUENCE OF BACnetPropertyValue
    }

UnconfirmedEventNotification-Request  :: = SEQUENCE {
    processIdentifier           [0] Unsigned,
    initiatingDeviceIdentifier  [1] BACnetObjectIdenifier,
    eventObjectIdentifier       [2] BACnetObjectIdentifier,
    timeStamp               [3] BACnetTimeStamp,
    notificationClass           [4] Unsigned,
    priority                    [5] Unsigned8,
    eventType               [6] BACnetEventType,
    messageText                 [7] CharacterString OPTIONAL,
    notifyType              [8] BACnetNotifyType,
    ackRequired                 [9] BOOLEAN OPTIONAL,
    fromState                   [10] BACnetEventState OPTIONAL,
    toState                     [11] BACnetEventState OPTIONAL,
    evenValues              [12] BACnetNotificationParameters OPTIONAL
    }

-- ********** 无证实远程设备管理服务 *********

I-Am-Request  :: = SEQUENCE {
    iAInDeviceIdentifier            BACnetObjectIdentifier,
    maxAPDULengthAccepted       Unsigned,
    segmentationSupported           BACnetSegmentation,
    vendorID                    Unsigned
    }

I-have-Request  :: = SEQUENCE {
    deviceIdentifier                BACnetObjectIdentifier,
    objectIdenifier                 BACnetObjectIdentifier,
    objectName                  CharacterString
    }
```

```
UnconfirmedPrivateTransfer-Request  :: = SEQUENCE {
    vendorID            [0] Unsigned,
    serviceNumber       [1] Unsigned,
    serviceParameters       [2] ABSTRACT-SYNTAX.&Type OPTIONAL
    }

UnconfirmedTextMessage-Request  :: = SEQU ENCE {
    textMessageSourceDevice     [0] BACnetObjectIdentifier,
    messageClass            [1] CHOICE {
                                  numeric   [0] Unsigned,
                                  characer  [1] CharacterString
                                  } OPTIONAL,
    messagePriority             [2] ENUMERATED {
                                  normal    (0),
                                  urgent    (1)
                                  },
    message                     [3] CharacterString
    }

TimeSynchroniation-Request  :: = SEQUENCE {
    time BACnetDateTime
    }

Who-Has-Request  :: = SEQUENCE {
    limits  SEQUENCE {
              deviceInstanceRangeLowLimit    [0] Unsigned (0..4194303),
              deviceInstanceRangeHighLimit    [1] Unsigned (0..4194303)
              } OPTIONAL,
    object  CHOICE {
              objectIdentifier                  [2] BACnetObjectIdenifier,
              objectName                    [3] CharacerString
              }
    }

Who-Is-Request  :: == SEQUENCE {
```

```
    DeviceInstanceRangeLowLimit  [0] Unsigned (0..4194303) OPTIONAL,
                                                    -- 必须成对使用，参见 16.9 节
    deviceInstanceRangeHighLimit    [1] Unsigned (0..4194303) OPTIONAL
                                                    -- 必须成对使用，参见 16.9 节
    }
```

--********** 差错报文的表述 **********

```
BACnetAbortReason  :: = ENUMERATED {
    Other                           (0),
    buffer-overflow                     (1),
    invalid-apdu-in-this-state          (2),
    preempted-by-higher-Priority-task   (3),
    segmentation-not-supported          (4),
    …….
    }
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
BACnet-Error  :: = CHOICE {
    other                       [127] Error,

-- 本表述其余的选择表示对有证实服务所定义的‘Result(-)’参数。

-- Alarm and Event Services
    acknowledgeAlarm            [0] Error,
    confirmedCOVNotification        [1] Error,
    confirmedEventNotification      [2] Error,
    getAlarmSummary             [3] Error,
    getEnrollmentSummary            [4] Error,
    subscribeCOV                [5]Error,

-- File Access Services
    atomicReadFile              [6]Error,
    atomicWriteFile                 [7] Error,
```

```
-- Object Access Services
    addListElement              [8] ChangeList-Error,
    removeListElement           [9] ChangeList-Error,
    createObject                    [10] CreateObject-Error,
    deleteObject                    [11] Error,
    readProPerty                    [12] Error,
    readPropertyConditional         [13] Error,
    readPropertyMultiple            [14] Error,
    writeProperty                   [15] Error,
    writeProPertyMultiPle           [16] WritePropertyMultiPle-Error,

-- Remote Device Management Services
    deviceCornmunicationControl     [17] Error,
    confirmedPrivateTransfer        [18]ConfirmedPrivateTransfer- Error,
    confirmedTextMessage            [19] Error,
    reinitializeDevice              [20] Error,

-- Virtual Terminal Services
    vtOPen                      [21] Error,
    vtClose                         [22] VTClose-Error,
    vtData                      [23] Error,

-- Security Services
    authenicate                     [24] Error,
    requestKey                  [25] Error
    }
```

-- 在编码中没有使用上下文特定标记 0..25 和 127。标记编号在 **BACnet 差错 PDU** 中作为差错选择参数传送。

--新定义的服务可以增加到上述的枚举类型中。本表述中的所有枚举值都保留为 ASHRAE 用作定义中。参见 23 节。

```
BACnetRejectReason  :: = ENUMERATED {
    other                       (0),
    buffer-overflow                 (1),
```

```
    inconsistent-Parameters             (2),
    invalid-Parameter-datatype      (3),
    invalid-tag                     (4),
    missing-required-parameter      (5),
    parameter-out-of range          (6),
    too-man-arguments           (7),
    undefined-enumeration           (8),
    unrecognized-service            (9),
    }
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
ChangeLiSt-Error  :: = SEQUENCE {
    errorType                   [0] Error,
    firstFailedElementNumber        [1] Unsigned
    }

CreateObect-Error  :: = SEQUENCE {
    errorType                    [0] Error,
    firstFailedElementNumber      [1] Unsigned
    }

ConfirmedPrivateTransfor-Error  :: =SEQUENCE{
    errorType                  [0] Error,
    vendorID                   [1] Unsigned,
    serviceNumber              [2] Unsigned,
    errorParameters                [3] ABSTRACT-SYNTAX.&Type OPTIONAL
    }

Error  :: = SEQUENCE {
```

--注：在 18 节中定义了错误类别和错误代码的有效结合。

```
    error-class     ENUMERATED {
                device      (0) ,
                object      (1) ,
```

```
                property        (2) ,
                resources   (3),
                security        (4),
                services        (5),
                vt          (6),
                ...
                },
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
    error-code  ENUMERATED {
                other                                   (0),
                authentication-failed                       (1) ,
                character-set-not-supported                     (41),
                configuration-in-progress                   (2),
                device-busy                                 (3),
                dynamic-creation-not-supported              (4),
                file-access-denied                          (5),
                incompatible-security-levels                    (6) ,
                inconsistent-parameters                         (7) ,
                inconsistent-selection-criterion                (8),
                Invalid-array-index                         (42),
                Invalid-data-type                           (9),
                invalid-file-access-method                  (10),
                invalid-file-start-position                     (11),
                operator-name                           (12),
                invalid-parameter-data-type                     (13),
                invalid-time-stamp                          (14),
                key-generation-error                        (15),
                missing-required-parameter                  (16),
                no-objects-of-specified-type                    (17),
                no-space-for-object                         (18),
                no-space-to-add-list-element                    (19),
                no-space-to-write-property                  (20) ,
                no-vt-sessions-available                        (21),
```

object-deletion-not-permitted (23),
object-identifier-already-exists (24),
operational-problem (25),
password-failure (26),
property-is-not-a-list (22),
read-access-denied (27),
security-not-supported (28) ,
service-request-denied (29) ,
timeout (30),
unknown-object (31),
unknown-property (32),
this enumeration was removed (33),
unknown-vt-class (34) ,
unknown-vt-session (35),
unsupported-object-type (36),
value-out-of range (37),
vt-session-already-closed (38),
vt-session-termination-failure (39),
write-access-denied (40),
...
}

--枚举值 0-255 保留为 ASHRAE 用作定义中。枚举值 256-65535 可以被其它机构根据 23 节中的规定使用。本版本使用的最后一个枚举值是 42。

}

WritePropertyMultiple-Error :: = SEQUENCE {
errorType [0] Error,
firstFailedWriteAttempt [1] BACnet Object Property Reference
}

VTClose-Error :: = SEQUENCE {
errorType [0] Error,
listOfVTSessionIdentifiers [1] SEQUENCE OF Unsigned8 OPTIONAL
}

--**********应用类型 **********

-- 下面的表述是关于应用数据类型的定义，参见 20.2.1.4 节。

```
NULL                [APPLICATION 0], equivalent to [UNIVERSAL 5]
BOOLEAN          [APPLICATION 1], equivalent to [UNIVERSAL 1]
Unsigned          :: = [APPLICATION 21 INTEGER (0..MAX)
UnSigned8         :: = Unsigned (0..255)
UnSigned16    :: = Unsigned (0..65535)
Unsigned32        :: = Unsigned (0..4294967295)
INTEGER          [APPLICATION 3], equivalent to [UNIVERSAL 2]
REAL                [APPLICATION 4], equivalent to [UNIVERSAL9] ANSI/IEEE-754
                    single precision floating point
Double         :: = [APPLICATION 51 OCTET STRING (SIZE(8)) -- ANSI/IEEE-754
                    double precision floating point
OCTET STRING    [APPLICATION 6], equivalent to [UNIVERSAL 4]
CharacterString :: = [APPLICATION 71 OCTET STRING -- see 20.2.9 for supported types
BIT STRING      [APPLICATION 8], equivalent to [UNIVERSAL 3]
ENUMERATED  [APPLICATION 9], equivalent to [UNIVERSAL 10]
Date            :: = [APPLICATION 10] OCTET STRING (SIZE(4)) --参见 20.2.12 节
--              first octet         year minus 1900             X  ‘  FF’   =
unspecified
--              second octet    month (1..12)               1 = January
                                                        X ‘FF’ = unspecified
--              third octet     day of month (1..31),       X ‘FF’ = unspecified
--              fourth octet    day of week (1..7)          1 = Monday
--                                                          7 = Sunday
--                                                          X ‘FF’ = unspecified
Time            :: = [APPLICATION 11] OCTET STRING (SIZE(4)) -- see 20.2.13
--              first octet         hour                (0..23),  (X  ‘ FF’)  =
Unspecified
--              second octet    minute              (0..59),   (X  ‘  FF’)   =
Unspecified
--              third octet     second              (0..59),   (X  ‘  FF’)   =
unspeCified
```

-- fourth octet hunredths (0..99), (X ‘ FF’) = Unspecified

BACnetObjectIdentifier :: = [APPLICATION 12] OCTET STRING (SIZE(4))

--参见 20.2.14 节

--********** **基本类型** **********

BACnetAction :: = ENUMERATED {

direct (0),

reverse (1)

}

BACnetActionCommand :: = SEQUENCE {

deviceIdentifier [0] BACnetObjectIdentifier OPTIONAL,

objectIdentifier [1] BACnetObjectIdenifier,

propertyIdenifier [2] BACnetPropertyIdentifier,

propertyArrayIndex [3] Unsigned OPTIONAL, --只与数组数据类型一起使用

propertyValue [4] ABSTRACT-SYNTAX.&Type,

priority [5] Unsigned (1..16) OPTIONAL, --只与命令属性一起使用

postDelay [6] Unsigned OPTIONAL,

quitOnFailure [7] BOOLEAN,

writeSuccessful [8] BOOLEAN

}

BACnetActionList :: = SEQUENCE{

action [0] SEQUENCE OF BACnetActionCornmand

}

BACnetAddress :: = SEQUENCE {

network-number Unsigned16,

mac-address OCTET STRING --长度为 0 的字节串表示广播

}

BACnetAddressBinding :: = SEQUENCE {

deviceObjectIdentifier BACnetObjectIdentifier,

```
    deviceAddress       BACnetAddress
    }

BACnetBinaryPV  :: = ENUMERATED {
    inactive    (0),
    active  (1)
    }

BACnetCalendarEntry  :: = CHOICE {
    date        [0] Date,
    dateRange   [1] BACnetDateRange,
    weekNDay    [2] BACnetWeekNDay
    }

BACnetDailySchedule  :: = SEQUENCE {
    day-schedule      [0] SEQUENCE OF BACnetTimeValue
    }

BACnetDateRange  :: = SEQUENCE {
    startDate   Date,
    endDate         Date
    }

BACnetDateTime  :: = SEQUENCE {
    date        Date,
    time        Time
    }

BACnetDaysOfWeek  :: = BIT STRING {
    monday      (0),
    tuesday         (1),
    wednesday   (2),
    thursday        (3),
    friday      (4),
    saturday    (5),
```

```
    sundav      (6)
    }

BACnetDestination  :: = SEQUENCE {
    validDays                   BACnetDaysOfWeek,
    fromTime                    Time,
    toTime                      Time,
    recipient                       BACnetRecipient,
    processIdentifier               Unsigned,
    issueConfirmedNotifications     BOOLEAN,
    transitions                     BACnetEventTransitionBits
    }

BACnetDeviceStatus  :: = ENUMERATED {
    Operational             (0),
    operational-read-only   (1),
    download-required       (2),
    download-in-progress    (3),
    non-operational         (4),
    }
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
BACnetEngineeringUnits  :: = ENUMERATED {
--Area
    square-meters                   (0),
    square-feet                         (1),
--currency
    currency1                       (105),
    currency2                       (106),
    currency3                       (107),
    currency4                       (108),
    currency5                       (109),
    currency6                       (110),
    currency7                       (111),
```

```
    currency8                       (112),
    currency9                       (113),
    currency10                      (114),
--Electrical
    milliamperes                        (2),
    amperes                             (3),
    ohms                            (4),
    kilohms                             (122),
    megohms                         (123),
    volts                           (5),
    millivolts                      (124),
    kilovolts                           (6),
    megavolts                       (7),
    volt-amperes                        (8),
    kilovolt-amperes                    (9),
    megavolt-amperes                    (10),
    volt-amperes-reactive               (11),
    kilovolt-amperes-reactive           (12),
    megavolt-amperes-reactive           (13),
    degrees-phase                   (14),
    power-factor                        (15),
--Energy
    joules                          (16),
    kilojoules                      (17),
    kilojoules-per-kilogram                 (125),
    megajoules                      (126),
    watt-hours                      (18),
    kilowatt-hours                  (19),
    btus                            (20),
    therms                          (21),
    ton-hours                       (22),
--Enthalpy
    joules-per-kilogram-dry-air             (23),
    btus-per-pound-dry-air              (24),
--Entropy
```

joules-per-degree-Kelvin (127),
joules-per-kilogram-degree-Kelvin (128),
--Frequency
cycles-per-hour (25) ,
cycles-per-minute (26) ,
hertz (27) ,
kilohertz (129) ,
megahertz (130) ,
per-hour (131) ,
--Humidity
grams-of-water-per-kilogram-dry-air (28) , .
percent-relative-humidity (29) ,
--Length
millimeters (30) ,
meters (31) ,
inches (32) ,
feet (33) ,
--Light
watts-per-square-foot (34) ,
Watts-per-square-meter (35),
lumens (36) ,
luxes (37),
foot-candles (38) ,
--Mass
kilograms (39) ,
pounds-mass (40) ,
tons (41) ,
--Mass Flow
kilograms-per-second (42),
kilograms -per-minute (43),
kilograms-per-hour (44) ,
pounds-mass-per-minute (45) ,
pounds - mass-per-hour (46) ,
--Power
milliwatts (132) ,

```
    watts                          (47) ,
    kilowatts                      (48) ,
    megawatts                      (49) ,
    btus-per-hour                      (50) ,
    horsepower                     (51) ,
    tons-refrigeration                 (52) ,
--Pressure
    pascals                            (53),
    hectopascals                       (133),
    kilopascals                        (54),
    millibars                          (134),
    bars                           (55),
    pounds-force-per-square-inch       (56),
    centimeters-of-water               (57),
    inches-of water                    (58),
    millimeters-of-mercury             (59),
    centimeters-of-mercury             (60),
    inches-of mercury                  (61),
--Temperature
    degrees-Celsius                    (62),
    degrees-Kelvin                 (63),
    degrees-Fahrenheit             (64),
    degree-days-Celsius                (65),
    degree-days-Fahrenheit             (66),
--Time
    years                          (67),
    months                         (68),
    weeks                          (69),
    days                           (70),
    hours                          (71),
    minutes                            (72),
    seconds                            (73),
--Velocity
    meters-per-second                  (74),
    kilometers-per-hour                (75),
```

```
    feet-per-second                 (76),
    feet-per-minute                 (77),
    miles-per-hour              (78),
--volume
    cubic-feet                  (79),
    cubic-meters                    (80),
    imperial-gallons                (81),
    liters                      (82),
    us-gallons                  (83),
--Volumetric Flow
    cubic-feet-per-minute           (84),
    cubic-meters-per-second         (85),
    cubic-meters-per-hour           (135),
    imperial-gallons-per-minute         (86),
    liters-per-second               (87),
    liters-per-minute               (88),
    liters-per-hour                 (136),
    us-gallons-per-minute           (89),
--Other
    degrees-angular                 (90),
    degrees-Celsius-per-hour        (91),
    degrees-Celsius-per-minute      (92),
    degrees-Fahrenheit-per-hour         (93),
    degrees-Fahrenheit-per-minute   (94),
    kilowatt-hours-per-square-meter     (137),
    kilowatt-hours-per-square-foot  (138),
    megajoules-per-square-meter     (139),
    megajoules-per-square-foot      (140),
    no-units                        (95),
    parts-per-million               (96),
    parts-per-billion               (97),
    percent                         (98),
    percent-per-second          (99),
    per-minute                  (100),
    per-second                  (101),
```

```
    psi-per-degree-Fahrenheit          (102),
    radians                          (103),
    revolutions-per-minute             (104),
    watts-per-square-meter-degree-kelvin (141),
    …
    }
```

--枚举值 0-255 保留为 ASHRAE 用作定义中。枚举值 256-65535 可以被其它机构根据 23 节中的规定使用。本版本使用的最后一个枚举值是 141，枚举值 115-121 已经被删除。

```
BACnetEventParameter  :: = CHOICE {
-- 这些选择与事件类型枚举值一一对应。

    change-of-bitstring   [0] SEQUENCE {
                          time-delay              [0] Unsigned,
                          bitmask             [1] BIT STRING,
                          list-of-bitstring-values   [2] SEQUENCE OF BIT STRING
                          },
    change-of state       [1] SEQUENCE {
                          time-delay        [0] Unsigned,
                          list-of-values    [1] SEQUENCE OF BACnetPropertyStates
                          },
    change-of-value       [2] SEQUENCE {
                          time-delay  [0] Unsigned,
                          cov-criteria  [1] CHOICE {
                                      bitmask                  [0] BIT STRING,
                                      referencd-property-increment  [1] REAL
                                      }
                          },
    command-failure       [3] SEQUENCE {
                          time-delay                 [0] Unsigned,
                          feedback-property-reference                    [1]
BACnetObjectPropertyReference
                          },
    floating-limit        [4] SEQUENCE {
                          time-del4y            [0] Unsigned,
```

```
                        setpoint-reference      [1]
BACnetObjectPropertyReference,
                        low-diff-limit          [2] REAL,
                        high-diff-limit         [3] REAL,
                        deadband            [4] REAL
                        },
    out-of-range        [5] SEQUENCE {
                        time-delay              [0] Unsigned,
                        low-limit               [1] REAL,
                        high-limit              [2] REAL,
                        deadband            [3] REAL
                        }
    }

BACnetEventState  :: = ENUMERATED {
    normal      (0),
    fault       (1),
    offnormal   (2),
    high-limit  (3),
    low-limit   (4),
    }
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
BACnetEventTransitionBits  :: == BIT STRING {
    to-offnormal    (0),
    to-fault        (1),
    to-normal   (2)
    }

BACnetEventType  :: = ENUMERATED {
    change-of-bitstring     (0),
    change-of-state         (1),
    change-of-value         (2),
    command-failure         (3),
```

```
    floating-limit          (4),
    out-of range            (5),
    }
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
BACnetFileAccessMethod  :: = ENUMERATED {
    record-access               (0),
    stream-access               (1),
    record-and-stream-access    (2)
    }

BACnetLimitEnable  :: = BIT STRING {
    lowLimitEnable      (0),
    highLimitAnatile        (1)
    }


BACnetNotificationParamsters  :: = CHOICE {
-- 这些选择与事件类型枚举值一一对应。
    change-of-bitstring   [0] SEQUENCE {
                            referened-bitstring     [0] BIT STRING,
                            status-flags        [1] BACnetStatusFlags
                            },
    change-of-state       [1] SEQUENCE {
                            new-state       [0] BACneProPertyStates,
                            status-flags        [1] BACnetStatusFlags
                            },
    change-of-value       [2] SEQUENCE {
                            new-value       [0] CHOICE {
                                                changed-bits  [0] BIT STRING,
                                                changed-value [1] REAL
                                                },
                            status-flags        [1] BACnetStatusFlags
                            },
```

```
    command-failure       [3] SEQUENCE {
                            command-value   [0] ABSTRACT-SYNTAX.&Type,
                                                -- 与引用属性有关
                            status-flags        [1] BACnetStatusFlags,
                            feedback-value  [2] ABSTRACT-SYNTAX.&Type
                                                -- 与引用属性有关
                            },
    floating-limit        [4] SEQUENCE {
                            reference-value     [0] REAL,
                            status-flags        [1] BACnetStatusFlags,
                            setpoint-value  [2] REAL,
                            error-limit         [3] REAL
                            },
    out-of-range          [5] SEQUENCE {
                            exceeding-value     [0] REAL,
                            status-flags        [1] BACnetStatusFlags,
                            deadband        [2] REAL,
                            exceeded-limit  [3] REAL
                            },
    complex-event-type  [6] SEQUENCE OF BACnetPropertyValue
    }

BACnetNotifyType  :: = ENUMERATED {
    alarm           (0),
    event           (1),
    ack-notification    (2)
    }

BACnetObjectPropertyReference  :: = SEQUENCE {
    objectIdentifier       [0] BACnetObjectIdentifier,
    propertyIdentifier     [1]BACnetPropertyIdentifier,
    propertyArrayIndex   [2] Unsigned OPTIONAL--只与数组数据类型一起使用
                            --如果数组中省略这个参数，表示引用整个数组
    }
```

```
BACnetObjectPrOpertyValue  :: = SEQUENCE {
    objectIdentifier       [0] BACnetObjectIdenifier,
    propertyIdenifier      [1] BACnetPropertyIdentifier,
    propertyArrayIndex   [2] Unsigned OPTIONAL, --只与数组数据类型一起使用
                              --如果数组中省略这个参数，表示引用整个数组
    value              [3] ABSTRACT-SYNTAX.&Type, --任何适合指定属性的数据类型
    priority               [4] Unsigned (1.. 16) OPTIONAL
    }

BACnetObjectType  :: = ENUMERATED {
    analog-input        (0),
    analog-output   (1),
    analog-value        (2),
    binary-input        (3),
    binary-output       (4),
    binary-value        (5),
    calendar            (6),
    command         (7),
    device          (8),
    event-enrollment    (9),
    file                (10),
    group           (11),
    loop            (12),
    multistate-input    (13),
    multistate-output   (14),
    notification-class  (15),
    program         (16),
    schedule        (17),
    }
```

--枚举值 0-127 保留为 ASHRAE 用作定义中。枚举值 128-1023 可以被其它机构根据 23 节中的规定使用。

```
BACnetObjectTypesSupported  :: = BIT STRING {
    analog-input        (0),
    analog-output   (1),
```

```
    analog-value        (2),
    binary-input        (3),
    binary-output       (4),
    binary-value        (5),
    calendar        (6),
    command         (7),
    device          (8),
    event-enrollment    (9),
    file            (10),
    group           (11),
    loop            (12),
    multistate-input    (13),
    multistate-output   (14),
    notification-class  (15),
    program         (16),
    schedule        (17)
    }
--比特位 0-127 保留为 ASHRAE 用作定义中。
BACnetPolarity  :: = ENUMERATED {
    normal          (0),
    reverse             (1)
    }

BACnetProrityArray  :: = SEQUENCE SIZE (1.. 16) OF BACnetPriorityValue
    -- 作为一个 BACnet 数组被访问

BACnetPriorityValue  :: = CHOICE {
    null                NULL,
    real                    REAL,
    binary              BACnetBinaryPV,
    integer                 Unsigned,
    constructedValue    [0] ABSTRACT-SYNTAX.&Type
    }

BACnetProgramError  :: = ENUMERATED {
```

```
    normal          (0),
    load-failed         (1),
    internal            (2),
    program         (3),
    other           (4),
    }
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
BACnetProgramRequest  :: = ENUMERATED {
    ready           (0),
    load            (1),
    run             (2),
    halt                (3),
    restart             (4),
    unload          (5)
    }

BACnetProgramState  :: = ENUMERATED {
    idle                (0),
    loading             (1),
    running             (2),
    waiting             (3),
    halted          (4),
    unloading       (5)
    }

BACnetPropertyIdentifier  :: = ENUMERATED {
    acked-transitions                       (0),
    ack-required                            (1),
    action                                  (2),
    action-text                             (3),
    active-text                             (4),
    active-vt-sessions                      (5),
    alarm-value                             (6),
```

```
alarm-values                                (7),
all                                         (8),
all-writes-successful                       (9),
apdu-segment-timeout                        (10),
apdu-timeout                                (11),
application-software-version                    (12),
archive                                     (13),
bias                                    (14),
change-of-state-count                       (15),
change-of-state-time                        (16) ,
-- see notification-class                       (17) ,
-- the property in this place was deleted     (18),
controlled-variable-reference                   (19) ,
controlled-variable-units                   (20),
controlled-variable-value                   (21),
cov -increment                          (22) ,
datelist                                    (23 ) ,
daylight-savings- status                        (24) ,
deadband                                (25) ,
derivative-constant                         (26) ,
derivative-constant-units                   (27),
description                                 (28) ,
description-of-halt                         (29) ,
device-address-binding                     (30),
device-type                                 (31) ,
effective-period                            (32) ,
elapsed-active-time                         (33) ,
error-limit                                 (34) ,
event-enable                                (35) ,
event-state                                 (36) ,
event-type                              (37) ,
event-parameters                            (83), -- 从以前的版本重命名而来
exception-schedule                      (38) ,
fault-values                                (39) ,
feedback-value                          (40) ,
```

file-access-method (41) ,
file-size (42) ,
file-type (43) ,
firmware-revision (44) ,
high-limit (45) ,
inactive-text (46) ,
in-process (47) ,
instance-of (48) ,
integral -constant (49) ,
integral -constant-units (50) ,
issue-confirmed-notifications (51) ,
limit-enable (52) ,
list-of-group-members (53) ,
list-of-object-property-references (54) ,
list-of-session-keys (55) ,
local-date (56) ,
local -time (57) ,
location (58) ,
low-limit (59),
manipulated-variable-reference (60) ,
maximum-output (61) ,
max-apdu-length-accepted (62) ,
max-info-frames (63) ,
max-master (64) ,
max-pres-value (65) ,
minimum-of-time (66) ,
minimum-on-time (67) ,
minimum-output (68) ,
min-pres-value (69) ,
model-name (70),
modification-date (71),
notification-class (17), --从以前的版本重命名而来
notify-type (72),
number-of APDU-retries (73),
number-of states (74),

object-identifier (75),
object-list (76),
object-name (77),
object-property-reference (78),
object-type (79),
optional (80),
out-of-service (81),
output-units (82),
-- see event-parameters (83),
polarity (84),
present-value (85),
priority (86),
priority-array (87),
priority-for-writing (88),
process-identifier (89),
program-change (90),
program-location (91),
program-state (92),
proportional-constant (93),
proportional-constant-units (94),
protocol-conformance-class (95),
protocol-object-types-supported (96),
protocol-services-supported (97),
protocol-version (98),
read-only (99),
reason-for-halt (100),
recipient (101),
recipient-list (102),
reliability (103),
relinquish-default (104),
required (105),
resolution (106),
segmentation-supported (107),
setpoint (108),
setpoint-reference (109),

```
    state-text                          (110),
    status-flags                            (111),
    system-status                           (112),
    time-delay                          (113),
    time-of-active-time-reset               (114),
    time-of-state-count-reset               (115),
    time-synchronization-recipients             (116),
    units                               (117),
    update-interval                         (118),
    utc-offset                          (119),
    vendor-identifier                       (120),
    vendor-name                             (121),
    vt-classes-supported                    (122),
    weekly-schedule                         (123),
    …
    }
```

-- 特定的属性标识符，不论是所有、可选、还是必需，保留为**条件读属性**服务和**读多个属性**服务所用，或者为本标准没有定义的其它服务所用。

--枚举值 0-511 保留为 ASHRAE 用作定义中。枚举值 512-4194303 可以被其它机构根据 23 节中的规定使用。本版本使用的最高枚举值是 123。

```
BACnetPropertyReference  :: = SEQUENCE{
    proPertyIdentifier          [0] BACnetPropertyIdentifier,
    propertyArrayIndex      [1] Unsigned OPTIONAL  --只与数组数据类型一起使用
                                --如果数组中省略这个参数，表示引用整个数组
    }
```

```
BACnetPropertyStates  :: = CHOICE {
```

-- 本表述表示属性可能的数据类型，这些属性具有离散值或枚举值。这个选择必须与**事件登记**对象中引用属性的数据类型一致。

```
    boolean-value   [0] BOOLEAN,
    binary-value        [1] BACnetBinaryPV,
    event-type      [2] BACnetEventType,
    Polarity            [3] BACnetPolarity,
    program-change  [4] BACneProgramRquest,
```

```
    Program-state   [5] BACnetProgramState,
    reason-for-halt     [6] BACnetProgramErmr,
    reliability         [7] BACnetReliability,
    state           [8] BACnetEvenState,
    system-status     [9] BACnetDeviceStatus,
    units               [10] BACnetEngineeringUnits,
    …
    }
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
BACnetPropertyValue  :: == SEQUENCE {
    propertyIdenifier    [0] BACnetPropertyIdentifier,
    propertyArrayIndx  [1] Unsigned OPTIONAL, --只与数组数据类型一起使用
                            --如果数组中省略这个参数，表示引用整个数组
    value            [2] ABSTRACT-SYNTAX.&Type, -- 任何适合指定属性的数据类型
    priority             [3] Unsigned (1..16) OPTIONAL  --只与命令属性一起使用
    }
BACnetRecipient  :: = CHOICE {
    device           [0] BACnetObjectIdentifier,
    address              [1] BACnetAddress
    }

BACnetRecipientProess  :: = SEQUENCE {
    recipient          [0] BACnetRecipient,
    processIdentifier    [1] Unsigned
    }

BACnetReliability  :: = ENUMERATED {
    no-fault-detected    (0),
    no-sensor        (1),
    over-range       (2),
    under-range          (3),
    open-loop        (4),
    shorted-loop         (5),
```

```
    no-output        (6),
    unreliable-other     (7),
    process-error        (8),
    }
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
BACnetSegmentation  :: = ENUMERATED {
    segmented-both      (0),
    segmented-transmit    (1),
    segmented-receive   (2),
    no-segmentation         (3)
    }

BACnetServicesSupported  :: == BIT STRING {
--Alarm and Event Service
    acknledgeAlarm             (0),
    confirmedCOVNotifiotfon        (1),
    confirmedEventNotification     (2 ),
    getAlarmSummary            (3),
    getEnrollmentSummary           (4),
    subscribeCOV               (5),

-- File Access Services
    atomicReadFile             (6),
    atomicWriteFile                (7),

-- Object Access Services
    addListElement             (8),
    removeListElement          (9),
    createObject                   (10),
    deleteObject                   (11),
    readProperty                   (12),
    readPropertyConditional        (13),
    readPropertyMultiple           (14),
```

```
    writeProperty                   (15),
    writePropertyMultiple           (16),

-- Remote Device Management Services
    deviceCommunicationControl    (17),
    confirmedPrivateTransfer        (18),
    confirmedTextMessage            (19),
    reinitializeDevice              (20),

-- Virtual Terminal Services
    vtOpen                      (21),
    vtClose                         (22),
    vtData                      (23),

-- Security Services
    authenticate                    (24),
    requestKey                  (25),

-- Unconfirmed Services
    i-Am                        (26),
    i-Have                      (27),
    unconfirmedCOVNotification   (28),
    unconfirmedEventNotification   (29),
    unconfirmedPrivateTransfer     (30),
    unconfirmedTextMessage     (31),
    timeSynchronization             (32),
    who-Has                     (33),
    who-Is                      (34)
    }

BACnetSessionKey  :: = SEQUENCE {
    sessionKey  OCTET STRING (SIZE(8)), -- 56 比特作为密钥，8 比特作为校验和
    peerAddress     BACnetAddress
    }
```

```
BACnetSetpointReference  :: = SEQUENCE {
    setpointReference   [0] BACnetObjectPropertyReferene OPTIONAL
    }

BACnetSpecialEvent  :: = SEQUENCE {
    period  CHOICE {
                calendarEntry   [0] BACnetCalendarEnry,
                calendarReference   [1] BACnetObjectIdentifier
                },
    listOfTimeValues    [2] SEQUENCE OF BACnetTimeValue,
    evenPriority        [3] Unsigned (1.. 16)
    }
BACnetStatusFlags  :: = BIT STRING {
    in-alarm        (0),
    fault           (1),
    overridden      (2),
    out-of-service  (3)
    }
BACnetTimeStamp  :: = CHOICE {
    time            [0] Time,
    sequenceNumber  [1] Unsigned (0..65535),
    dateTime        [2] BACnetDateTime
    }

BACnetTimeValue  :: = SEQUENCE {
    time    Time,
    value   ABSTRACT-SYNTAX.&Type
    }

BACnetVTClass  :: = ENUMERATED {
    default-terminal    (0),
    ansi-x3-64      (1),
    dec-vt52            (2),
    dec-vt100       (3),
    dec-vt220       (4),
```

```
    hP-700-94        (5),
    ibm-3130         (6),
    ...
    }
```

--枚举值 0-63 保留为 ASHRAE 用作定义中。枚举值 64-65535 可以被其它机构根据 23 节中的规定使用。

```
BACnetVTSession  :: = SEQUENCE {
    local-vtSessionID       Unsigned8,
    remote-vtSessionID  Unsigned8,
    remote-vtAddress        BACnetAddress
    }

BACnetWeekNDay  :: = OCTET STRING (SIZE (3))
--  first octet     month (1..12)       January = 1,     X'FF' = any month
--  second octet    weekOfMonth         where:  1 = days numbered 1-7
--                                              2 = days numbered 8-14
--                                              3 = days numbered 15-21
--                                              4 = days numbered 22-28
--                                              5 = days numbered 29-31
--                                              6 = last 7 days of this month.
--                                              255 = any week of this month
--  third octet     dayOfWeek (1..7)          where : 1 = Monday
--                                              7 = Sunday
--                                              X 'FF' = any day of week

ReadAccessResult  :: = SEQUENCE {
    objectIdentifier    [0] BACnetObjectIdentifier,
    listOfResults       [1] SEQUENCE OF SEQUENCE {
            propertyIdentifier    [2] BACnetPropertyIdentifier,
            propertyArrayIndex  [3] Unsigned OPTIONAL, --只与数组数据类型一起使用
                                        --如果数组中省略这个参数，表示引用整个数组
            readResult  CHOICE {
                        propertyValue          [4]ABSTRACT-SYNTAX.&Type,
```

```
                        propertyAccessError     [5] Error
                        }
            } OPTIONAL
    }

ReadAccessSpecification  :: = SEQUENCE{
    objectIdentifier            [0] BACnetObjectIdentifier,
    listOfPropertyReferences    [1] SEQUENCE OF BACnetPropertyReference
    }

WriteAccessSpeification  :: = SEQUENCE {
    ObectIdenifier          [0] BACnetObjectIdentifier,
    listOfProperties            [1] SEQUENCE OF BACnePropertyValue
    }

END
```

22. 一致性及其规范

BACnet 定义了一套具有广泛适应性的对象类型和应用服务，其目的是要满足在分布式的、具有层次结构的楼宇自动控制系统中，各个层次的控制设备对数据通信的要求。这里需要说明的是，并不要求楼宇自动控制系统中所有设备都必须支持 BACnet 的所有功能才能实现 BACnet 数据通信。本节定义一套为了实现 BACnet 数据通信功能，设备必须满足的要求。本节定义了六个一致性类别和十三个“功能组”。这六个一致性类别编号为 1-6，具有分级结构，每一个类别的要求都包括了编号数小于该类别的所有类别的要求。功能组是若干 BACnet 标准对象类型和应用服务的集合，它们一起工作，可以满足一个单项的楼宇自动控制功能的通信要求。功能组的特性如下：

- 根据要实现的应用的要求，可以将功能组所规范的功能包含在任何一个一致性类别之中。
- 功能组所规范的功能必须能够被用户清楚地理解，而且使用户能够明确地确定其功能就是某个系统或者设备的特征要求。
- 功能组所规范的功能是能够在一个设备内实现的、满足任何一致性类别要求的增加的功能。

按照这种方式对 BACnet 对象和服务进行分类，给楼宇自动控制系统的设计者和使用者提供了一种方法，可以用来对多生产商的产品进行互操作的要求说明。

22.1 BACnet 一致性类别

本节定义六个 BACnet 协议的一致性类别，每一个类别的要求都用表格形式给出。对于应用服务来说，起始服务请求的能力的要求和执行服务请求的能力的要求是不同的。在表格中，用“x”标注在不同的列来说明这些要求。“x”在“起始”列内标注，说明设备能够起始服务请求，而“x”在“ 执行”列内标注，则说明设备能够执行服务请求。“对象”列包含一个标准对象列表，说明设备要满足这个一致性类别的要求所必须支持的标准对象。如果省略“对象”列，说明这个一致性类别不再专门要求增加对某个对象的支持。一致性类别的结构是一个分级结构，每一个类别的要求包含所有小于这个类别编号的类别的所有要求。

22.1.1 一致性类别 1

表 22-1 给出一致性类别 1 的要求。

表 22-1. 一致性类别 1 的要求

应用服务	起始	执行	对象
读属性服务		x	**设备**对象

22.1.2 一致性类别 2

表 22-2 给出一致性类别 2 的要求。

表 22-2. 一致性类别 2 的要求

应用服务	起始	执行
写属性服务		x

22.1.3 一致性类别 3

表 22-3 给出一致性类别 3 的要求。

表 22-3. 一致性类别 3 的要求

应用服务	起始	执行

I-Am 服务	x	x
I-Have 服务	x	
读多个属性服务		x
写多个属性服务		x
Who-Has 服务		x
Who-Is 服务		x

22.1.4 一致性类别 4

表 22-4 给出一致性类别 4 的要求。

表 22-4. 一致性类别 4 的要求

应用服务	起始	执行
添加列表元素服务	x	x
删除列表元素服务	x	x
读属性服务	x	
读多个属性服务	x	
写属性服务	x	
写多个属性服务	x	

22.1.5 一致性类别 5

表 22-5 给出一致性类别 5 的要求。

表 22-5. 一致性类别 5 的要求

应用服务	起始	执行
创建对象服务		x
删除对象服务		x
条件读属性服务		x
Who-Has 服务	x	
Who-Is 服务	x	

22.1.6 一致性类别 6

一致性类别 6 的 BACnet 设备要满足一致性类别 5 的所有要求，同时还要支持在 22.2 节中所定义的如下 5 个功能组：**时钟**功能组、PCWS 功能组、**事件起始**功能组、**事件响应**功能组、和**文件**功能组。

22.2 BACnet 功能组

功能组对 BACnet 对象和服务进行分类，每一个功能组定义一组应用服务和标准对象组合，实现 BACnet 对特定楼宇自动控制功能的通信要求的支持。BACnet 功能组表示可以添加到任何一致性类别的设备中的能力。功能组给楼宇自动控制系统的设计者和使用者提供了一种方法，可以用来对多生产商的产品进行互操作的要求说明和设计。参见附件 B。

对于包含在功能组中的应用服务来说，起始服务请求的能力的要求和执行服务请求的能力的要求是不同的。在表格中，用“x”标注在不同的列来说明这些要求。“x”在“起始”列内标注，说明设备能够起始服务请求，而“x”在“ 执行”列内标注，则说明设备能够执行服务请求。“对象”列包含一个标准对象列表，说明设备要满足这个功能组的要求所必须支持的标准对象。如果省略“对象”列，说明这个功能组不要求必须对某个对象的支持。

22.2.1 时钟功能组

这个功能组提供与时钟相关的能力。支持这个功能组要求能够执行实现**时间同步**服务的服务程序，同时支持**设备**对象的**本地时间**属性、**本地日期**属性、**时差**属性、和**夏令时状态**属性。支持这个功能组的设备还要具有测量时间过程的手段。

22.2.2 手提操作设备(HHWS)功能组

这个功能组提供与手提操作设备相关的能力。表 22-6 给出 HHWS 功能组的要求。

表 22-6. HHWS 功能组的要求

应用服务	起始	执行
Who-Is 服务	x	x
I-Am 服务	x	
读属性服务	x	
写属性服务	x	
重新初始化设备服务	x	
时间同步服务	x	

22.2.3 个人计算机工作站(PCWS)功能组

这个功能组提供与个人计算机相关的能力。表 22-7 给出 PCWS 功能组的要求。

表 22-7. PCWS 功能组的要求

应用服务	起始	执行
Who-Is 服务	x	x
I-Am 服务	x	
读属性服务	x	
条件读属性服务	x	
读多个属性服务	x	
写属性服务	x	
写多个属性服务	x	
重新初始化设备服务	x	
时间同步服务	x	
创建对象服务	x	

删除对象服务	x	
添加列表元素服务	x	
删除接表元素服务	x	
基本读文件服务	x	
基本写文件服务	x	

22.2.4 事件起始（Event Initiation）功能组

这个功能组提供定义报警和事件的能力、探测报警或者事件发生的能力、和起始通知报警或者事件发生的通知报文的能力。支持这个功能组的设备要求能够支持在 13.2 节中定义的内部报告，或者支持在 13.3 节中定义的算法改变报告，或者两者都支持。如果支持内部报告，则要求支持**通知类**对象和至少一个表 13-2 中所列的对象的内部报告能力。如果支持算法改变报告，则要求支持**事件登记**对象。表 22-8 给出事件起始功能组的其它要求。

表 22-8. 事件起始功能组的要求

应用服务	起始	执行
确认报警服务		x
有证实事件通知服务	x	
获得报警摘要服务		x
无证实事件通知服务	x	

22.2.5 事件响应（Event Resonse）功能组

这个功能组提供对于事件发生通知的响应能力。表 22-9 给出这个功能组的要求。

表 22-9. 事件响应功能组的要求

应用服务	起始	执行
确认报警服务	x	
有证实事件通知服务		x

22.2.6 COV 事件起始功能组

这个功能组提供起始关于 COV 事件发生的通知报文的能力。表 22-10 给出这个功能组的要求。

表 22-10. COV 事件起始功能组的要求

应用服务	起始	执行
预订 COV 服务		x
有证实 COV 通知服务	x	

22.2.7 COV 事件响应功能组

这个功能组提供预订和接收关于 COV 事件发生的通知报文的能力。表 22-11 给出这个功能组的要求。

表 22-11. COV 事件响应功能组的要求

应用服务	起始	执行
预订 COV 服务	x	
有证实 COV 通知服务		x

22.2.8 文件功能组

这个功能组提供读、写、上载、和下载文件的能力。表 22-12 给出这个功能组的要求。

表 22-12. 文件功能组的要求

应用服务	起始	执行	对象
基本读文件服务		x	**文件**对象
基本写文件服务		x	

22.2.9 重新初始化功能组

这个功能组表示对远程设备重新初始化的能力。这个功能组的要求是设备要支持对于**重新初始化设备**服务请求的执行。

22.2.10 虚拟操作员接口功能组

这个功能组提供一个虚拟终端会话的操作员端所需的能力。表 22-13 给出这个功能组的要求。

表 22-13.　虚拟操作员接口功能组的要求

应用服务	起始	执行
VT 开启服务	x	
VT 关闭服务	x	x
VT 数据服务	x	x

22.2.11 虚拟终端功能组

这个功能组提供一个虚拟终端会话的服务器端所需的能力。表 22-14 给出这个功能组的要求。

表 22-14.　虚拟终端功能组的要求

应用服务	起始	执行
VT 开启服务		x
VT 关闭服务	x	x
VT 数据服务	x	x

22.2.12 设备通信功能组

这个功能组提供使能和禁止对某个设备的通信的能力，主要用于诊断目的。支持这个功能组要求能够执行**设备通信控制**服务。

2.2.13 时间主站功能组

这个功能组提供自动起始时间同步服务的能力。支持这个功能组要求同时支持时钟功能组和设备对象的**时间同步容器**属性。

22.3 BACnet 路由器

BACnet 路由器支持在 6.4 节中定义的网络层协议报文。如果路由器不支持 PTP 数据链路层，则也不要求这个路由器支持 I-Could-Be-Router-To-Network 报文。路由器必须能够实现在 6.5 节中规范的对于本地和远程的收发报文进行路由的功能。一个路由器可以提供 BACnet 应用功能，也可以不提供 BACnet 应用功能。

22.4 协议实现一致性声明

协议实现一致性声明（PICS）是由设备制造商提供的书面文件，标出在这个设备中实现的 BACnet 规范中的功能。由于 BACnet PICS 是公共文件，任何对它感兴趣的组织都可使用它，因此，一个 BACnet PICS 至少要包含以下信息：

(a) 标明生产商和描述 BACnet 设备的基本信息。

(b) 标明设备的一致性类别。

(c) 标明设备支持的所有功能组。

(d) 标明设备支持的所有标准应用服务和专有应用服务，指明设备是能够起始服务请求，还是能够响应服务请求，或者两者都能够。

(e) 标明设备支持的所有标准对象和专有对象的列表。

(f) 对于每一个支持的对象，

1. 标明每一个支持的可选属性，
2. 标明那些属性可以使用 BACnet 服务写入，
3. 标明这些对象是否可以使用 BACnet 服务动态创建或者删除，
4. 标明对于属性数据取值范围的限制。

(g) 标明支持的数据链接层的选项。

(h) 标明是否支持分段请求。

(i) 标明是否支持分段响应。

附件 A 给出了 BACnet PICS 的正式样本。

22.5 对于 BACnet 的一致性

以下规定为了实现 BACnet 一致性所必须满足的要求。

22.5.1 最小的一致性要求

所有符合 BACnet 协议的设备必须实现在一致性类别 1 中定义的对象类型和服务，实现第 6 节中定义的应用于非路由设备的网络层报文和协议。

22.5.2 PICS

所有符合 BACnet 协议的设备必须具有一份标明该设备实现的所有 BACnet 功能的 PICS。这份 PICS 必须按照附件 A 的格式，给出 22.4 节中规定的所有信息。

22.5.3 一致性测试

为了符合 BACnet 协议，所有的设备都必须通过一致性测试，以证明设备正确实现了在 PICS 中指定的标准对象类型和服务。一致性测试包含正向测试和反向测试，正向测试是对

设备声明支持的每个对象类型和服务进行测试;反向测试是对于设备不支持的标准对象和服务测试设备的响应，以证明设备不会具有破坏行为。

22.5.4 数据链路层和物理层

为了符合 BACnet 协议,所有的设备必须支持第 7 节到第 11 节中定义的 5 种数据链接层中的一种，支持与这些数据链路层兼容的一种物理层。

22.5.5 关于与非标准数据链路层的兼容问题

在某些特别场合，为了与其它的网络设备进行互操作，可能需要一个 BACnet 设备支持本标准定义的 5 个数据链路层之外的某个数据链路层技术和物理层技术,这种设备只要满足 22.5.1 节到 22.5.3 节规定的准则，就称为具有非标准数据链路层的符合 BACnet 的设备。

这种设备可以使用与数据链路层和物理层不相同的非标准协议层传送标准的 BACnet LSDU，这个 LSDU 包含本标准定义的、按照第 20 节和第 6 节给出的准则编码的应用层和网络层信息。允许对 BACnet LSDU 进行分段操作。附件 H 给出了一个这样的例程，使用的是网际网协议和 Novell 互联网数据报协议。

23. 扩展 BACnet 容纳生产商专有信息

BACnet 协议的目的是为楼宇自动控制设备交换信息提供一种机制。为了实现互操作性，BACnet 定义了一组标准的数据结构，称之为对象。标准对象中包含的信息是在大多数楼宇系统中普遍使用的信息，同时 BACnet 也允许在设备中交换非标准化的信息。有 4 种扩展 BACnet 容纳非标准信息交换的方式：

(a) 生产商对于 BACnet 中使用的枚举值定义专有的扩展值。

(b) 生产商使用**专有转换**服务调用一个专有服务。

(c) 生产商对标准对象添加新的专有属性。

(d) 生产商定义新的专有对象。

在每一种情况中，BACnet 报文都隐含地引用生产商的标识符代码，指明生产商的枚举值、服务、属性、或者对象。生产商标识符代码由 ASHRAE 管理，对每一个生产商分配一个代码。ASHRAE 自己的**生产商标识符**代码是 0。可以通过读取**设备**对象的**生产商标识符**属性值获得某个设备的**生产商标识符**。从 ASHRAE 标准管理机构可以获得生产商标识符代码值列表。

23.1 扩展枚举值

在有一些实例中，生产商需要通过向一个枚举值中增加一些值来扩展 BACnet。实现的方法是在给定的枚举值中使用大于 BACnet 保留的值范围之外的值。表 23-1 给出了可以被扩展的枚举值和 BACnet 保留使用的取值范围。所有没有在表 23-1 中出现的枚举值都是不能被扩展的。

表 23-1. 可扩展的枚举值

枚举值名称	保留的取值范围	最大的可取值
错误类	0－63	65535
错误代码	0－255	65535
BACnet 中止原因	0－63	255
BACnet 设备状态	0－63	65535
BACnet 工程单位	0－255	65535
BACnet 事件状态	0－63	65535
BACnet 事件类型	0－63	65535
BACnet 对象类型	0－127	1023
BACnet 程序错误	0－63	65535
BACnet 属性标识符	0－511	4194303
BACnet 属性状态	0－63	65535
BACnet 可靠性	0－63	65535
BACnet 拒绝原因	0－63	255
BACnetVT 类	0－63	65535

23.2 使用专有传输服务调用非标准服务

BACnet 定义了一组适合楼宇自动控制系统的应用层服务。虽然这些标准服务已经具有广泛代表性，生产商仍然可以任意创建新的服务，同时尽可能地使用标准服务。

生产商可以向 BACnet 中添加专有服务，使用**专有传输**服务调用这些专有服务。BACnet 不对专有服务的服务类型和参数提出限制,但是要求使用**有证实专有传输**服务和**无证实专有传输**服务进行传送。用于处理这些 APDU 的协议机制的实现方法要与本标准规定的相同。

使用**专有传输**服务调用的专有应用层服务的格式必须遵守本标准的编码准则。如果使用**专有传输**服务，必须注意，差错 APDU 不能分段。实现方法必须保证差错 APDU 中的参数不能长于分段所要求的长度。

23.3 向标准对象添加专有属性

BACnet 定义了一组标准对象，每一个对象具有一组属性，可以使用 BACnet 服务对这些属性进行访问和操作。BACnet 允许生产商给对象添加属性，从而扩展对象的能力。专有属性可以得到与标准属性相同的 BACnet 服务支持，所以，也能够使用与标准属性相同的方法对专有属性进行访问和操作。

如果一个专有属性要成为可命令的属性,则需要向每一个这样的可命令属性提供附加的属性，执行标准属性中**优先级数组**属性和**释放缺省**属性的功能，还要应用 19 节中规范的优先级判断机制。

生产商可以通过修改设备中定义的对象的方法，向一个标准对象添加专有属性。在设备的**属性标识符**属性列表中，添加专有属性的**属性标识符**值，这些值必须去大于 512 的值。任何使用**属性标识符**作为参数的 BACnet 服务都可以使用这些专有属性的属性标识符。

专有属性标识符隐含地引用了所驻留设备中的**设备**对象的**生产商标识符**属性,这个机制解决了不同生产商使用相同的枚举值表示完全不同的属性的问题。

23.4 向 BACnet 中添加专有对象类型

为了能够容纳标准对象不能解决的楼宇自动控制应用，BACnet 允许生产商添加专有对象类型，同时尽可能地使用标准对象类型。为了增强可扩展性，BACnet 向专有对象提供与标准对象相同的支持。

23.4.1 专有对象类型枚举值

生产商可以通过使用扩展 **BACnet 对象类型**枚举值的方法来增加专有对象类型。专有对象类型的枚举值取大于 128 的**对象类型**值。任何使用**对象类型**属性值作为参数的 BACnet 服务都可以使用这些表示专有对象的**对象类型**值。

23.4.2 专有属性的数据类型

生产商的专有对象的属性值可以是标准属性数据类型，也可以是专有属性数据类型。只能使用 20.2.1.4 节中定义的应用数据类型构造专有数据类型。

23.4.3 在专有对象类型中要求的属性

非标准对象类型必须支持下列属性：

对象标识符

对象名称

对象类型

这些属性的功能必须与标准 BACnet 对象中的对应属性相同，这就意味着在 BACnet 设备中，**对象标识符**属性和**对象名称**属性是唯一的。对象名称字符串至少是一个字符，并且必须是可打印的字符。

23.5 在扩展 BACnet 中的限制

在扩展 BACnet 时，有下列限制：

(a) APDU 类型 8 到类型 15 保留给 ASHRAE 将来使用。

(b) 只能通过使用**有证实专有传输**服务和**无证实专有传输**服务的方法增加服务，即，**BACnet 有证实服务选择**枚举值和 **BACnet 无证实服务选择**枚举值不能被扩展。

24. 网络安全

本节为 BACnet 定义一个有限的安全体系结构。在 BACnet 中，网络安全是可选项。本协议的安全体系提供对等实体鉴别（authentication）、数据来源鉴别、操作员身份鉴别和数据保密性与完整性方面的安全机制，不提供诸如访问控制和非否定的通信安全机制，如果系统中需要这样的安全机制，可以使用专有的方法将它们加入到 BACnet 中。

24.1 安全体系结构

24.1.1 概述

在 BACnet 中，使用握手方式实现对等实体鉴别和数据来源鉴别。握手方式可以保证通信是建立在一个通信网络内的两个已知节点之间。对于下列两种情况，要使用握手方式：

(a) 客户设备需要鉴别一个有证实服务请求确实被声明的设备执行，使用这个机制，无证实服务请求不能被鉴别。

(b) 服务器设备需要鉴别一个有证实服务请求确实被声明的设备执行，使用这个机制，无证实服务请求不能被鉴别。

实现这个机制的方法是，检验使用一个公共**会话密钥（Session Key, SK）**在两节点之间通信捆绑的加密数据。**会话密钥**从一个称为密钥服务器的设备获得。

在 BACnet 中，使用鉴别用户密码的方式实现操作员身份鉴别。密码可以通过本地设备鉴别，也可以被跨网络鉴别。如果密码要被跨网络鉴别，则密钥服务器必须包含有主站密码列表。

在 BACnet 中，使用加密方式实现数据保密性和完整性。使用从密钥服务器中获得的公共**会话密钥**对数据进行加密。采用这种机制，广播报文或者多播报文不能被加密。

通过密钥服务器向需要安全的 BACnet 设备分配**会话密钥**。一个设备使用**请求密钥（RequestKey）**服务，向密钥服务器请求一个它与另一个设备之间的密钥。也可以在网络初始化时，或者在固定的时间间隔，由密钥服务器向设备传送密钥。密钥服务器向设备传送**会话密钥**的方法是，使用**添加列表元素**服务向**设备**对象的**会话密钥列表**属性写入**会话密钥**的值。发送到 BACnet 网络上的**会话密钥**将使用每个设备的唯一的**私有密钥（Private Key）**进行加密。在设备中，可以对**会话密钥**进行解密使用。每个**会话密钥**的有效期由生产商自行确定。

24.1.2 私有加密密钥

一个需要鉴别对等实体或者加密数据的 BACnet 设备（假设设备名称为 X），必须具有一个唯一的**私有密钥（PK_X）**。PK_X是一个 56 比特的**数据加密标准（DES）**的加密密钥。PK_X用于对**会话密钥**的解密。如何在 BACnet 中产生和分发公共密钥的方法，由生产商自行确定。

24.1.3 会话加密密钥

密钥服务器产生和分发所有的需要的**会话密钥**。为了做到这一点，密钥服务器必须拥有用于 BACnet 网络中的**会话密钥**的备份。**会话密钥**也是一个 56 比特的 DES 加密密钥。如何产生一个**会话密钥**的方法由生产商自行确定。使用**添加列表元素**服务分发**会话密钥**。

24.1.4 加密算法

使用 DES 算法实现数据的加密和解密。在**联邦信息处理标准（Federal Information Procissing Standard）**46-2（也称为 ANSI X3.92）中对 DES 进行了规范。为了防止数据或者加密密钥的损坏，在加密之前，要将 APDU 用随机数据填充到接收设备支持的最大长度。

24.2. 鉴别机制

24.2.1 获得会话密钥的过程

获得**会话密钥**的过程如下：

(a) 设备 A 要求鉴别，向密钥服务器发送一个**请求密钥**服务请求，指明对远程客户 B 进行鉴别。

(b) 密钥服务器使用 24.2.4 节中的报文起始鉴别过程中的 PK_A，鉴别**请求密钥**服务请求是由设备 A 起始的。

(c) 密钥服务器为设备 A 和设备 B 产生一个**会话密钥（SK_{AB}）**。

(d) 密钥服务器把 SK_{AB} 和设备 A 的 BACnet 地址与设备 B 的 PK_B 一起加密编码，然后使用**添加列表元素**服务请求把这个加密的信息发送给设备 B。

(e) 设备 B 接收到添加列表元素请求之后，使用 24.2.4 节中的报文起始鉴别过程中的 PK_B，鉴别密钥确实是密钥服务器发送的。

(f) 一旦得到鉴别，设备 B 解密 SK_{AB}，将它放置在自己的**设备**对象中。向密钥服务器返回一个对于**添加列表元素**请求的 ‘Result(+)’。

(g) 密钥服务器把 SK_{AB} 和设备 B 的 BACnet 地址与设备 A 的 PK_A 一起加密编码，然后使用**添加列表元素**服务请求把这个加密的信息发送给设备 A。

(h) 设备 A 接收到添加列表元素请求之后，使用 24.2.4 节中的报文起始鉴别过程中的 PK_A，鉴别密钥确实是密钥服务器发送的。

(i) 一旦得到鉴别，设备 A 解密 SK_{AB}，将它放置在自己的**设备**对象中。向密钥服务器返回一个对于**添加列表元素**请求的 ‘Result(+)’。

24.2.2 对等实体鉴别机制

对等实体鉴别的过程归纳如下：

(a) 一旦设备 A 接收到 SK_{AB}，就开始对设备 B 进行鉴别。

(b) 设备 A 产生一个**鉴别（Authenticate）**服务请求，并且发送给设备 B。这个 **BACnet 有证实请求** PDU 的**服务请求**部分与 SK_A 一起加密编码。这个 APDU 的头部和所有低层次的 PCI 都不被加密。

(c) 如果设备 B 能够对**鉴别**服务请求进行解密，则按照在**鉴别**服务过程中规定的方式将‘**伪随机数（Pseudo Random Number）**’修改成为‘**修改的随机数（Modified Random**

Number)’，使用 SK_A加密‘**修改的随机数**’，向设备 A 返回一个**复杂确认**报文，其中的**鉴别确认**部分包含加密的‘**修改的随机数**’。

(d) 如果设备 A 接收到包含有正确的‘**修改的随机数**’的复杂确认报文，则完成对设备 B 的鉴别。

24.2.3 报文执行鉴别

一个 BACnet 客户设备需要鉴别它发送的一个报文确实被正确的 BACnet 服务器所执行，在 BACnet 中，实现这个鉴别的方法是增强的对等鉴别机制，其中绑定了一个**事务处理状态机（TSM）**。这就是说，只能使用有证实服务请求完成报文执行鉴别。报文执行鉴别过程包括两个部分：获得一个 SK 和报文接收的鉴别。获得一个 SK 的过程已经在 24.2.1 节中进行了描述，以下是客户设备 A 鉴别服务器设备 B 正在执行某个报文的过程：

(a) 一旦客户设备 A 接收到 SK_{AB}，就开始对服务器设备 B 进行报文执行鉴别。

(b) 客户设备 A 产生一个**鉴别**服务请求，并且发送给服务器设备 B。这个报文的‘**期待的调用 ID（Expected InvokeID）**’参数中包含有么被鉴别的**有证实请求 APDU** 的调用 ID。这个 **BACnet 有证实请求** PDU 的**服务请求**部分与 SK_A一起加密编码。这个 APDU 的头部和所有低层次的 PCI 都不被加密。

(c) 客户设备 A 发送了**鉴别**服务请求之后，立即将要被鉴别的**有证实请求 APDU** 发送给服务器设备 B。这个**有证实请求 APDU** 的调用 ID 应该与**鉴别**服务请求中的‘**期待的调用 ID**’参数匹配。

(d) 如果服务器设备 B 能够对**鉴别**服务请求进行解密，则等待 30 秒时间，期待接收具有‘**期待的调用 ID**’参数的那个**有证实请求 APDU**。如果服务器 B 接收到那个 APDU，则按照在**鉴别**服务过程中规定的方式将‘**伪随机数**’修改成为‘**修改的随机数**’，使用 SK_A加密‘**修改的随机数**’，向客户设备 A 返回一个**复杂确认**报文，其中的**鉴别确认**部分包含加密的‘**修改的随机数**’。

(e) 如果客户设备 A 接收到包含有正确的‘**修改的随机数**’的复杂确认报文，则完成对服务器设备 B 的报文执行鉴别。

注：这个过程也可以被服务器用来对正确的客户是否接收到简单确认或者复杂确认进行鉴别。

24.2.4 报文起始鉴别

一个BACnet服务器设备需要鉴别它接收的一个报文确实是声明的BACnet客户设备起始发送的，在 BACnet 中，实现这个鉴别的方法是增强的对等鉴别机制，其中绑定了一个**事务处理状态机**。这就是说，只能使用有证实服务请求完成报文起始鉴别。如果报文起始鉴别的双方有一方涉及到密钥服务器，则使用一个**私有密钥 PK** 对信息进行加密合解密。如果报文起始鉴别的双方都不涉及到密钥服务器，则使用一个**会话密钥 SK** 对信息进行加密合解密。

以下是服务器设备 B 鉴别客户设备 A 已经起始某个报文的过程：

(a) 如果服务器设备 B 和客户设备 A 都不是密钥服务器，则在开始进行鉴别之前，必须获得一个 SK_{AB}。

(b) 服务器设备 B 产生一个**鉴别**服务请求，并且发送给客户设备 A。这个报文的‘**期待的调用 ID**’参数中包含有要被鉴别的**有证实请求 APDU** 的调用 ID。这个 **BACnet 有证实请求** PDU 的**服务请求**部分与 SK_A 一起加密编码。如果设备 A 或者设备 B 中有一个是密钥服务器，则与一个合适的 PK 一起加密编码。这个 APDU 的头部和所有低层次的 PCI 都不被加密。

(c) 如果客户设备 A 能够对**鉴别**服务请求进行解密，则搜寻自己的 TSM，查找具有‘**期待的调用 ID**’参数的**有证实请求 PDU**。如果搜寻到 TSM，则按照在**鉴别**服务过程中规定的方式将‘**伪随机数**’修改成为‘**修改的随机数**’，使用 SK_A 加密‘**修改的随机数**’，向设备 A 返回一个**复杂确认**报文，其中的**鉴别确认**部分包含加密的‘**修改的随机数**’。

(d) 如果服务器设备 B 接收到包含有正确的‘**修改的随机数**’的复杂确认报文，则完成对客户设备 A 的报文起始鉴别。

24.2.5 操作员身份鉴别

BACnet 提供允许操作员登录网络设备的方法。操作员身份鉴别的过程归纳如下：

(a) 操作员登录到客户设备 A，给出操作员名称和密码。对于名称和密码的唯一要求是它们必须是可打印字符。如果客户设备 A 有能力，就进行本地密码鉴别，否则，继续下列步骤。

(b) 客户设备 A 产生一个**鉴别**服务请求，并且发送给密钥服务器。这个**鉴别**请求报文包含有‘**操作员名称**’参数和‘**操作员密码**’参数。这个 **BACnet 有证实请求** PDU 的**服务请求**部分与客户设备 A 的 **PK** 一起加密编码。这个 APDU 的头部和所有低层次的 PCI 都不被加密。

(c) 如果密钥服务器能够对**鉴别**服务请求进行解密，则检验‘**操作员名称**’参数与‘**操作员密码**’参数的正确性。如果密钥服务器鉴别了操作员身份，则按照在**鉴别**服务过程中规定的方式将‘**伪随机数**’修改成为‘**修改的随机数**’，使用客户设备 A 的 **PK** 加密‘**修改的随机数**’，向客户设备 A 返回一个**复杂确认**报文，其中的**鉴别确认**部分包含加密的‘**修改的随机数**’。如果密钥服务器不能鉴别操作员身份，则返回一个‘**Result(-)**’，其中包含有使用客户设备 A 的 **PK** 加密编码的 **BACnet 差错 PDU** 的差错部分。

(d) 如果客户设备 A 接收到包含有正确的‘**修改的随机数**’的复杂确认报文，则完成对操作员身份的鉴别。

24.3 数据保密性机制

保证数据的保密性和完整性的过程由两部分组成：获得一个会话密钥 SK 和加密数据。获得一个 SK 的过程已经在 24.2.1 节中进行了描述，以下是加密数据的过程：

(a) 一旦客户设备 A 接收到 SK_{AB}，就开始进行设备 A 和设备 B 之间数据加密。

(b) 客户设备 A 使用**鉴别**服务请求开始与服务器设备 B 进行一个**加密会话(Enciphered Session)**。如果鉴别请求成功，继续进行步骤 3。否则，不能建立**加密会话**。

(c) 如果 A 需要向 B 发送数据，则 A 使用密钥 SK_{AB} 加密编码 APDU 的数据部分，然后将这个 APDU 发送给 B。这个 APDU 的固定部分和所有低层次的 PCI 都不被加密。

(d) 如果 B 接收到来自 A 的数据，则使用密钥 SK_{AB} 对 APDU 的数据部分进行解密。

(e) 如果 B 需要向 A 发送数据，则 B 使用密钥 SK_{AB} 加密编码 APDU 的数据部分，然后将这个 APDU 发送给 A。这个 APDU 的固定部分和所有低层次的 PCI 都不被加密。

(f) 终止加密会话的方法是，A 或者 B 中任何一个设备都可以发送一个**鉴别**服务请求，其中的‘**开始加密会话 (Start Enciphered Session)**’参数设置为 FALSE。

24.3.1 请求一个加密会话

请求一个加密会话的过程如下：

(a) 客户设备 A 产生一个**鉴别**服务请求，并且发送给服务器设备 B。这个**鉴别**请求报文包含有‘**开始加密会话**’参数，并且设置为 TRUE 。这个 **BACnet 有证实请求** PDU 的**服务请求**部分与 SK_{AB} 一起加密编码。这个 APDU 的头部和所有低层次的 PCI 都不被加密。

(b) B 在处理这个报文之前，使用 24.2.4 节描述的报文起始鉴别过程鉴别这个报文确实是 A 发送的。如果 B 能够对**鉴别**服务请求进行解密并且接受加密会话，则按照在**鉴别**服务过程中规定的方式将‘**伪随机数**’修改成为‘**修改的随机数**’，使用 SK_A 加密‘**修改的随机数**’，向客户设备 A 返回一个**复杂确认**报文，其中的**鉴别确认**部分包含加密的‘**修改的随机数**’。否则，则返回一个‘Result(-)’，其中包含有使用 SK_{AB} 加密编码的 **BACnet 差错 PDU** 的差错部分。

(c) 如果客户设备 A 接收到包含有正确的‘**修改的随机数**’的复杂确认报文，则开始一个加密会话。

24.3.2 结束一个加密会话

结束一个加密会话的过程如下：

(a) 设备 A 产生一个**鉴别**有证实请求，并且发送给设备 B。这个**鉴别**请求报文包含有‘**开始加密会话**’参数，并且设置为 FALSE。这个 **BACnet 有证实请求** PDU 的**服务请求**部分与 SK_{AB} 一起加密编码。这个 APDU 的头部和所有低层次的 PCI 都不被加密。

(b) B 在处理这个报文之前，使用 24.2.4 节描述的报文起始鉴别过程鉴别这个报文确实是 A 发送的。如果 B 能够对**鉴别**服务请求进行解密并且接受终止加密会话，则

按照在**鉴别**服务过程中规定的方式将‘**伪随机数**’修改成为‘**修改的随机数**’，使用 SK_A 加密‘**修改的随机数**’，向客户设备 A 返回一个**复杂确认**报文，其中的**鉴别确认**部分包含加密的‘**修改的随机数**’。否则，则返回一个‘Result(-)’，其中包含有使用 SK_{AB} 加密编码的 **BACnet 差错** PDU 的差错部分。

(c) 如果客户设备 A 接收到包含有正确的‘**修改的随机数**’的复杂确认报文，则结束一个加密会话。

24.4 请求密钥服务

一个客户端的 BACnet 用户使用**请求密钥（RequestKey）**服务请求在它自己与另一个 BACnet 用户之间设置一个**会话密钥 SK**。为了保证数据的完整性，这个 **BACnet 有证实请求** PDU 的**服务请求**部分与客户的 **PK** 一起加密编码。这个 APDU 的头部和所有低层次的 PCI 都不被加密。注：使用这个服务时，可以要求将起始设备的 **APDU 超时**属性值加大，从而能够容纳在密钥服务器客户两端的一个鉴别会话的时间要求。

24.4.1 结构

表 24-1 表示**请求密钥**服务原语的结构。

表 24-1　请求密钥服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
请求设备标识符(Requesting Device Identifier)	M	M(=)		
请求设备地址(Requesting Device Address)	M	M(=)		
远程设备标识符(Remote Device Identifier)	M	M(=)		
远程设备地址(Remote Device Identifier)	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

24.4.1.1 参数

这个参数为**请求密钥**有证实服务请求传送参数值。

24.4.1.1.1 请求设备标识符

这个参数是 **BACnet 对象标识符**类型，传送请求产生一个会话密钥 SK 的客户设备的标识符。因为这个参数被加密保护，所以它的完整性得到保证。

24.4.1.1.2 请求设备地址

这个参数是 BACnet **地址**类型，传送请求产生一个**会话密钥** SK 的客户设备的地址。因为这个参数被加密保护，所以它的完整性得到保证。

24.4.1.1.3 远程设备标识符

这个参数是 BACnet **对象标识符**类型，传送要接收一个**会话密钥** SK 的相关远程设备的标识符。因为这个参数被加密保护，所以它的完整性得到保证。

24.4.1.1.4 远程设备地址

这个参数是 BACnet **地址**类型，传送要接收一个**会话密钥** SK 的相关远程设备的地址。因为这个参数被加密保护，所以它的完整性得到保证。

24.4.1.2 Result(+)

参数‘Result(+)’指明服务请求已经成功。如何鉴别请求设备不接收‘Result(+)’的安全性策略由生产商自行确定。如果希望对这个报文进行鉴别，则使用报文执行鉴别。

24.4.1.3 Result(-)

参数‘Result(-)’指明服务请求完全失败。失败的原因在‘**错误类型**’参数中说明。为了保证数据的完整性，使用客户的 PK 对‘**错误类型**’参数进行加密保护。

24.4.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。**错误类**是 SECURITY，**错误代码**在 18.5 节中定义。

24.4.2 服务过程

服务提供者（密钥服务器）在处理请求之前，使用 24.2.4 节中的**报文起始鉴别**过程鉴别这个请求是否是请求设备起始的。如果解码成功以及鉴别了请求的合法性，则密钥服务器产生请求的**会话密钥** SK。然后，密钥服务器将 SK 分别传送给远程设备和请求设备，传送的方法是使用**添加列表元素**服务向**设备**对象的**会话密钥列表**属性写入 SK 的值。使用接收设备的 PK 对**会话密钥列表**属性值进行加密编码。如果这些成功，密钥服务器返回一个关于这个服务的‘Result(+)’参数。如果这些失败，密钥服务器返回一个关于这个服务的‘Result(-)’参数。使用请求设备的 PK 对‘Result(-)’参数进行加密编码。

24.5 鉴别服务

一个客户端的 BACnet 用户使用**鉴别（Authenticate）**服务对对等实体、起始设备、执行设备、和操作员身份进行鉴别。为了保证数据的完整性，对这个 **BACnet 有证实请求** PDU 的**服务请求**部分进行加密。如果这个鉴别过程是在客户设备与密钥服务器之间进行，则使用客户的 PK 对参数进行加密和解密。如果这个鉴别过程是在两个非密钥服务器设备之间进行，则使用相应的 SK 对参数进行加密和解密。

24.5.1 结构

表 24-2 表示**鉴别**服务原语的结构。

表 24-2　鉴别服务原语结构

参数名称	Req	Ind	Rsp	Cnf
参数(Argument)	M	M(=)		
伪随机数(Pseudo Random Number)	M	M(=)		
期待的调用 ID(Expected Invoke ID)	U	U(=)		
操作员名称(Operator Name)	U	U(=)		
操作员密码(Operator Password)	C	C(=)		
开始加密会话(Start Enciphered Session)	U	U(=)		
Result(+)			S	S(=)
修改的随机数(Modified Random Number)			M	M(=)
Result(-)			S	S(=)
错误类型(Error Type)			M	M(=)

24.5.1.1 参数

这个参数为**鉴别**服务请求传送参数值。

24.5.1.1.1 伪随机数

这个参数是 **32 位无符号整型**类型，是由服务请求者产生的一个伪随机数。这个参数为‘**Result(+)**’参数调用**修改的随机数**所用。

24.5.1.1.2 期待的调用 ID

这个参数是 **8 位无符号整型**类型，是一个可选的参数，用于执行**报文起始鉴别**过程和**报文执行鉴别**过程之中。它是要被鉴别的报文的‘**调用 Id**’参数。

24.5.1.1.3 操作员名称

这个参数是**字符串**类型，是一个条件参数，用于执行**操作员身份鉴别**过程之中。如果提供这个参数，它表示请求访问的操作员的名称。

24.5.1.1.4 操作员密码

这个参数是**字符串**类型，是一个条件参数，用于执行**操作员身份鉴别**过程之中。如果‘**操作员名称**’参数存在，则本参数必须存在。它表示请求访问的操作员的密码。本参数的长度不能大于 20 个字节。

24.5.1.1.5 开始加密会话

这个可选的参数是**布尔**类型，表示请求的 BACnet 用户是要开始（TRUE）一个加密会话，还是要结束（FALSE）一个加密会话。如果本参数不存在，则没有建立加密会话，或者已经结束了加密会话。

24.5.1.2 Result(+)

参数‘Result(+)’指明服务请求已经成功。如何鉴别请求设备不接收‘Result(+)’的安全性策略由生产商自行确定。如果希望对这个报文进行鉴别，则使用**报文执行鉴别**。‘Result(+)’参数返回在 24.5.2 中定义的**修改的随机数**。

24.5.1.3 Result(-)

参数‘Result(-)’指明服务请求完全失败。失败的原因在‘**错误类型**’参数中说明。为了保证数据的完整性，使用客户的 PK 对‘**错误类型**’参数进行加密保护。

24.5.1.3.1 错误类型

这个参数有两个部分：(1) ‘**错误类**’(2) ‘**错误代码**’。**错误类**是 SECURITY，**错误代码**在 18.5 节中定义。

24.5.2 服务过程

如果解码成功以及鉴别了请求的合法性之后，服务提供者执行下列 4 个任务之一：

(a) 报文执行鉴别，

(b) 报文起始鉴别，

(c) 操作员身份鉴别，

(d) 加密的会话确定。

如果鉴别成功，则服务提供者生成修改的随机数，生成的方法是将接收的伪随机数每一个字节的最高位和最低位互换，并且将这个数装入‘Result(+)’原语中返回。否则，发送

一个‘Result(-)’。在每一种情况中，都对 APDU 的参数进行加密。如果这个事务处理发生在客户设备与密钥服务器之间，则使用客户的 PK。如果这个事务处理发生在非密钥服务器设备之间，则使用相应的 SK 对参数进行加密和解密。

24.5.2.1 报文执行鉴别

报文执行鉴别过程包含下列的步骤：

(a) 解码鉴别的 APDU，开始关于‘**期待的调用 ID**’的事务处理状态机。

(b) 等待 30 秒钟时间，期待接收来自请求设备的具有‘**期待的调用 ID**’参数的另一个 APDU。

(c) 如果接收到具有‘**期待的调用 ID**’参数的 APDU，则生成一个‘Result(+)’参数，并且返回给请求设备。否则，生成一个‘Result(-)’参数。

24.5.2.2 报文起始鉴别

报文起始鉴别过程包含下列的步骤：

(a) 解码鉴别的 APDU，确定关于‘**期待的调用 ID**’的事务处理状态机是否存在关于请求设备的。

(b) 如果具有‘**期待的调用 ID**’参数的 APDU 是未完成的，则生成一个‘Result(+)’参数，并且返回给请求设备。否则，生成一个‘Result(-)’原语。

24.5.2.3 操作员身份鉴别

操作员身份鉴别过程包含下列的步骤：

(a) 解码鉴别的 APDU，确定是否有一个合法的操作员名称能够与 APDU 中的‘**操作员名称**’参数匹配。如果没有，则返回一个‘Result(-)’原语。

(b) 如果操作员名称存在，检验‘**操作员密码**’的合法性。如果密码合法，则生成一个‘Result(+)’原语，并且返回给请求设备。否则，返回一个‘Result(-)’原语。

24.5.2.4 加密的会话确定

加密的会话确定过程包含下列的步骤：

(a) 解码鉴别的 APDU，向请求设备发送一个**报文起始鉴别**请求。

(b) 如果请求设备对这个**报文起始鉴别**请求作出肯定的响应，则执行设备将确定是否接纳这个关于开始一个加密会话或者结束一个加密会话的请求。如果可以接纳这个请求，则发送一个‘Result(+)’原语。否则，发送一个‘Result(-)’原语。

25. 参考文献

ANSI/IEEE Standard 754(1985), *IEEE Standard for Binary Floating-Point Arithmetic*.

ANSI X3.4 (1977), *American National Standard Code for Information Interchange.*

ANSI X3.41 (1974), *American National Standard Code Extension Techniques for Use with the 7-bit Code Character Set of American National Standard Code for Information Interchange.*

ANSI X3.92 (1981), *American National Standard Data Encryption Algorithm.*

ATA/ANSI 878.1(1992), *ARCNET Local Area Network Standard.*

DDN Protocol Handbook, Volumes 1-3 NIC 50004, 50005, and 50006.

Echelon, *LonMark™ Layers 1-6 Interoperability Guidelines Version 3.2.*

Echelon, *LonTalk® Protocol Specification Version 3.0.*

EIA-232-E (1991), *Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange.*

EIA-458 (1983), *Standard for Electrical Characteristics of Generate and Receivers for Use in Balanced Digital Multipoint Systems.*

FIPS 46-2(1993), *Federal Information Processing Standard-Data Encryption Standard.*

ISO 7498(1984), *Information processing systems-Open Systems Interconnection-Basic Reference Model.*

ISO TR 8509 (1987), *Information processing systems-Open Systems Interconnection-service conventions.*

ISO 8649 (1988), *Information processing systems-Open Systems Interconnection-service definition for the Association Control Service Element.*

ISO 8802-2 (1989), *Information processing systems-Local area network-Part 2:Local link control.*

ISO/IEC 8802-3 (1993), *Information processing systems-Local area network-Part 3:*

Carrier sense multiple access with collision detection (CAMA/CD) access method and physical layer specifications.

ISO 8822 (1988), *Information processing systems-Open Systems Interconnection-Connection-oriented presentation service definition.*

ISO/IEC 8824 (1990), *Information technology-Open Systems Interconnection-Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 8825 (1990), *Information technology-Open Systems Interconnection-Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

ISO 9545 (1989), *Information technology-Open Systems Interconnection-Application Layer Structure (ALS).*

ISO/IEC 10646-1 (1993), *IT-Universal Multiple-Octet Coded Character Set (UCS)-Part 1:Architecture and Basic Multilingual Plane.*

JES C 6226 (1983), *Code of the Japanese Graphic Character Set for Information Interchange. Japan Institute for Standardization.*

参考文献资源

ANSI: American National Standards Institute, 11 W.42nd St., 13th Floor, New York, NY 10036.

DDN: Available from the Defense Data Network Information Center, SRI International, 333 Ravenswood Ave., Room E J291, Menlo Park, CA 94025.

Echelon: Echelon Corporation, 4015 Miranda Ave., Palo Alto, CA 94304

EIA: Electronics Industries Association, 2001 Eye St. N. W., Washington, D. C. 2006.

FIPS: National Institute of Standards and Technology, Gaithersburg, ND 20899.

IEEE: The Institute of Electrical and Electronics Engineers, Inc., 345 E.47th St., New York, NY 10017

ISO: Available from ANSI.

JIS: Available from ANSI.

附件 A — 协议实现一致性声明（规范）

（本附件是标准的一部分，要求实现。）

BACnet 协议实现一致性声明

生产商名称（Vendor Name）：_____

产品名称（Product Name）：_____

产品型号（Product Model Number）：_____

产品描述

所支持的 BACnet 一致性类别

类别 1	☐	类别 4	☐
类别 2	☐	类别 5	☐
类别 3	☐	类别 6	☐

所支持的 BACnet 功能组

时钟功能组	☐	文件功能组	☐
手提操作设备功能组	☐	重新初始化功能组	☐
个人计算机工作站功能组	☐	虚拟操作员接口功能组	☐
事件起始功能组	☐	虚拟终端功能组	☐
事件响应功能组	☐	设备通信功能组	☐
COV 事件起始功能组	☐	时间主站功能组	☐
COV 事件响应功能组	☐		

所支持的 BACnet 标准应用服务

应用服务	起始请求		执行请求	
确认报警服务	□		□	
有证实 COV 通告服务	□		□	
有证实事件通告服务	□		□	
获得报警摘要服务		□		□
获得登记摘要服务		□		□
预订 COV 服务	□		□	
无证实 COV 通告服务	□		□	
无证实事件通告服务		□		□
基本读文件服务	□		□	
基本写文件服务	□		□	
添加列表元素服务		□		□
删除列表元素服务	□		□	
创建对象服务		□		□
删除对象服务		□		□
读属性服务	□		□	
条件读属性服务		□		□
读多个属性服务	□		□	
写属性服务	□		□	
写多个属性服务	□		□	

所支持的 BACnet 标准应用服务（续）

应用服务	起始请求		执行请求	
设备通信控制服务		□		□
有证实专有传输服务	□		□	
无证实专有传输服务	□		□	
重新初始化设备服务	□		□	
有证实文本报文服务	□		□	
无证实文本报文服务	□		□	
时间同步服务		□		□
Who-Has 服务	□		□	
I-Have 服务	□		□	
Who-Is 服务	□		□	
I-Am 服务	□		□	
VT 开启服务	□		□	
VT 关闭服务	□		□	
VT 数据服务	□		□	
鉴别服务		□		□
请求密钥服务	□		□	

所支持的标准对象类型

对象类型	是否支持	是否可动态创建	是否可动态删除	可选的属性支持	可写的属性
模拟输入对象	☐	☐	☐	_____	
模拟输出对象	☐	☐	☐	_____	
模拟值对象	☐	☐	☐	_____	_____
二进制输入对象	☐	☐	☐	_____	_____
二进制输出对象	☐	☐	☐	_____	_____
二进制值对象	☐	☐	☐	_____	
日期表对象	☐	☐	☐	_____	_____
命令对象	☐	☐	☐	_____	
设备对象	Yes	N/A	N/A	_____	
事件登记对象	☐	☐	☐	_____	
文件对象	☐	☐	☐	_____	
组对象	☐	☐	☐	_____	_____
环对象	☐	☐	☐	_____	_____
多态输入对象	☐	☐	☐	_____	
多态输出对象	☐	☐	☐	_____	
通告类对象	☐	☐	☐	_____	
程序对象	☐	☐	☐	_____	
时间表对象	☐	☐	☐	_____	_____

数据链路层选项

☐ ISO 8802-3, 10BASE5 ☐ ARCNET, coax star ☐ MS/TP master, baud rate(s):

_____

☐ ISO 8802-3, 10BASE2 ☐ ARCNET, coax bus ☐ MS/TP slave, baud rate(s):

_____

☐ ISO 8802-3, 10BASET ☐ ARCNET, twisted pair star ☐ Point-To-Point, EIA232, baud rate(s): ____

☐ ISO 8802-3, Fiber ☐ ARCNET, twisted pair star ☐ Point-To-Point, modem, baud rate(s): ____

☐ ARCNET, fiber star ☐ LonTalk, medium: _____

☐ 其它

所支持的字符集

表明支持多种字符集并不意味着它们能被同时支持。

☐ ANSI X3.4 ☐ IBM™/Microsoft™ DBCS ☐ JIS C 6226

☐ ISO 10646 (ICS-4) ☐ ISO 10646 (UCS2) ☐ ISO 8859-1

特殊功能

分段请求支持 ☐ yes ☐ no 窗口尺寸 _____

分段响应支持 ☐ yes ☐ no 窗口尺寸 _____

路由器 ☐

描述支持的路由能力

_____

_____

_____

_____

属性值域的限定：

附件 B — 规范 BACnet 设备的指南（资料）

（本附件不是标准的一部分，仅仅作为资料使用）

在第 22.1 节中所定义的一致性类别和在第 22.2 节中所定义的功能组是要为那些设计、规范以及运行包含有 BACnet 设备的楼宇自动控制系统的人们提供一个有用的工具。这种分类的方法实际上是两个相互冲突的目标的折衷。一个目标是通过限制被支持的标准 BACnet 对象和服务的不同的组合，来提高系统的互操作性。另一个目标是避免对 BACnet 设备生产商进行不必要的限制，这种限制可能使得他们生产的设备具有用来满足一致性要求但在实际使用中却从不使用的 BACnet 功能。如果要求所有的 BACnet 设备支持完全相同的标准对象类型和应用服务的组合，当然可以达到最大的互操作性。另一方面，如果生产商拥有完全的自由，就不可避免地导致在设备所支持的具体的对象类型和应用服务方面存在广泛的差异，这样设备之间就只能实现部分的互操作。这种互操作性将被限制在只对所有设备都支持的应用服务和对象类型。

包含在一致性类别中的思想是组合那些完成特定功能所必需的 BACnet 协议部分。当为一个自动化系统设计或者规范 BACnet 设备时，一个合适的方法是指明满足应用需要的一致性类别和所要求的功能组。只要设备处于相同的一致性类别之中并且支持相同的功能组，那么设备就可以对一个给定的功能进行互操作。如果设备不处于相同的一致性类别之中，那么设备只能在低级别的一致性类别中和设备均支持的共同的功能组中进行互操作。许多设计良好的系统都含有代表一致性类别和功能组的混合的设备。

某个生产商可能想要制造的设备支持比某一个一致性类别（或功能组）有更多的功能，但是又不要达到高一级的一致性类别的要求。这可以在协议实现一致性声明（PICS）中确定，方法是将所支持的对象类型和应用服务的枚举值与设备所支持的一致性类别与功能组进行对比。对于系统设计者来说，下面这一点是可以保证的，只要两个设备都声明支持某个服务，并且这两个设备对于这个服务的起始和执行具有所要求的组合，那么这两台设备肯定可以对这个服务进行互操作。

附件 C — 对象类型结构形式描述（资料）

（本附件不是标准的一部分，仅仅作为资料使用）

```
ANALOG-INPUT :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name                 [77] CharacterString,
        object-type                 [79] BACnetObjectType,
        present-value               [85] REAL,
        description                 [28] CharacterSting OPTIONAL,
        device-type                 [31] CharacterSting OPTIONAL,
        status-flags                [111] BACnetStatusFlags,
        event-state                 [36] BACnetEventState,
        reliability                 [103] BACnetReliability OPTIONAL,
        out-of-service              [81] BOOLEAN,
        update-interval             [118] Unsigned OPTIONAL,
        units                       [117] BACnetEngineeringUnits,
        min-pres-value              [69] REAL OPTIONAL,
        max-pres-value              [65] REAL OPTIONAL,
        resolution                  [106] REAL OPTIONAL,
        cov-increment               [22] REAL OPTIONAL,
        time-delay                  [113] Unsigned OPTIONAL,
        notification-class          [17] Unsigned OPTIONAL,
        high-limit                  [45] REAL OPTIONAL,
        low-limit                   [59] REAL OPTIONAL,
        deadband                    [25] REAL OPTIONAL,
        limit-enable                [52] BACnetLimitEnable OPTIONAL,
        event-enable                [35] BACnetTransitionBits OPTIONAL,
        acked-transitions           [0] BACnetTransitionBits OPTIONAL,
        notify-type                 [72] BACnetNotifyType OPTIONAL
        }

ANALOG-OUTPUT :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name                 [77] CharacterString,
        object-type                 [79] BACnetObjectType,
```

```
        present-value               [85] REAL,
        description             [28] CharacterSting OPTIONAL,
        device-type             [31] CharacterSting OPTIONAL,
        status-flags                [111] BACnetStatusFlags,
        event-state             [36] BACnetEventState,
        reliability                 [103] BACnetReliability OPTIONAL,
        out-of-service              [81] BOOLEAN,
        units                       [117] BACnetEngineeringUnits,
        min-pres-value              [69] REAL OPTIONAL,
        max-pres-value          [65] REAL OPTIONAL,
        resolution                  [106] REAL OPTIONAL,
        priority-array              [87] BACnetPriorityArray,
        relinquish-default          [104] REAL,
        cov-increment               [22] REAL OPTIONAL,
        time-delay              [113] Unsigned OPTIONAL,
        notification-class          [17] Unsigned OPTIONAL,
        high-limit                  [45] REAL OPTIONAL,
        low-limit                   [59] REAL OPTIONAL,
        deadband                    [25] REAL OPTIONAL,
        limit-enable                [52] BACnetLimitEnable OPTIONAL,
        event-enable                [35] BACnetTransitionBits OPTIONAL,
        acked-transitions           [0] BACnetTransitionBits OPTIONAL,
        notify-type             [72] BACnetNotifyType OPTIONAL
        }

ANALOG-VALUE :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name             [77] CharacterString,
        object-type             [79] BACnetObjectType,
        present-value               [85] REAL,
        description             [28] CharacterSting OPTIONAL,
        status-flags                [111] BACnetStatusFlags,
        event-state             [36] BACnetEventState,
        reliability                 [103] BACnetReliability OPTIONAL,
        out-of-service              [81] BOOLEAN,
```

```
        units                       [117] BACnetEngineeringUnits,
        min-pres-value              [69] REAL OPTIONAL,
        max-pres-value          [65] REAL OPTIONAL,
        resolution                  [106] REAL OPTIONAL,
        priority-array              [87] BACnetPriorityArray,
        relinquish-default          [104] REAL,
        cov-increment               [22] REAL OPTIONAL,
        time-delay              [113] Unsigned OPTIONAL,
        notification-class          [17] Unsigned OPTIONAL,
        high-limit                  [45] REAL OPTIONAL,
        low-limit                   [59] REAL OPTIONAL,
        deadband                    [25] REAL OPTIONAL,
        limit-enable                [52] BACnetLimitEnable OPTIONAL,
        event-enable                [35] BACnetTransitionBits OPTIONAL,
        acked-transitions           [0] BACnetTransitionBits OPTIONAL,
        notify-type             [72] BACnetNotifyType OPTIONAL
        }

BINARY-INPUT :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name             [77] CharacterString,
        object-type             [79] BACnetObjectType,
        present-value               [85] REAL,
        description             [28] CharacterSting OPTIONAL,
        device-type             [31] CharacterSting OPTIONAL,
        status-flags                [111] BACnetStatusFlags,
        event-state             [36] BACnetEventState,
        reliability                 [103] BACnetReliability OPTIONAL,
        out-of-service              [81] BOOLEAN,
        polarity                    [84] BACnetPolarity,
        inactive-text               [46] CharacterString OPTIONAL,
        active-text             [4] CharacterString OPTIONAL,
        change-of-state-time        [16] BACnetDateTime OPTIONAL ,
        change-of-state-time        [15] Unsigned OPTIONAL ,
        time-of-state-count-reset       [115] BACnetDateTime OPTIONAL,
```

```
        elapsed-active-time         [33] Unsigned32 OPTIONAL,
        time-of-active-time-reset   [114] BACnetDateTime OPTIONAL,
        time-delay              [113] Unsigned OPTIONAL,
        notification-class          [17] Unsigned OPTIONAL,
        alarm-value             [6] BACnetBinaryPV OPTIONAL,
        event-enable                [35] BACnetTransitionBits OPTIONAL,
        acked-transitions           [0] BACnetTransitionBits OPTIONAL,
        notify-type             [72] BACnetNotifyType OPTIONAL
        }

BINARY-OUTPUT :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name             [77] CharacterString,
        object-type             [79] BACnetObjectType,
        present-value               [85] REAL,
        description             [28] CharacterSting OPTIONAL,
        device-type             [31] CharacterSting OPTIONAL,
        status-flags                [111] BACnetStatusFlags,
        event-state             [36] BACnetEventState,
        reliability                 [103] BACnetReliability OPTIONAL,
        out-of-service              [81] BOOLEAN,
        polarity                    [84] BACnetPolarity,
        inactive-text               [46] CharacterString OPTIONAL,
        active-text             [4] CharacterString OPTIONAL,
        change-of-state-time        [16] BACnetDateTime OPTIONAL ,
        change-of-state-time        [15] Unsigned OPTIONAL ,
        time-of-state-count-reset       [115] BACnetDateTime OPTIONAL,
        elapsed-active-time         [33] Unsigned32 OPTIONAL,
        time-of-active-time-reset   [114] BACnetDateTime OPTIONAL,
        minimum-off-time            [66] Unsigned32 OPTIONAL,
        minimum-on-time         [67] Unsigned32 OPTIONAL,
        priority-array              [87] BACnetPriorityArray,
        relinquish-default          [104] REAL,
        time-delay              [113] Unsigned OPTIONAL,
        notification-class          [17] Unsigned OPTIONAL,
```

feedback-value [40] BACnetBinaryPV OPTIONAL,
event-enable [35] BACnetTransitionBits OPTIONAL,
acked-transitions [0] BACnetTransitionBits OPTIONAL,
notify-type [72] BACnetNotifyType OPTIONAL
}

BINARY-VALUE :: = SEQUENCE {
object-identifier [75] BACnetObjectIdentifier,
object-name [77] CharacterString,
object-type [79] BACnetObjectType,
present-value [85] REAL,
description [28] CharacterSting OPTIONAL,
status-flags [111] BACnetStatusFlags,
event-state [36] BACnetEventState,
reliability [103] BACnetReliability OPTIONAL,
out-of-service [81] BOOLEAN,
inactive-text [46] CharacterString OPTIONAL,
active-text [4] CharacterString OPTIONAL,
change-of-state-time [16] BACnetDateTime OPTIONAL ,
change-of-state-time [15] Unsigned OPTIONAL ,
time-of-state-count-reset [115] BACnetDateTime OPTIONAL,
elapsed-active-time [33] Unsigned32 OPTIONAL,
time-of-active-time-reset [114] BACnetDateTime OPTIONAL,
minimum-off-time [66] Unsigned32 OPTIONAL,
minimum-on-time [67] Unsigned32 OPTIONAL,
priority-array [87] BACnetPriorityArray,
relinquish-default [104] REAL,
time-delay [113] Unsigned OPTIONAL,
notification-class [17] Unsigned OPTIONAL,
alarm-value [6] BACnetBinaryPV OPTIONAL,
event-enable [35] BACnetTransitionBits OPTIONAL,
acked-transitions [0] BACnetTransitionBits OPTIONAL,
notify-type [72] BACnetNotifyType OPTIONAL
}

```
CALENDAR :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name             [77] CharacterString,
        object-type             [79] BACnetObjectType,
        description             [28] CharacterSting OPTIONAL,
        present-value               [85] REAL,
        date-list                   [23] SEQUENCE OF BACnetCalendarEntry
        }

COMMAND :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name             [77] CharacterString,
        object-type             [79] BACnetObjectType,
        description             [28] CharacterSting OPTIONAL,
        present-value               [85] REAL,
        in-process              [47] BOOLEAN,
        all-writes-successful       [9] BOOLEAN,
        action                  [2] SEQUENCE OF BACnetActionList, -- accessed as a BACnetARRAY
        action-text             [3] SEQUENCE OF OF CharacterString OPTIONAL,
                                    -- 作为一个 BACnet 数组被访问
        }

DEVICE :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name             [77] CharacterString,
        object-type             [79] BACnetObjectType,
        system-status               [112] BACnetDeviceStatus,
        vendor-name             [121] CharacterString,
        vendor-identifier           [120] Unsigned16,
        model-name              [70] CharacterString,
        firmware-revision           [44] CharacterString,
        application-software-version    [12] CharacterString,
        location                    [58] CharacterString OPTIONAL,
        description             [28] CharacterSting OPTIONAL,
        protocol-version            [98] Unsigned,
```

```
        protocol-conformance-class      [95] Unsigned (1..6),
        protocol-services-supported         [97] BACnetServicesSupported,
        protocol-object-types-supported     [97] BACnetObjectTypesSupported,
        object-list                     [76] SEQUENCE OF BACnetObjectIdentifier,
                                            -- 作为一个 BACnet 数组被访问
        max-APDU-length-supported [62] Unsigned,
        segmentation-supported          [107] BACnetSegmentation,
        vt-classes-supported            [122] SEQUENCE OF BACnetVTClass OPTIONAL,
        active-vt-sessions              [5] SEQUENCE OF BACnetVTSession OPTIONAL,
        local-time                  [57] Time OPTIONAL,
        local-date                      [56] Date OPTIONAL,
        utc-offset                      [119] INTEGER OPTIONAL,
        daylight-savings-status         [24] BOOLEAN OPTIONAL,
        apdu-segment-timeout            [10] Unsigned,
        apdu-timeout                    [11] Unsigned,
        number-of-APDU-reteies  [73] Unsigned,
        list-of-APDU-retries            [55] SEQUENCE OF BACnetSessionKey OPTIONAL,
        time-synchronization-recipients [116] SEQUENCE OF BACnetRecipient OPTIONAL,
                                            -- 对于时间主站是必需的
        max-master                  [64] Unsigned(1..127) OPTIONAL,
                                            -- 对于 MS/TP 主站是必需的，参见 12.9 节
        max-info-frames             [63] Unsigned OPTIONAL,
                                            --对于 MS/TP 主站是必需的，参见 12.9 节
        device-address-binding          [30] SEQUENCE OF BACnetAddressBinding
        }

EVENT-ENROLLMENT :: = SEQUENCE {
        object-identifier               [75] BACnetObjectIdentifier,
        object-name                 [77] CharacterString,
        object-type                 [79] BACnetObjectType,
        description                 [28] CharacterSting OPTIONAL,
        event-type                  [37] BACnetEventType,
        notify-type                 [72] BACnetNotifyType,
        event-parameters                [83] BACnetEventParameter,
        object-property-reference   [78] BACnetObjectPropertyReference,
```

```
        event-state                 [36] BACnetEventState,
        event-enable                    [35] BACnetEventTransitionBits,
        acked-transitions               [0] BACnetEventTransitionBits,
        notification-class              [17] Unsigned OPTIONAL,
        recipient                       [101] BACnetRecipient OPTIONAL,
        process-identifier              [89] Unsigned OPTIONAL,
        priority                        [86] Unsigned OPTIONAL,
        issue-confirmed-notifications   [51] BOOLEAN OPTIONAL
        }

FILE :: = SEQUENCE {
        object-identifier               [75] BACnetObjectIdentifier,
        object-name                 [77] CharacterString,
        object-type                 [79] BACnetObjectType,
        description                 [28] CharacterSting OPTIONAL,
        file-type                       [43] CharacterString,
        file-size                       [42] Unsigned,
        modification-date               [71] BACnetDateTime,
        archive                     [13] BOLLEAN,
        read-only                       [99] BOLLEAN,
        file-access-method              [41] BACnetFileAccessMethod
        }

GROUP :: = SEQUENCE {
        object-identifier               [75] BACnetObjectIdentifier,
        object-name                 [77] CharacterString,
        object-type                 [79] BACnetObjectType,
        description                 [28] CharacterSting OPTIONAL,
        list-of-group-members           [53] SEQUENCE OF ReadAccessSpecification,
        present-value                   [85] SEQUENCE OF ReadAccessResult
        }

LOOP :: = SEQUENCE {
        object-identifier               [75] BACnetObjectIdentifier,
        object-name                 [77] CharacterString,
```

```
object-type                 [79] BACnetObjectType,
present-value                   [85] REAL,
description                 [28] CharacterSting OPTIONAL,
status-flags                    [111] BACnetStatusFlags,
event-state                 [36] BACnetEventState,
reliability                     [103] BACnetReliability OPTIONAL,
out-of-service                  [81] BOOLEAN,
update-interval                 [118] Unsigned OPTIONAL,
output-units                    [82] BACnetEngineeringUnits,
manipulated-variable-reference [60] BACnetObjectPropertyReference,
controlled-variable-reference  [19] BACnetObjectPropertyReference,
controlled-variable-value       [21] REAL,
controlled-variable-units       [20] BACnetEngineeringUnits,
setpoint-reference              [109] BACnetSetpointReference,
setpoint                    [108] REAL,
action                      [2] BACnetAction,
proportional-constant           [93] REAL OPTIONAL,
proportional-constant-units         [94] BACnetEngineeringUnits OPTIONAL,
integral-constant               [49] REAL OPTIONAL,
integral-constant -units        [50] BACnetEngineeringUnits OPTIONAL,
derivative-constant             [26] REAL OPTIONAL,
derivative -constant-units      [27] BACnetEngineeringUnits OPTIONAL,
bias                        [14] REAL OPTIONAL,
maximum-output              [61] REAL OPTIONAL,
minimum-output              [68] REAL OPTIONAL,
priority-for-writing            [88] Unsigned (1..16),
cov-increment               [22] REAL OPTIONAL,
time-delay                  [113] Unsigned OPTIONAL,
notification-class              [17] Unsigned OPTIONAL,
error-limit                     [34] Unsigned OPTIONAL,
event-enable                    [35] BACnetTransitionBits OPTIONAL,
acked-transitions               [0] BACnetTransitionBits OPTIONAL,
notify-type                 [72] BACnetNotifyType OPTIONAL
}
```

```
MULTI-STATE-INPUT :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name             [77] CharacterString,
        object-type             [79] BACnetObjectType,
        present-value               [85] REAL,
        description             [28] CharacterSting OPTIONAL,
        device-type             [31] CharacterSting OPTIONAL,
        status-flags                [111] BACnetStatusFlags,
        event-state             [36] BACnetEventState,
        reliability                 [103] BACnetReliability OPTIONAL,
        out-of-service              [81] BOOLEAN,
        number-of-states            [74] Unsigned,
        state-text                  [110] SEQUENCE OF CharacterString OPTIONAL,
                                    -- 作为一个 BACnet 数组被访问
        time-delay              [113] Unsigned OPTIONAL,
        notification-class          [17] Unsigned OPTIONAL,
        alarm-values                [7] SEQUENCE OF Unsigned OPTIONAL,
        fault-values                [39] SEQUENCE OF Unsigned OPTIONAL,
        event-enable                [35] BACnetTransitionBits OPTIONAL,
        acked-transitions           [0] BACnetTransitionBits OPTIONAL,
        notify-type             [72] BACnetNotifyType OPTIONAL
        }

MULTI-STATE-OUTPUT :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name             [77] CharacterString,
        object-type             [79] BACnetObjectType,
        present-value               [85] REAL,
        description             [28] CharacterSting OPTIONAL,
        device-type             [31] CharacterSting OPTIONAL,
        status-flags                [111] BACnetStatusFlags,
        event-state             [36] BACnetEventState,
        reliability                 [103] BACnetReliability OPTIONAL,
        out-of-service              [81] BOOLEAN,
        number-of-states            [74] Unsigned,
```

```
        state-text                  [110] SEQUENCE OF CharacterString OPTIONAL,
                                        -- 作为一个 BACnet 数组被访问
        priority-array              [87] BACnetPriorityArray,
        relinquish-default          [104] REAL,
        time-delay                  [113] Unsigned OPTIONAL,
        notification-class          [17] Unsigned OPTIONAL,
        feedback-value              [40] BACnetBinaryPV OPTIONAL,
        event-enable                [35] BACnetTransitionBits OPTIONAL,
        acked-transitions           [0] BACnetTransitionBits OPTIONAL,
        notify-type                 [72] BACnetNotifyType OPTIONAL
        }

NOTIFICATION-CLASS :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name                 [77] CharacterString,
        object-type                 [79] BACnetObjectType,
        present-value               [85] REAL,
        description                 [28] CharacterSting OPTIONAL,
        notification-class          [17] Unsigned OPTIONAL,
        priority                    [86] Unsigned OPTIONAL,
        ack-required                [1] BACnetEventTransitionBits,
        recipient-list              [102] SEQUENCE OF BACnetDestination
        }

PROGRAM :: = SEQUENCE {
        object-identifier           [75] BACnetObjectIdentifier,
        object-name                 [77] CharacterString,
        object-type                 [79] BACnetObjectType,
        program-state               [92] BACnetProgramState,
        program-change              [90] BACnetProgramRequest,
        reason-for-halt             [100] BACnetProgramError OPTIONAL,
        description-of-halt         [29] CharacterString OPTIONAL,
        program-location            [91] CharacterString OPTIONAL,
        description                 [28] CharacterSting OPTIONAL,
        instance-of                 [48] CharacterSting OPTIONAL,
```

status-flags [111] BACnetStatusFlags,
reliability [103] BACnetRwliability OPTIONAL,
out-of-service [81] BOOLEAN
}

SCHEDULE :: = SEQUENCE {
object-identifier [75] BACnetObjectIdentifier,
object-name [77] CharacterString,
object-type [79] BACnetObjectType,
present-value [85] REAL,
description [28] CharacterSting OPTIONAL,
effective-period [32] BACnetDateRange,
weekly-schedule [123] SEQUENCE SIZE(7) OF BACnetDailySchedule
OPTIONAL, -- 作为一个 BACnet 数组被访问
exception-schedule [38] SEQUENCE OF BACnetSpecialEvent OPTIONAL,
list-of-object-property-references [54] SEQUENCE OF
BACnetObjectPropertyReference,
priority-for-writing [88] Unsigned (1..16)
}

附件 D — 标准对象类型例程（资料）

（本附件不是标准的一部分，仅仅作为资料使用）

本附件提供 11 节所定义的 BACnet 标准对象类型的例程。

D.1 模拟输入对象例程

下面是用于空调控制器中的混合空气温度模拟输入对象的例程。对象支持属性值改变报告和内部报告。

Property: Object_Identifier = (Analog Input, Instance 1)
Property: Object_Name = “1AH1MAT”
Property: Object_Type = ANALOG_INPUT
Property: Present_Value = 58.1
Property: Description = “Mixed Air Temperature”
Property: Device_Type = “1000 OHM RTD”
Property: Status_Flag = {FALSE, FALSE, FALSE, FALSE}
Property: Event_State = NORMAL
Property: Reliability = NO_FAULT_DETECTED
Property: Out_of_Service = FALSE
Property: Update_Interval = 10
Property: Units = DEGREES-FAHRENHEIT
Property: Min_Pres_Value = -50.0
Property: Max_Pres_Value = 250.0
Property: Resolution = 0.1//分辨率??
Property: COV_Increment = 0.2
Property: Time_Delay = 10
Property: Notification_class = 3
Property: High_Limit = 60.0
Property: Low_Limit = 55.0
Property: Deadband = 1.0
Property: Limit_Enable = {TRUE, TURE}
Property: Event_Enable = {TRUE, FALSE, TRUE}
Property: Acked_Transitions = {TRUE, TRUE, TRUE}

Property: Notify_Type = EVENT

D.2 模拟输出对象例程

下面是用于空调控制器中风门模拟输出对象的例程。对象即不支持属性值改变报告，也不支持内部报告。

Property: Object_Identifier = (Analog Output, Instance 1)

Property: Object_Name = “1AH1DMPR”

Property: Object_Type = ANALOG_OUTPUT

Property: Present_Value = 75.0

Property: Description = “Damper Actuator”

Property: Device_Type = “3-8 PSI Actuator”

Property: Status_Flag = {FALSE, FALSE, FALSE, FALSE}

Property: Event_State = NORMAL

Property: Reliability = NO_FAULT_DETECTED

Property: Out_of_Service = FALSE

Property: Units = PERCENT

Property: Min_Pres_Value = 0.0

Property: Max_Pres_Value = 100.0

Property: Resolution = 0.1

Property: Priority_Array = {NULL, NULL, NULL, NULL, 75.0…NULL}

Property: Relinquish_Default = 50.0

D.3 模拟值对象例程

下面是用于热函计算的模拟值对象的例程。对象即不支持属性值改变报告，也不支持内部报告。

Property: Object_Identifier = (Analog Value, Instance 1)

Property: Object_Name = “1AH1ENTH”

Property: Object_Type = ANALOG_VALUE

Property: Present_Value = 38.1

Property: Description = “Enthalpy”

Property: Status_Flag = {FALSE, FALSE, FALSE, FALSE}

Property: Event_State = NORMAL

Property: Reliability = NO_FAULT_DETECTED

Property: Out_of_Service = FALSE

Property: Units = BTU-PER-POUND-DRY-AIR

D.4 二进制输入对象例程

例 1：典型二进制输入对象。

在本例中，二进制输入与一个常开静态高压开关相连。超出开关的静态压力极限，开关便会关闭。实际输入的**活动**状态表明开关关闭。对象支持属性值改变报告和内部报告。

Property: Object_Identifier = (Binary Input, Instance 1)

Property: Object_Name = “HighPressSwitch”

Property: Object_Type = BINARY_INPUT

Property: Present_Value = ACTIVE

Property: Description = “Penthouse Supply High Static”

Property: Device_Type = “ABC Pressure Switch”

Property: Status_Flag = {TRUE, FALSE, FALSE, FALSE}

Property: Event_State = OFFNORMAL

Property: Reliability = NO_FAULT_DETECTED

Property: Out_of_Service = FALSE

Property: Polarity = NORMAL

Property: Inactive_Text = “Static Pressure OK”

Property: Active_Text = “High Pressure Alarm”

Property: Change_of_State_Time = (23-MAR-1995, 19:01:34.0)

Property: Change_of_State_Count = 134

Property: Time_of_State_Count_Reset = (1-JAN-1995, 00:00:00.0)

Property: Elapsed_Active_Time = 401

Property: Time_of_Active_Time_Reset = (1-JAN-1995, 00:00:00.0)

Property: Time_Delay = 10

Property: Notification_class = 3

Property: Alarm_Value = ACTIVE

Property: Event_Enable = {TRUE, FALSE, TRUE}

Property: Acked_Transitions = {FALSE, TRUE, TRUE}

Property: Notify_Type = ALARM

例 2：脱离服务的二进制输入对象。

在这第二个例程中，上例中的控制系统发现了有断路发生。输入脱离服务。

Property: Object_Identifier = (Binary Input, Instance 1)

Property: Object_Name = “HighPressSwitch”

Property: Object_Type = BINARY_INPUT

Property: Present_Value = INACTIVE

Property: Description = “Penthouse Supply High Static”

Property: Device_Type = “ABC Pressure Switch”

Property: Status_Flag = {FALSE, TRUE, FALSE, TRUE}

Property: Event_State = NORMAL

Property: Reliability = OPEN_LOOP

Property: Out_of_Service = TRUE

Property: Polarity = NORMAL

Property: Inactive_Text = “Static Pressure OK”

Property: Active_Text = “High Pressure Alarm”

Property: Change_of_State_Time = (23-MAR-1995, 19:01:34.0)

Property: Change_of_State_Count = 135

Property: Time_of_State_Count_Reset = (1-JAN-1995, 00:00:00.0)

Property: Elapsed_Active_Time = 451

Property: Time_of_Active_Time_Reset = (1-JAN-1995, 00:00:00.0)

Property: Time_Delay = 10

Property: Notification_class = 3

Property: Alarm_Value = ACTIVE

Property: Event_Enable = {TRUE, FALSE, TRUE}

Property: Acked_Transitions = {TRUE, TRUE, TRUE}

Property: Notify_Type = ALARM

D.5 二进制输出对象例程

例 1：典型二进制输出对象。

在本例中，一个风扇由一个和常闭继电器相连的二进制输出控制。继电器不工作时，风扇将保持运行。二进制输出对象的**非活动**状态经极性属性反转，驱动继电器关闭风扇。注意：本例中**优先值数组**属性包含全空，因此**释放缺省**属性确定**当前值**属性。对象不支持内部报告。

Property: Object_Identifier = (Binary Output, Instance 1)

Property: Object_Name = “Floor3ExhaustFan”

Property: Object_Type = BINARY_OUTPUT

Property: Present_Value = INACTIVE

Property: Description = “Third floor bathroom exhaust fan”

Property: Device_Type = “ABC 100 Relay”

Property: Status_Flag = {FALSE, FALSE, FALSE, FALSE}

Property: Event_State = NORMAL

Property: Reliability = NO_FAULT_DETECTED

Property: Out_of_Service = FALSE

Property: Polarity = REVERSE

Property: Inactive_Text = “Fan is turned off”

Property: Active_Text = “Fan is running”

Property: Change_of_State_Time = (23-MAR-1995, 19:01:34.0)

Property: Change_of_State_Count = 47

Property: Time_of_State_Count_Reset = (1-JAN-1995, 00:00:00.0)

Property: Elapsed_Active_Time = 650

Property: Time_of_Active_Time_Reset = (1-JAN-1995, 00:00:00.0)

Property: Minimum_Off_Time = 100

Property: Minimum_On_Time = 10

Property: Priority_Array = {NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL,

NULL, NULL, NULL, NULL}

Property: Relinquish_Default = INACTIVE

例 2：被管制的二进制输出对象。

在这第二个例程中，风扇被发现是不可运行的。因为风扇在修复前是不能运行的，所以操作员管制了控制系统，以保持风扇处于非活动状态。

Property: Object_Identifier = (Binary Output, Instance 1)

Property: Object_Name = “Floor3ExhaustFan”

Property: Object_Type = BINARY_OUTPUT

Property: Present_Value = INACTIVE

Property: Description = “Third floor bathroom exhaust fan”

Property: Device_Type = “ABC 100 Relay”

Property: Status_Flag = {FALSE, TRUE, TRUE, FALSE}

Property: Event_State = NORMAL

Property: Reliability = OPEN_LOOP

Property: Out_of_Service = FALSE

Property: Polarity = REVERSE

Property: Inactive_Text = “Fan is turned off”

Property: Active_Text = “Fan is running”

Property: Change_of_State_Time = (23-MAR-1995, 19:01:34.0)

Property: Change_of_State_Count = 134

Property: Time_of_State_Count_Reset = (1-JAN-1995, 00:00:00.0)

Property: Elapsed_Active_Time = 401

Property: Time_of_Active_Time_Reset = (1-JAN-1995, 00:00:00.0)

Property: Minimum_Off_Time = 100

Property: Minimum_On_Time = 10

Property: Priority_Array = {NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL}

Property: Relinquish_Default = INACTIVE

D.6 二进制值对象例程

在本例中，二进制值是一种允许排气扇由操作员决定是否使能的机制。风扇控制逻辑利用这个值决定如果其它条件符合时风扇是否可以运行。例如，如果火灾警报发生或系统关闭，风扇可以不关闭。对象不支持内部报告。

Property: Object_Identifier = (Binary Value, Instance 1)
Property: Object_Name = “ExhaustFanEnable”
Property: Object_Type = BINARY_VALUE
Property: Present_Value = ACTIVE
Property: Description = “Exhaust Fan Operator Enable”
Property: Status_Flag = {FALSE, FALSE, FALSE, FALSE}
Property: Event_State = NORMAL
Property: Reliability = NO_FAULT_DETECTED
Property: Out_of_Service = FALSE
Property: Inactive_Text = “Enabled by Operator”
Property: Active_Text = “Fan Not Enabled by Operator”
Property: Change_of_State_Time = (23-MAR-1995, 19:01:34.0)
Property: Change_of_State_Count = 134
Property: Time_of_State_Count_Reset = (1-JAN-1995, 00:00:00.0)
Property: Elapsed_Active_Time = 401
Property: Time_of_Active_Time_Reset = (1-JAN-1995, 00:00:00.0)
Property: Minimum_Off_Time = 0
Property: Minimum_On_Time = 0
Property: Priority_Array = {NULL … NULL, ACTIVE}
Property: Relinquish_Default = INACTIVE

D.7 日期表对象例程

下面是为学校指定假日的日历对象的例程。

Property: Object_Identifier = (Calendar, Instance 1)

Property: Object_Name = “HOLIDAYS”

Property: Object_Type = CALENDAR

Property: Description = “1995 School District Holidays”

Property: Present_Value = TRUE

Property: DateList = ((19-FEB-1995), (28-MAY-1995), ((24-DEC-1995)-(4-JAN-1996)))

这是一个称为“假日”的日历。假日定于 2 月 19 日的总统日，5 月 28 日的阵亡将士纪念日和 12 月 24 日至次年 1 月 4 日的圣诞节假期。在这些日子中日历的当前值为真。其余的日子中当前值为假。当然，真正的校历在日期列表中会有更多的内容，这里为了简便起见只用了三条。

D.8 命令对象例程

例 1：有人和无人区

在本例中，办公楼的一个特殊区域有两个温度设置点：一个对应于无人时的操作，另一个对应于有人时的操作。在无人时期区域照明管制强行关闭照明。设置**命令**对象的**当前值**为 1 选择无人模式，选 2 则为有人模式。

假定对象：	Object_Identifier	Object_Name	Object_Type
	X’00800005’	ZONE43	ANALOG_VALUE
	X’01000003’	LIGHTING43	BINARY_OUTPUT

Property: Object_Identifier = (Command, Instance 1)

Property: Object_Name = “ZONE43CONTROL”

Property: Object_Type = COMMAND

Property: Description = “Fourth Floor, West Wing Office Suite”

Property: Present_Value = 1

Property: In_Process = FALSE

Property: All_Writes_Successful = TRUE

Property: Action =

{

(
(,(Analog Value,Instance 5), Present_Value,,65.0,,TRUE, TRUE),
(,(Binary Output, Instance 3), Present_Value,,INACTIVE, 8, 1,TRUE, TRUE)
),
(
(,(Analog Value, Instance 5), Present_Value,,72.0,,TRUE, TRUE),
(,(Binary Output, Instance 3), Present_Value,, ACTIVE, 8, 2, TRUE, TRUE)
)
}

Property: Action_Text = {“Unoccupied”, “Occupied”}

例 2：有人和无人区

本例在前一个例程的基础上增加了在包含**命令**对象的设备与控制电梯子系统的其它设备之间进行通信。当区域设置成为无人模式时，大楼该层的电梯服务锁住。在本例中，电梯子系统在 BACnet 设备“DDC4”中。

假定对象：	Object_Identifier	Object_Name	Object_Type
	X’00800005’	ZONE43	ANALOG_VALUE
	X’01000003’	LIGHTING43	BINARY_OUTPUT
	X’02000001’	DDC4	DEVICE
	X’01400001’	FL4	BINARY_VALUE

Property: Object_Identifier = (Command, Instance 1)
Property: Object_Name = “ZONE43CONTROL”
Property: Object_Type = COMMAND
Property: Description = “Fourth Floor, West Wing Office Suite”
Property: Present_Value = 2
Property: In_Process = FALSE
Property: All_Writes_Successful = TRUE
Property: Action =
{
(
(,(Analog Value, Instance 5), Present_Value,,65.0, 8, TRUE, TRUE),

```
    ( , (Binary Output, Instance 3), Present_Value,,INACTIVE, 8, 1,TRUE, TRUE)
),
(
    ( Device, Instance 1),(Binary Value, Instance 1), Present_Value,,INACTIVE,8 ,1,
        TRUE, TRUE)
),
(
    ( ,(Analog Value, Instance 5), Present_Value,,72.0, 8, 2, TRUE, TRUE),
    ( ,(Binary Output, Instance 3),Present_Value,, ACTIVE, 8,, TRUE, TRUE)
),
(
    (Device, Instance 1),(Binary Value,Instance 1), Present_Value,,ACTIVE,8 ,, TRUE,
        TRUE)
)
}
```

Property: Action_Text = {“Unoccupied”, “Occupied”}

D.9 设备对象例程

例 1:“复杂的”BACnet 设备。

```
Property:  Object_Identifier =          (Device, Instance 1)
Property:  Object_Name =            “AC1 System Controller”
Property:  Object_Type =               DEVICE
Property:  System_Status =          OPERATIONAL
Property:  Vendor_Name =            “ABC Controls”
Property:  Vendor_Identifier =          1001
Property:  Model_Name =             “1000 Plus”
Property:  Firmware_Revision =      “1.2”
Property:  Application_Software_Version =    “V4.0 - April 12, 1989”
Property:  Location =               “Basement Mechanical Room”
Property:  Description =                “AC1 Controller”
Property:  Protocol_Version =           1
Property:  Protocol_Conformance_Class = 6
Property:  Protocol_Service_Supported =     B’
```

11111111111111111111111
11111111111’

Property: Protocol_Object_Type_Supported = B’
11111111111111111’

Property: Object_List = ((Analog Input, Instance 1),
(Analog Input, Instance 2),…)

Property: Max_APDU_Length_Accepted = 400

Property: Segmentation_Supported = SEGMENTED_BOTH

Property: VT_Classes_Supported = (DEFAULT-TERMINAL, VT100)

Property: Active_VT_Sessions =
((29,5,1,X’0000C0B3EF13’),
(23,8,2,x’05’))

Property: Local_Time = “12:34:56.78”

Property: Local_Date = “29-SEP-1989”

Property: UTC_Offset = 6.0

Property: Daylight_Savings_Status = FALSE

Property: APDU_Segment_Timeout = 2,000

Property: APDU_Timeout = 60,000

Property: Number_of_APDU_Retries = 3

Property: List_Of_Session_Keys = ((X’3799246237984589’, 1, 3),
(X’4446214686489744’, 1, 5))

Property: Time_Synchronization_Recipient = (Device, Instance 18)

Property: Device_Address_Binding = (((Device, Instance 1), 1, 1),
(Device, Instance 12), 1, 23),
(Device, Instance 40), 2,
X’02608C41A606’),…)

例 2:“简单的”BACnet 设备

Property: Object_Identifier = (Device, Instance 2)

Property: Object_Name = “Room 101 VAV Controller”

Property: Object_Type = DEVICE

Property: System_Status = DOWNLOAD_REQUIRED
Property: Vendor_Name = “XYZ Controls”
Property: Vendor_Identifier = 1001
Property: Model_Name = “VAV 1000”
Property: Firmware_Revision = “1.0”
Property: Application_Software_Version = “2-1-88”
Property: Protocol_Version = 1
Property: Protocol_Conformance_Class = 2
Property: Protocol_Service_Supported = B’
11110100000010010000100000
111100011’
Property: Protocol_Object_Type_Supported = B’
110110110100010001’
Property: Object_List = ((Analog Input, Instance
1),
(Analog Input, Instance 2),
(Analog Output, Instance 1),
(Binary Input, Instance 1),
(Binary Output, Instance 1),
(Device, Instance 9))
Property: Max_APDU_Length_Accepted = 50
Property: Segmentation_Supported = NO_SEGMENTATION
Property: APDU_Segment_Timeout = 2,000
Property: APDU_Timeout = 60,000
Property: Number_of_APDU_Retries = 3
Property: Max_Master = 85
Property: Max_Info_Frames = 3
Property: Device_Address_Binding = (((Device, Instance 1),
1, 1),
(Device, Instance 12), 1, 23))

D.10 事件登记对象例程

在下面的例程中，假定了模拟输入对象。

ANALOG_INPUT

Property: Object_Identifier = (Analog Input, Instance 2)

Property: Object_Name = “Zone1_Temp”

Property: Object_Type = ANALOG_INPUT

Property: Present_Value = 86.0

Property: Description = “Receptionist Lobby Temp”

Property: Device_Type = “PT 3K RTD”

Property: Status_Flag = {FALSE, FALSE, FALSE, FALSE}

Property: Event_State = NORMAL

Property: Reliability = NO_FAULT_DETECTED

Property: Out_of_Service = FALSE

Property: Update_Interval = 5

Property: Units = DEGREES-FAHRENHEIT

Property: Min_Pres_Value = 55.0

Property: Max_Pres_Value = 95.0

Property: Resolution = 0.1

例 1：**偏离范围**事件类型。

这是一个**偏离范围**事件类型的例程。从**正常**到**高阈值**的状态转换没有被确认。接收设备的管理将使用**通告类**对象完成。

EVENT_ENROLLMENT

Property: Object_Identifier = (Event Enrollment, Instance 1)

Property: Object_Name = “Zone1_Alarm”

Property: Object_Type = EVENT_ENROLLMENT

Property: Description = “Zone1_Alarm”

Property: Event_Type = OUT_OF_RANGE

Property: Notify_Type = ALARM

Property: Event_Parameters = (30, 65.0, 85.0, 0.25)

Property: Object_Property_Reference = ((Analog Input, Instance 2), Present_Value)

Property: Event_State = HIGH_LIMIT
Property: Event_Enable = (TRUE, TRUE, TRUE)
Property: Acked_Transitions = (FALSE, TRUE, TRUE)
Property: Notification_Class = 1

例 2：**值改变**事件类型。

这是一个**值改变**事件类型的例程。接收设备的管理将使用**事件登记**对象的可选属性替代**通告类**对象完成。

EVENT_ENROLLMENT

Property: Object_Identifier = (Event Enrollment, Instance 2)
Property: Object_Name = “Zone1TempCOV”
Property: Object_Type = EVENT_ENROLLMENT
Property: Description = “Zone1 Temperature COV”
Property: Event_Type = CHANGE_OF_VALUE
Property: Notify_Type = EVENT
Property: Event_Parameters = (5, 0.25)
Property: Object_Property_Reference = ((Analog Input, Instance 2), Present_Value)
Property: Event_State = NORMAL
Property: Event_Enable = (TRUE, FALSE, FALSE)
Property: Acked_Transitions = (TRUE, TRUE, TRUE)
Property: Recipient = (Device, Instance 33)
Property: Process_Identifier = 4
Property: Priority = 87
Property: Issue_Comfirmed_Notification = TRUE

例 3：**比特位串改变**事件类型。

本例说明了**比特位串改变**事件的使用。

EVENT_ENROLLMENT

Property: Object_Identifier = (Event Enrollment, Instance 3)
Property: Object_Name = “Zone1 Rel”
Property: Object_Type = EVENT_ENROLLMENT
Property: Description = “Reliability alarm for zone 1 temperature”
Property: Event_Type = CHANGE_OF_BITSTRING
Property: Notify_Type = ALARM
Property: Event_Parameters = (30, B’0111’, (B’0100’, B’0010’, B’0001’, B’0110’, B’0101’, B’0011’))
Property: Object_Property_Reference = ((Analog Input, Instance 2), Status_Flags)
Property: Event_State = NORMAL
Property: Event_Enable = (TRUE, TRUE, FALSE)
Property: Acked_Transitions = (TRUE, TRUE, TRUE)
Property: Notification_Class = 3

D.11 文件对象例程

文件对象保存趋向信息：

Property: Object_Identifier = (File, Instance 7)
Property: Object_Name = “TREND_AI1”
Property: Object_Type = FILE
Property: Description = “Trend of AI1”
Property: File_Type = “TREND”
Property: File_Size = 45
Property: Modification_Date = (1-NOV-1995, 08:30:49.0)
Property: Archive = FALSE
Property: Read_Only = FALSE
Property: File_Access_Method = RECORD_ACCESS

D.12 组对象例程

下面是在楼宇的一部分用作参考温度的组对象的例程。

Property: Object_Identifier = (Group, Instance 1)
Property: Object_Name = “ZONE1_TEMPS”
Property: Object_Type = GROUP
Property: Description = “Zone 1 Temperature Group”
Property: List_of_Group_Menbers = (((Analog Input, Instance 8),
(present_Value,Reliability,Description)),
((Analog Input, Instance 9),
(present_Value,Reliability,Description)),
((Analog Input, Instance 10),
(present_Value,Reliability,Description)),
((Analog Input, Instance 11),
(present_Value,Reliability,Description)),
((Analog Input, Instance 12),
(present_Value,Reliability,Description)))
Property: Present_Value = (((Analog Input, Instance 8),Present_Value,
69.7, Reliability, NO_FAULT_DETECTED,
Description, “Room 1”),
((Analog Input, Instance 9),Present_Value,
71.2, Reliability, NO_FAULT_DETECTED,
Description, “Room 2”),
((Analog Input, Instance 10),Present_Value,
-50.0, Reliability, NO_FAULT_DETECTED,
Description, “Room 3”),

((Analog Input, Instance 11),Present_Value,
69.7, Reliability, NO_FAULT_DETECTED,
Description, “Room 4”),
((Analog Input, Instance 12),Present_Value,
73.3, Reliability, NO_FAULT_DETECTED,
Description, “Room 5”),

D.13 环对象例程

下面是用于空调控制器中的提供空气温度控制的**环**对象的例程。代表算法是如下的定位 PI 算法。

Output = {Proportional_Constant * [Error + (Integral_Constant * “Integral of the Error”)]} + Bias

Where: Error = (Process_Variable_Value - Setpoint) and “Integral of the Error” has units of **℉-min**

假定对象：

Object_Identifier	Object_Name	Object_Type
X’00400005’	AHU_VALUE	ANALOG_OUTPUT
X’00000003’	AHU_SAT	ANALOG_OUTPUT
X’00000007’	RESET_OAT	ANALOG_VALUE

这个对象支持 COV 报告。

Property: Object_Identifier = (Loop, Instance 1)
Property: Object_Name = “AHU_SAT_LOOP”
Property: Object_Type = LOOP
Property: Present_Value = 8.3
Property: Description = “Supply air temp. PI control”
Property: Status_Flag = {FALSE, FALSE, FALSE, FALSE}
Property: Event_State = NORMAL
Property: Reliability = NO_FAULT_DETECTED
Property: Out_of_Service = FALSE

Property: Update_Interval = 1

Property: Output_Units = POUNDS-FORCE-PER-SQUARE-INCH

Property: Manipulated_Variable_Reference = ((Analog Output, Instance 5), Present_Value)

Property: Controlled_Variable_Reference = ((Analog Input, Instance 3), Present_Value)

Property: Controlled_Variable_Value = 56.1

Property: Controlled_Variable_Units = DEGREES-FAHRENHEIT

Property: Setpoint_Reference = ((Analog Value, Instance 7), Present_Value)

Property: Setpoint = 57.0

Property: Action = DIRECT

Property: Proportional_Constant = 0.5

Property: Proportional_Constant_Units = PSI-PER-DEGREE-FAHRENHEIT

Property: Integral_Constant = 0.1

Property: Integral_Constant_Units = PER-MINUTE

Property: Derivative_Constant = 0.0

Property: Derivative_Constant_Units = NO-UNITS

Property: Bias = 9.0

Property: Maximum_Output = 15.0

Property: Minimum_Output = 3.0

Property: Priority_For_Writing = 10

Property: COV_Increment = 0.2

Property: Time_Delay = 3

Property: Notification_Class = 1

Property: Error_Limit = 5.0

Property: Event_Enable = {TRUE, TRUE, TRUE}

Property: Acked_Transitions = {TRUE, TRUE, TRUE}

Property: Notify_Type = ALARM

D.14 多态输入对象例程

例 1：双速风扇

在本例中输入＃1 同与低速启动器联系的“aux”连接，输入＃2 同与高速启动器联系的“aux”连接。如果低速启动器为“关”，那么高速启动器将禁止。下表显示了两个输入与当

前值之间的联系。用于确定和建立当前值的实际逻辑由生产商自行确定。这个对象支持内部报告。

Present_Value	Low-Speed Starter	High-Speed Starter
1	OFF (inactive)	OFF (inactive)
2	ON (active)	OFF (inactive)
3	ON (active)	ON (active)

Table B-1

Property: Object_Identifier = (Multi-state Input, Instance 1)
Property: Object_Name = “Fan1_Input”
Property: Object_Type = MULTISTATE_INPUT
Property: Present_Value = 2
Property: Description = “2-speed Fan#1”
Property: Status_Flags = {FALSE, FALSE, FALSE, FALSE}
Property: Event_State = NORMAL
Property: Reliability = NO_FAULT_DETECTED
Property: Out_of_Service = FALSE
Property: Number_of_States = 3
Property: State_Text = (“Off”, “On_Low”, “On_High”)
Property: Time_Delay = 3
Property: Notification_Class = 4
Property: Alarm_Values = 3
Property: Fault_Values = 2
Property: Event_Enable = {TRUE, TRUE, TRUE}
Property: Acked_Transitions = {TRUE, TRUE, TRUE}
Property: Notify_Type = EVENT

例 2：Hand-off-auto 开关。

在本例中 three-position hand-off-auto 开关的状态作为二进制输入被监视着。如果第一个输入是有效的，则当前值被设为 1（手动）；如果第二个输入是有效的，当前值则设

为 2（关）；如果第三个输入是有效的，则当前值为 3（自动）。对象不支持内部报告。

Property: Object_Identifier = (Multi-state Input, Instance 2)
Property: Object_Name = “H-O-A”
Property: Object_Type = MULTISTATE_INPUT
Property: Present_Value = 1
Property: Description = “Hand-Off-Auto 1”
Property: Device_Type = “ZZZ switch”
Property: Status_Flags = {FALSE, FALSE, FALSE, FALSE}
Property: Event_State = NORMAL
Property: Reliability = NO_FAULT_DETECTED
Property: Out_of_Service = FALSE
Property: Number_of_States = 3
Property: State_Text = (“Hand”, “Off”, “Auto”)

D.15 多态输出对象例程

例 1：双速风扇。

在本例中输出＃1 同低速启动器的线圈连接，输出＃2 同高速启动器的线圈”连接。如果低速启动器为“关”，那么高速启动器将禁止。下表显示了两个输出与当前值之间的联系。用于确定和建立当前值的实际逻辑由生产商自行确定。这个对象支持内部报告。

Present Value	Low-Speed Starter	High-Speed Starter
1	OFF (inactive)	OFF (inactive)
2	ON (active)	OFF (inactive)
3	ON (active)	ON (active)

Table B-2

Property: Object_Identifier = (Multi-state Output, Instance 1)
Property: Object_Name = “Fan1_Output”
Property: Object_Type = MULTISTATE_OUTPUT
Property: Present_Value = 2

Property: Description = “2-speed Fan#1”
Property: Device_Type = “ABC Fan Model A-6”
Property: Status_Flags = {FALSE, FALSE, FALSE, FALSE}
Property: Event_State = OFFNORMAL
Property: Reliability = NO_FAULT_DETECTED
Property: Out_of_Service = FALSE
Property: Number_of_States = 3
Property: State_Text = {“Off”, “On_Low”, “On_High”}
Property: Priority_Array = {NULL, NULL…2…NULL}
Property: Relinquish_Default = 1
Property: Time_Delay = 3
Property: Notification_Class = 4
Property: Feedback_Value = 3
Property: Event_Enable = {TRUE, TRUE, TRUE}
Property: Acked_Transitions = {TRUE, TRUE, TRUE}
Property: Notify_Type = EVENT

例 2：Three-position 开关。

在本例中 three-position 开关的状态作为二进制输出被控制着。如果当前值为 1，则第一个输出是活动的。如果当前值为 2，则第二个输出是活动的。如果当前值为 3，则第三个输出是活动的。这个对象不支持内部报告。

Property: Object_Identifier = (Multi-state Output, Instance 2)
Property: Object_Name = “3-POS-SW”
Property: Object_Type = MULTISTATE_OUTPUT
Property: Present_Value = 1
Property: Description = “3 POSITION SWITCH # 1”
Property: Status_Flags = {FALSE, FALSE, FALSE, FALSE}
Property: Event_State = NORMAL
Property: Reliability = NO_FAULT_DETECTED
Property: Out_of_Service = FALSE

Property: Number_of_States = 3

Property: State_Text = {“Position_1”, “Position_2”, “Position_3”}

Property: Priority_Array = {NULL…NULL}

Property: Relinquish_Default = 1

D.16 通告类对象例程

下面是用于包含一个与时间有关的接收列表的关键性系统警报的**通告类**对象的例程。

Property: Object_Identifier = (Notification Class, Instance 1)

Property: Object_Name = “Alarms1”

Property: Object_Type = NOTIFICATION_CLASS

Property: Description = “Critical System Alarms”

Property: Notification_Class = 1

Property: Priority = (3, 10, 10)

Property: Ack_Required = (TRUE, TRUE, TURE)

Property: Recipient_List = (((Monday, Tuesday, Wednesday,
Thursday, Friday), 6:00, 20:00,
(Device, Instance 12), 21, TRUE,
(TRUE, TRUE, TRUE)),
((Monday, Tuesday, Wednesday,
Thursday, Friday, Saturday, Sunday),
0:00,
6:00, (Device, Instance 18), 5,
TRUE, (TRUE, TRUE, FALSE)),
((Monday, Tuesday, Wednesday,
Thursday, Friday, Saturday, Sunday),
20:00, 24:00, (Device, Instance 18),

5,
TRUE, (TRUE, TRUE, FALSE))

D.17 程序对象例程

下面的部分将提供几个应用程序,这些程序使用**程序**对象作为它们网络可见的活动表现形式以及程序特定参数形式。例程中的生产商指定属性纯粹用于说明的目的，并不是本标准的要求。特别地，在每个例程中利用助记名说明属性标识符的使用。这些属性的实际实现将使用每个生产商自定的数值代码，来代表标准中没有定义的属性。

例 1：平均程序。

在本例中，应用程序使用三个属性 Value1，Value2，Value3 作为输入，并且得出如公式表达的三输入平均的输出：

Average = (Value1+Value2+Value3)/3.

程序对象已经通过增加四个生产商自定义的属性 Value1，Value2，Value3 和 Average 进行了扩展。

Property: Object_Identifier = (Program, Instance 1)
Property: Object_Name = “SomeAverage”
Property: Object_Type = PROGRAM
Property: Program_State = RUNNING
Property: Program_Change = READY
Property: Reason_For_Halt = NORMAL
Property: Description_Of_Halt = “Normal”
Property: Program_Location = “Line 2”
Property: Description = “Average of Somethings”
Property: Instance_Of = “ThreeWayAverager”
Property: Status_Flags = {FALSE, FALSE, FALSE, FALSE}
Property: Reliability = NO_FAULT_DETECTED
Property: Out_Of_Service = FALSE
Property*: Value 1 = 10
Property*: Value 2 = 22
Property*: Value 3 = 28

Property*: Average = 20

注：上面带星号“”的属性被称为生产商自定义属性。它们在这里纯粹用于说明的目的，不是本标准的要求。*

例 2：最大流量程序。

在本例中，应用程序使用三个属性 Ref1，Ref2，Ref3 作**为 BACnet 属性引用**属性，这是用于指向作为输入和得到如公式表达的三输入最大值输出的其它三个对象属性的：

Max = Maximum_Of(Ref1, Ref2, Ref3).

程序对象已经通过增加四个生产商自定义的属性 Ref1，Ref2，Ref3 和 Max 进行了扩展。

Property: Object_Identifier = (Program, Instance 2)
Property: Object_Name = “MaxFlow”
Property: Object_Type = PROGRAM
Property: Program_State = RUNNING
Property: Program_Change = READY
Property: Reason_For_Halt = NORMAL
Property: Description_Of_Halt = “Normal”
Property: Program_Location = “Line 5”
Property: Description = “Maximum of three Flows”
Property: Instance_Of = “Max3Refs”
Property: Status_Flags = {FALSE, FALSE, FALSE, FALSE}
Property: Reliability = NO_FAULT_DETECTED
Property: Out_Of_Service = FALSE
Property: Ref1 = ((Analog Input, Instance 272), Present_Value)
Property: Ref2 = ((Analog Input, Instance 273), Present_Value)
Property: Ref3 = ((Analog Input, Instance 274), Present_Value)
Property: Max = 45.6

注：上面带星号“”的属性被称为生产商自定义属性。它们在这里纯粹用于说明的目的，不是本标准的要求。*

D.18 时间表对象例程

下面是用于在学年中控制教室的时间表对象的例程。在本例中，假定定义一个不同的时间表对象用于年历中剩余的部分。时间表的引用属性是一个**二进制输出**对象的**当前值**属性，这个**二进制输出**对象控制为 208 房间提供空调的房顶单元。

```
    Property:          Object_Identifier =          (Schedule, Instance 2)
    Property:          Object_Name =            “Rm208Sched”
    Property:          Object_Type =              SCHEDULE
    Property:          Present_Value =          ACTIVE
    Property:          Description =                “Room 208 Schedule”
    Property:          Effective_Period =
    ((5-SEP-1995)-(10-JUN-1996))
    Property:          Weekly_Schedule =            {((8:00,          ACTIVE),
(17:00,INACTIVE)),
                                                    ((8:00, ACTIVE)),
                                                    ((8:00, ACTIVE), (17:00,INACTIVE)),
                                                    ((8:00, ACTIVE), (17:00,INACTIVE),
                                                     (19:00, ACTIVE), (23:30, INACTIVE)),
                                                    ((8:00, ACTIVE), (17:00,INACTIVE),
                                                    ((00:00, INACTIVE)),
                                                    ((10:00, ACTIVE), (17:00,INACTIVE))}
    Property:          Exception_Schedule =         {((23-NOV-1995),
(0:00,INACTIVE), 10),
                                                    ((HOLIDAYS, (0:00,INACTIVE), 11),
                                                    ((5-MAR-1996) - (7-MAR-1996),
                                                    ((9:00, ACTIVE), (14:00,INACTIVE)),
                                                6)}
    Property:          List_Of_Object_Property_References = ((Binary Output, Instance 9),
Present_Value)
    Property:          Priority_For_Writing =       15
```

在一般情况下，星期一、星期三和星期五为房间提供空调的房顶部分在上午 8 点激活，

在下午 5 点关闭。一般星期二房顶部分在上午 8 点激活，另外不被时间表影响。一般星期四房顶部分在上午 8 点激活，在下午 5 点关闭，在下午 7 点再次激活，在夜晚 11:30 关闭。

1995 年 11 月 23 日，星期五，星期五一般时间表被例外时间表（Exception_Schedule）列表中的第一条特殊事件（SPECIALEVENT）覆盖，房顶部分将整天关闭。

1996 年 2 月 19 日，星期一，根据 D.7 中所述的假日（HOLIDAYS）日历，星期一一般时间表被例外时间表列表中的第二条特殊事件覆盖。因为 1996 年 2 月 19 日，总统日，是在假日对象的日期列表中。房顶部分将整天关闭。

1996 年 3 月 5 日到 1996 年 3 月 7 日，因为教师协商会的原因，一般时间表被第三条特殊事件覆盖。在这三天中，房顶部分将在上午 9 点激活，在下午 2 点关闭。

附件 E — BACnet 应用服务的例程（资料）

（本附件不是标准的一部分，仅仅作为资料使用）

本附件提供 13-17 节所定义的应用服务的使用例程。在这些例程中，属性名称使用混合大小写字母表示，具有字符串数据类型的属性值封装在双引号中，枚举类型的值使用全大写字母表示。例程中不包含参数的编码，只有参数的未编码的符号值。例程中的编码见附件 F。

E.1 报警和事件服务

E.1.1 确认报警服务例程

参见 E.1.3 的与**有证实事件通告**服务协同的**确认报警**服务和 E.1.8 的与**无证实事件通告**服务协同的**确认报警**服务。

E.1.2 有证实 COV 通告服务例程

下面的例程说明了被监视的**模拟输入**对象当前值改变的通告。

```
Service                         = ConfirmedCOVNotification
     'Subscriber Process Identifier'     = 18
     'Initiating Device Identifier'          = (Device, Instance 4)
     'Monitored Object Identifier'       = (Analog Input, Instance 10)
     'Time Remaining'                = 0
 'List of Values'                = ((Present_Value,  65.0),  (Status_Flags,
(FALSE,
                                        FALSE, FALSE, FALSE,)))
```

E.1.3 有证实事件通告服务例程

Assumed objects:	Object Identifier	Object Name	Object Type
	(Analog Input, Instance 2)	Zone 1_Temp	ANALOG_INPUT

考虑一个名为“Zone 1_Temp”的支持内部事件报告的**模拟输入**对象，假定在区域一中温度刚升至超过高阈值，导致报警进入活动状态，在**确认转换**属性中的**进入异常**比特位被清除。进一步假定没有与之相联系的报文文本。设备将发出一个带有如下参数的**有证实事件通告**服务请求原语：

Service = ConfirmedEventNotification
‘Process Identifier’ = 1
‘Initiating Device Identifier’ = (Device, Instance 4)
‘Event Object Identifier’ = (Analog Input, Instance 2)
‘Time Stamp’ = 16
‘Notification Class’ = 4
‘Priority’ = 100
‘Event Type’ = OUT_OF_RANGE
‘Notify Type’ = ALARM
‘AckRequired’ = TRUE
‘From State’ = NORMAL
‘To State’ = HIGH_LIMIT
‘Event Values’ = ((Exceeding_Value, 80.1), (Status_Flags, (TRUE, FALSE, FALSE, FALSE)), (Deadband, 1.0), (Exceeded_Limit, 80.0))

这个 PDU 将被发送到每一个由与此**模拟输入**对象相关联的**通告类**对象指定的设备。假定 PDU 正确收到，作为报文已经收到的证实，将从每一个接收者返回一个‘Result(+)’证实原语。因为‘**要求确认**’参数值为 TRUE，这将被认为操作员已经注意和确认了警报。在这之后，将收到**确认报警**指示原语。原语参数如下：

Service = AcknowledgeAlarm
‘Acknowledging Process Identifier’ = 1
‘Event Object Identifier’ = (Analog Input, Instance 2)
‘Event State Acknowledged’ = HIGH_LIMIT
‘Time Stamp’ = 16
‘Acknowledgment Source’ = “MDL”
‘Time Of Acknowledment’ = (21-Jun-1992, 13:03:41.9)

本地 BACnet 设备将会查找相应的**模拟输入**对象实例，设置**确认转换**属性中的**进入异常**比特位，发出‘Result(+)’响应原语。注意在本例中‘**时间戳**’参数为一顺序号，其值同**有证实事件通告**一起发送。

E.1.4 获得报警摘要服务例程

Assumed objects: Object Identifier Object Name Object Type

(Analog Input, Instance 2) Zone 1_Temp ANALOG_INPUT

(Analog Input, Instance 3) Zone 2_Temp ANALOG_INPUT

一台 BACnet 设备在试图从另一台 BACnet 设备获取活动报警列表时，将发出如下服务请求：

Service = GetAlarmSummary

这个请求的典型响应将是不包含参数的‘**Result(+)**’，指示没有活动的报警；或者是传送如下活动报警列表的‘**Result(+)**’。

‘List of Alarm Summaries’ = (((Analog Input, Instance 2), HIGH_LIMIT,

B ‘011’), ((Analog Input, Instance 3), LOW_LIMIT, B ‘111’))

注意“Zone 2_Temp”中的**低阈值**已经确认，但“Zone 1_Temp”中的**高阈值**没有确认。

E.1.5 获得登记摘要服务例程

例 1：获取所有没有确认的报警的摘要。

Assumed objects: Object Identifier Object Name Object Type

(Analog Input, Instance 2) Zone 1_Temp ANALOG_INPUT

(Event Enrollment Instance 6) CW_Flow_Alar EVENT_ENROLLMENT

Service = GetEnrollmentSummary

‘Acknowledgment Filter’ = NOT-ACKED

这个请求的典型响应将是不包含参数的‘**Result(+)**’，指示没有未确认的报警；或者是传送如下活动报警列表的‘**Result(+)**’。

‘List of Enrollment Summaries’ = (((Analog Input, Instance 2),

OUT_OF_RANGE,

HIGH_LIMIT, 100, 4), ((Event Enrollment,

Instance 6),

CHANGE_OF_STATE, NORMAL, 50, 2))

例 2：获取关于某一具体设备预定的优先级在 6-10 之间的所有事件登记的摘要。

Assumed objects:	Object Identifier	Object Name	Object Type
	(Analog Input, Instance 2)	Zone 1_Temp	ANALOG_INPUT
	(Analog Input, Instance 3)	Zone 2_Temp	ANALOG_INPUT
	(Analog Input, Instance 4)	Zone 3_Temp	ANALOG_INPUT
	(Event Enrollment Instance 6)	CW_Temp_Alarm	EVENT_ENROLLMENT
	(Device, Instance 17)	Console 1	DEVICE

Service = GetEnrollmentSummary

'Acknowledgment Filter' = ALL

'Enrollment Filter' = ((Device, Instance 17), 9)

'Priority Filter' = (6, 10)

这个请求的典型响应将是不包含参数的'**Result(+)**'，指示没有符合特定要求的事件登记；或者是传送如下符合要求的事件登记对象列表的'**Result(+)**'。

'List of Enrollment Summaries' = (((Analog Input, Instance 2), OUT_OF_RANGE, NORMAL, 8, 4),

((Analog Input, Instance 3), OUT_OF_RANGE, NORMAL, 8, 4),

((Analog Input, Instance 4), OUT_OF_RANGE, NORMAL, 8, 4),

((Event Enrollment, Instance 7), FLOATING_LIMIT, NORMAL, 3, 8))

E.1.6 预订 COV 服务例程

本例说明了利用**预订 COV** 服务从一个**模拟输入**对象不确定地预订 COV 通告。

Service = SubscribeCOV

'Subscriber Process Identifier' = 18

‘Monitored Object Identifier’ = (Analog Input, Instance 10)
‘Issue Confirmed Notifications’ = TRUE
‘Lifetime’ = 0

E.1.7 无证实 COV 通告服务例程

下面的例程说明了被监视的**模拟输入**对象当前值改变的通告。

Service = UnconfirmedCOVNotification
‘Subscriber Process Identifier’ = 18
‘Initiating Device Identifier’ = (Device, Instance 4)
‘Monitored Object Identifier’ = (Analog Input, Instance 10)
‘Time Remaining’ = 0
‘List of Values’ = ((Present_Value, 65.0), (Status_Flags, (FALSE,
FALSE, FALSE, FALSE,)))

E.1.8 无证实事件通告服务例程

Assumed objects:

Object Identifier	Object Name	Object Type
(Analog Input, Instance 2)	Zone 1_Temp	ANALOG_INPUT

考虑一个名为“Zone 1_Temp”的支持内部事件报告的模拟输入对象。假定在区域一中温度刚升至超过高阈值，导致报警进入活动状态，在**确认转换**属性中的**进入异常**比特位被清除。进一步假定没有与之相关联的报文文本。设备将发出一个带有如下参数的**无证实事件通告**服务请求原语：

Service = UnconfirmedCOVNotification
‘Process Identifier’ = 1
‘Initiating Device Identifier’ = (Device, Instance 9)
‘Event Object Identifier’ = (Analog Input, Instance 2)
‘Time Stamp’ = 16
‘Notification Class’ = 4
‘Priority’ = 100
‘Event Type’ = OUT_OF_RANGE
‘Notify Type’ = ALARM
‘AckRequired’ = TRUE
‘From State’ = NORMAL

‘To State’ = HIGH_LIMIT

‘Event Values’ = ((Present_Value, 80.1), (Status_Flags, (TRUE, FALSE, FALSE, FALSE)), (Deadband, 1.0), (High_Limit, 80.0))

这个 PDU 将根据与这个**模拟输入**对象相关联的**通告类**对象的**接收者列表**属性值被发送到一个特定接收者、本地广播、远程广播、或者全局广播。因为‘**要求确认**’参数值为 TRUE，这将被认为操作员已经注意和确认了警报。在这之后，将收到**确认报警**指示原语。原语参数如下：

Service = AcknowledgeAlarm

‘Acknowledging Process Identifier’ = 1

‘Event Object Identifier’ = (Analog Input, Instance 2)

‘Event State Acknowledged’ = HIGH_LIMIT

‘Time Stamp’ = 16

‘Acknowledgment Source’ = “MDL”

‘Time Of Acknowledment’ = (21-Jun-1992, 13:03:41.9)

本地 BACnet 设备将会查找相应的**模拟输入**对象实例，设置**确认转换**属性中的**进入异常**比特位，发出‘**Result(+)**’响应原语。注意在本例中‘**时间戳**’参数为一顺序号，其值同**无证实事件通告**一起发送。

E.2 文件访问服务

下面的例程说明了文件的典型使用。文件的实际格式和内容由生产商根据应用自行定义。

E.2.1 基本读文件服务例程

例 1：从文件中读数据。

Assumed objects:	Object Identifier	Object Name	Object Type
	(File, Instance 1)	ChillerData	FILE

Service = AtomicReadFile

‘File Identifier’ = (File, Instance 1)
‘Stream Access’ :
‘File Start Position’ = 0
‘Requested Octet Count’ = 27

‘Result(+)’参数为：

‘End Of File’ = FALSE
‘Stream Access’ :
‘File Start Position’ = 0
‘File Data’ = “Chiller01 On-Time = 4.3 Hours”

例 2：从文件中读记录。

Assumed objects: Object Identifier Object Name Object Type
(File, Instance 2) ChillerTrendLog FILE
Service = AtomicReadFile
‘File Identifier’ = (File, Instance 2)
‘Stream Access’ :
‘File Start Position’ = 14
‘Requested Record Count’ = 3

‘Result(+)’参数为：

‘End Of File’ = TRUE
‘Stream Access’ :
‘File Start Record’ = 14
‘Requested Record Count’ = 2
‘File Record Data’ = ((12:00, 45.6), (12:15, 44.8))

注意在本例中并不是所有的请求的数据都返回。

E.2.2 基本写文件服务例程

例 1：向文件中写数据。

```
Assumed objects:    Object Identifier          Object Name      Object Type
                    (File, Instance 1)         ChillerData      FILE
    Service                     = AtomicWriteFile
     'File Identifier'              = (File, Instance 1)
     'Stream Access' :
            'File Start Position'       = 30
            'File Data'                 = "Chiller01 On-Time = 4.3 Hours"
```

'Result(+)' 参数为：

```
     'Stream Access' :
            'File Start Position'       = 30
```

例 2：向文件中追加两个记录。

```
Assumed objects:    Object Identifier          Object Name      Object Type
                    (File, Instance 2)         ChillerTrendLog  FILE
    Service                     = AtomicWriteFile
     'File Identifier'              = (File, Instance 2)
     'Record Access' :
            'File Start Record'         = -1
            'Record Count'          = 2
            'File Record Data'      = ((12:00, 45.6),
                                      (12:15, 44.8))
```

'Result(+)' 参数为：

```
     'Record Access' :
            'File Start Record'         = 14
```

E.3 对象访问服务

E.3.1 添加列表元素服务例程

例 1：向组对象增加成员。

Assumed objects:	Object Identifier	Object Name	Object Type
	(Group, Instance 3)	AHU1_GRAPH	GROUP
	(Analog Input, Instance 9)	AHU1_SA_TEMP	ANALOG_INPUT
	(Analog Input, Instance 10)	AHU1_RA_TEMP	ANALOG_INPUT
	(Analog Input, Instance 11)	AHU1_OAD_POS	ANALOG_INPUT
	(Analog Input, Instance 12)	AHU1_SA_PRESS	ANALOG_INPUT
	(Analog Input, Instance 13)	AHU1_CW_VALVE	ANALOG_INPUT
	(Analog Input, Instance 14)	OA_TEMP	ANALOG_INPUT
	(Analog Input, Instance 15)	OA_HUMID	ANALOG_INPUT

考虑包含用于图形显示的如下组对象的 BACnet 设备。

Property: Object_Identifier = (Group, Instance 3)
Property: Object_Name = “AHU1_GRAPH”
Property: Object_Type = GROUP
Property: Description = “Points for AHU1 graphic”
Property: List_Of_Group_Members = (((Analog Input Instance 9), (Present_Value, Reliability)),
((Analog Input Instance 10), (Present_Value, Reliability)),
((Analog Input Instance 11), (Present_Value, Reliability)),
((Analog Input Instance 12), (Present_Value, Reliability,
Description)),
((Analog Input Instance 13), (Present_Value, Reliability,
Description)),
((Analog Input Instance 14), (Present_Value)))
Property: Present_Value = (65.2, NO_FAULT_DETECTED, 72.4,
NO_FAULT_DETECTED, 99, NO_
FAULT_DETECTED, “Inches of water”, 32,

NO_FAULT_DETECTED, “% open”, 68.3)

系统操作员决定升级 AHU1 中的控制软件来使用热函节约装置回路。于是，操作员想向“AHU1_GRAPH”增加一个湿度读数。**添加列表元素**服务原语带有如下参数：

Service = AddListElement
‘Object Identifier’ = (Group, Instance 3)
‘Property Identifier’ = List_Of_Group_Members
‘List of Elements’ = ((Analog Input, Instance 15), (Present_Value,
Reliability))

假定服务请求成功，发出‘**Result(+)**’服务原语，对象“AHU1_GRAPH”现在具有如下属性：

Property: Object_Identifier = (Group, Instance 3)
Property: Object_Name = “AHU1_GRAPH”
Property: Object_Type = GROUP
Property: Description = “Points for AHU1 graphic”
Property: List_Of_Group_Members = (((Analog Input Instance 9), (Present_Value, Reliability)),
((Analog Input Instance 10), (Present_Value, Reliability)),
((Analog Input Instance 11), (Present_Value, Reliability)),
((Analog Input Instance 12), (Present_Value, Reliability,
Description)),
((Analog Input Instance 13), (Present_Value, Reliability,
Description)),
((Analog Input Instance 14), (Present_Value)),

((Analog Input Instance 15), (Present_Value, Reliability)))

Property: Present_Value = (65.2, NO_FAULT_DETECTED, 72.4, NO_FAULT_DETECTED, 99, NO_FAULT_DETECTED, “Inches of water”, 32, NO_FAULT_DETECTED, “% open”, 68.3,42.1, NO_FAULT_DETECTED)

注：在本例中，新增加的元素在列表的末端。这是符合逻辑的放法，因为列表必须被遍历以决定“新”的元素是否已经存在。本标准不要求一定要在末端增加新的列表元素。

E.3.2 删除列表元素服务例程

例 1：从组中删除成员。

Assumed objects:

Object Identifier	Object Name	Object Type
(Group, Instance 3)	AHU1_GRAPH	GROUP
(Analog Input, Instance 9)	AHU1_SA_TEMP	ANALOG_INPUT
(Analog Input, Instance 10)	AHU1_RA_TEMP	ANALOG_INPUT
(Analog Input, Instance 11)	AHU1_OAD_POS	ANALOG_INPUT
(Analog Input, Instance 12)	AHU1_SA_PRESS	ANALOG_INPUT
(Analog Input, Instance 13)	AHU1_CW_VALVE	ANALOG_INPUT
(Analog Input, Instance 14)	OA_TEMP	ANALOG_INPUT

这是使用删除列表元素服务来改变已存在的组对象的例程。假定组对象“AHU1_GRAPH”如下定义：

Property: Object_Identifier = (Group, Instance 3)

Property: Object_Name = “AHU1_GRAPH”

Property: Object_Type = GROUP

Property: List_Of_Group_Members = (((Analog Input Instance 9), (Present_Value, Reliability)),

((Analog Input Instance 10), (Present_Value, Reliability)),

((Analog Input Instance 11), (Present_Value, Reliability)),

((Analog Input Instance 12), (Present_Value, Reliability,
Description)),
((Analog Input Instance 13), (Present_Value, Reliability,
Description)),
((Analog Input Instance 14), (Present_Value)))

Property: Present_Value = (65.2, NO_FAULT_DETECTED, 72.4,
NO_FAULT_DETECTED, 99.0,
NO_FAULT_DETECTED, 0.67,
NO_FAULT_DETECTED, “Inches of water”, 32.0,
NO_FAULT_DETECTED, “% open”, 68.3)

系统操作员正在升级图形显示，决定组中**描述**属性实际上不使用，希望将它们删除。虽然**描述**是属性列表元素，但由于它嵌套在**组成员列表**中，所以它不能被本服务删除。如下所示需要两步过程。

下面的服务请求被发出：

Service = Remove List Element
‘Object Identifier’ = (Group, Instance 3)
‘Property Identifier’ = List_Of_Elements
‘List of Elements’ = (((Analog Input, Instance 12), (Present_Value,
Reliability, Description)),
((Analog Input, Instance 13), (Present_Value,
Reliability, Description)))

服务请求成功，在这点对象“AHU1_GRAPH”状态为：

Property: Object_Identifier = (Group, Instance 3)
Property: Object_Name = “AHU1_GRAPH”
Property: Object_Type = GROUP
Property: List_Of_Group_Members = (((Analog Input Instance 9), (Present_Value, Reliability)),
((Analog Input Instance 10), (Present_Value,

Reliability)),

((Analog Input Instance 11), (Present_Value, Reliability)),

((Analog Input Instance 14), (Present_Value)))

Property: Present_Value = (65.2, NO_FAULT_DETECTED, 72.4, NO_FAULT_DETECTED, 99.0, NO_FAULT_DETECTED, 68.3)

添加列表元素服务现在用来替代已经删除但仍需要用来作图形显示的组成员。

下面的服务请求被发出：

Service = AddListElement

'Object Identifier' = (Group, Instance 3)

'Property Identifier' = "List_Of_Group_Members"

'List of Elements' = (((Analog Input, Instance 12), (Present_Value, Reliability)),

((Analog Input, Instance 13), (Present_Value, Reliability)))

服务请求成功，现在"AHU1_GRAPH"为预想的格式：

Property: Object_Identifier = (Group, Instance 3)

Property: Object_Name = "AHU1_GRAPH"

Property: Object_Type = GROUP

Property: List_Of_Group_Members = (((Analog Input Instance 9), (Present_Value, Reliability)),

((Analog Input Instance 10), (Present_Value, Reliability)),

((Analog Input Instance 11), (Present_Value, Reliability)),

((Analog Input Instance 14), (Present_Value)))

((Analog Input Instance 12), (Present_Value, Reliability)),

((Analog Input Instance 13), (Present_Value,

Reliability)))

Property: Present_Value = (65.2, NO_FAULT_DETECTED, 72.4,
NO_FAULT_DETECTED, 99.0,
NO_FAULT_DETECTED, 68.3, 0.67,
NO_FAULT_DETECTED, 32.0,
NO_FAULT_DETECTED)

E.3.3 创建对象服务例程

这是一个使用**创建对象**服务来创建一个用于趋向记录的文件对象。

Service = CreateObject
'Object Identifier' = FILE
'List of Initial Values' = ((Object_Name, "Trend 1"),
(File_Access_Method,
RECORD_ACCESS))

假定**创建对象**服务请求正确收到，而且设备有创建文件对象的能力，则发出传送新创建的对象的对象标识符的'**Result(+)**'服务原语。

'Object_Identifier' = (File, Instance 13)

注意在本例中，只有新建对象的**文件类型**属性和**文件访问方法**属性作为对象创建时的参数进行初始化。新建对象的其它属性将包含生产商提供的缺省值。如果这些缺省值不可接受，那么属性的初始值可以是包含在**创建对象**服务的'**初始值列表**'参数中的值。或者，作为另一种选择，对象的任意可写属性可以在对象创建以后利用**写属性**服务或者**写多个属性**服务请求进行初始化和写操作。

E.3.4 删除对象服务例程

Assumed objects:	Object Identifier	Object Name	Object Type
	(Group, Instance 6)	ZONE1_TEMPS	GROUP
	(Group, Instance 7)	NotDeletable	GROUP

考虑一包含两个组对象"ZONE1_TEMPS"和"NotDeletable"的 BACnet 设备。对象"NotDeletable"在配置时被创建和保护，不能被此协议服务删除。

例 1：对象的成功删除。

下面的服务请求被发出：

Service = DeleteObject

‘Object Identifier’ = (Group, Instance 6)

对象被删除，且服务方发送“**Result(+)**”响应原语。

例 2：删除组对象的不成功尝试。

下面的服务请求被发出：

Service = DeleteObject

‘Object Identifier’ = (Group, Instance 7)

对象被保护，不能被此协议服务删除。服务方发出如下响应原语。

‘Result (-)’ :

‘Error Type’ = (SERVICES, OBJECT_NOT_DELETABLE)

E.3.5 读属性服务例程

Assumed objects:	Object Identifier	Object Name	Object Type
	(Analog Input, Instance 5)	SPACE_TEMP	ANALOG_INPUT

我们希望读一名为“SPACE_TEMP”的模拟输入的当前值。

Service = ReadProperty

‘Object Identifier’ = (Analog Input, Instance 5)

‘PropertyIdentifier’ = Present_Value

假定目标机器能够根据此 ID 和请求的属性寻找对象，则结果为：

‘Object Identifier’ = (Analog Input, Instance 5)

‘PropertyIdentifier’ = Present_Value

‘Value’ = 72.3

结果表明“SPACE_TEMP”的当前值为 72.3。

E.3.6 条件读属性服务例程

例 1：用于获取所有二进制输入对象的对象标识符的参数。

Assumed objects:	Object Identifier	Object Name	Object Type
	(Binary Input, Instance 1)	AC1 Status	BINARY_INPUT
	(Binary Input, Instance 2)	AC2 Status	BINARY_INPUT
	(Binary Input, Instance 3)	AC1 Freezestat	BINARY_INPUT
	(Binary Input, Instance 4)	AC2 Freezestat	BINARY_INPUT

假设一位用户希望找出一台 BACnet 设备中所有二进制输入对象的对象标识符。

Service = ReadPropertyConditional

‘Object Selection Criteria’:

‘Selection Logic’ = AND (note that either AND or OR may be used here because there is only one selection criterion)

‘List of Selection Criteria’ = (Object_Type, =, BINARY_INPUT)

这将产生在‘**Result(+)**’原语中传送的该 BACnet 设备中所有二进制输入对象的一个列表。

‘List of Read Access Results’ = ((Binary Input, Instance 1), (Binary Input, Instance2), (Binary Input, Instance 3), (Binary Input, Instance 4))

在恰当的选择标准下，**条件读属性**服务可用于返回与公共属性值相联系的任意对象组的对象标识符，而不管属性是否为对象类型，**当前值**，**可靠性**，**状态**或者一些其它属性。

例 2：用于**模拟输入**对象的条件读的参数。

Assumed objects:	Object Identifier	Object Name	Object Type
	(Analog Input, Instance 1)	Room 101 Space Temp	ANALOG_INPUT
	(Analog Input, Instance 2)	AC1 HWS Temp	ANALOG_INPUT
	(Analog Input, Instance 3)	AC1 HWR Temp	ANALOG_INPUT

假设我们希望读出**当前值**大于 100.0 的所有**模拟输入**对象的**当前值**属性。

Service = ReadPropertyConditional Service
'Object Selection Criteria':
'Selection Logic' = AND
'List of Selection Criteria' = ((Object_Type, =, ANALOG_INPUT),
(Present_Value, >, 100.0))

这可导致如下列表：

'List of Read Access Results' = (((Analog Input, Instance 1), Present_Value, 102.3),
((Analog Input, Instance 2), Present_Value, 180.7),
((Analog Input, Instance 3), Present_Value, 142.1))

例 3：用于查找不可靠的或脱离服务的对象的参数。

Assumed objects:	Object Identifier	Object Name	Object Type
	(Binary Input, Instance 1)	AC1 Status	BINARY_INPUT
	(Analog Input, Instance 2)	AC1 Discharge Temp	ANALOG_INPUT

下面的参数可用来查找所有**可靠性**属性不为 NO_FAULT_DETECTED 或者**脱离服务**属性为 TRUE 的对象。

Service = ReadPropertyConditional Service
'Object Selection Criteria':
'Selection Logic' = OR
'List of Selection Criteria' = ((Reliability, ≠ ,

NO_FAULT_DETECTED),

(Out_Of_Service, =, TRUE))

'List of Property Identifiers' = (Present_Value, Reliability, Out_Of_Service)

典型结果可能如下：

'List of Read Access Results' = (((Binary Input, Instance 1), Present_Value, ACTIVE,

Reliability, NO_FAULT_DETECTED, Out_Of_ Service, TRUE)),

((Analog Input, Instance 2), Present_Value, 250.0,

Reliability, UNRELIABLE_OTHER, Out_Of_ Service, FALSE)))

例 4：

Assumed objects:	Object Identifier	Object Name	Object Type
	(Analog Input, Instance 4)	AC1 Supply Temp	ANALOG_INPUT
	(Analog Input, Instance 7)	CWP1 Pressure	ANALOG_INPUT
	(Analog Input, Instance 8)	Chiller 1 Freon Pressure	ANALOG_INPUT
	(Analog Input, Instance 10)	AC2 Supply Temp	ANALOG_INPUT
	(Analog Input, Instance 12)	AC3 Supply Temp	ANALOG_INPUT

下面的参数可用来查找所有**对象名称**与包含通配符“*”或“？”的比较相符的对象的标识符。

Service = ReadPropertyConditional Service

'Object Selection Criteria':

'Selection Logic' = OR

'List of Selection Criteria' = ((Object_Name, =, “C* Pressure”),

(Object_Name, =, “AC? Supply Temp”))

'List of Property Identifiers' = (Object_Name)

典型结果可能如下：

‘List of Read Access Results’ = (((Analog Input, Instance 4), “AC1 Supply Temp ”),
((Analog Input, Instance 7), “CWP1 Pressure”),
((Analog Input, Instance 8), “Chiller 1 Freon Pressure”),
((Analog Input, Instance 10), “AC2 Supply Temp”),
((Analog Input, Instance 12), “AC3 Supply Temp”),

E.3.7 读多个属性服务例程

例 1：用于读单一对象中多个属性的参数。

Assumed objects:	Object Identifier	Object Name	Object Type
	(Analog Input, Instance 16)	SPACE_TEMP	ANALOG_INPUT

我们希望读一名为“SPACE_TEMP”的模拟点的当前值和可靠性。

Service = ReadPropertyMultiple
‘List of Read Access Specifications’ = ((Analog Input, Instance 16), ((Present_Value, Reliability))

假定目标机器能够利用此标识符和请求的属性寻找对象，结果将为：

‘List of Read Access Specifications’ = ((Analog Input, Instance 16), ((Present_Value, 72.3),
(Reliability, NO_FAULT_DETECTED)))

结果表明“SPACE_TEMP”的当前值为 72.3，而且只要是被能确定的，即为可靠的。

例 2：用于读若干个对象的属性的参数。

Assumed objects:	Object Identifier	Object Name	Object Type
	(Analog Input, Instance 33)	CW_STEMP	ANALOG_INPUT
	(Analog Input, Instance 34)	CW_RTEMP	ANALOG_INPUT
	(Analog Input, Instance 35)	CW_FLOW	ANALOG_INPUT

假设我们希望访问在同一冷却水系列中属性组合在名为“CW_STEMP”，“CW_RTEMP”，和“CW_FLOW”的对象中的三个模拟输入的当前值。

Service = ReadPropertyMultiple

‘List of Read Access Specifications’ = (((Analog Input, Instance 33), ((Present_Value)),

((Analog Input, Instance 50), ((Present_Value)),

((Analog Input, Instance 35), ((Present_Value)))

假定三个请求的值中成功访问两个：

‘List of Read Access Results’ = (((Analog Input, Instance 33), (Present_Value, 42.3)),

((Analog Input, Instance 33), (Present_Value,

‘Property Access Error’ = (OBJECT, UNKNOWN_OBJECT))),

((Analog Input, Instance 35), (Present_Value, 435.7)))

结果表明供应温度访问成功，值为 42.3，流量为 435.7。但是返回温度当前值访问失败。请求中与此相关的错误类型为**错误类型**对象，**错误代码**为**未知的对象**。这是因为对象标识符错误，可能是由于操作员错误的操作。

E.3.8 写属性服务例程

Assumed objects:	Object Identifier	Object Name	Object Type
	(Analog Input, Instance 1)	HW_Setpoint	ANALOG_INPUT

我们希望修改一个对象名为“HW_Setpoint”的模拟值对象的当前值。

Service = WriteProperty
‘Object Identifier’ = (Analog Value, Instance 1)
‘PropertyIdentifier’ = Present_Value
‘Property Value’ = 180.0

假定目标机器能够利用特定的**对象标识符**寻找对象，并修改**当前值**属性，那么结果将是发出一不带参数的‘**Result(+)**’原语。此确认将在一不带参数的**简单确认** PDU 中传送给服务请求方。

E.3.9 写多个属性服务例程

Assumed objects:	Object Identifier	Object Name	Object Type
	(Analog Input, Instance 5)	Room 1 Temp Setpoint	ANALOG_INPUT
	(Analog Input, Instance 6)	Room 2 Temp Setpoint	ANALOG_INPUT
	(Analog Input, Instance 7)	Room 3 Temp Setpoint	ANALOG_INPUT

我们希望修改用于三个空间温度环控制的三个不同的模拟变量对象的**当前值**。

Service = WritePropertyMultiple
‘List of Read Access Specifications’ = (((Analog Input, Instance 5), ((Present_Value, 67.0)),
((Analog Input, Instance 6), ((Present_Value, 67.0)),
((Analog Input, Instance 7), ((Present_Value, 72.0)))

假定响应方 BACnet 用户能够寻找三个对象中的每一个，那么结果将是这些属性中先前的值分别被提供的值替换，并且将发出一不带附加参数的‘**Result(+)**’原语。这个确认在一个不带参数的**简单确认** PDU 中传送给服务请求方。

E.4 远程设备管理服务

E.4.1 设备通信控制服务例程

当在一个 BACnet 网络中处理问题时，有必要对来自某一特殊设备的通信交换停止几分钟。通过如下的**设备通信控制**服务完成这个任务。

```
Service                    = DeviceCommunicationControl
  'Time Duration'              = 5
  'Enable/Disable'             = DISABLE
  'Password'                   = "#egbdf!"
```

E.4.2 有证实专有传输服务例程

这是一个利用 BACnet 有证实服务的典型专有服务请求。

```
Service                    = ComfirmedPrivateTransfer
  'VendorID'                   = 25
  'Service Number '            = 8
  'Service Parameters'         = (72.4, X '1649')
```

假定服务成功执行，但不需向请求方 BACnet 用户返回结果，那么将返回一个传送如下信息的 **'Result(+)'** 原语。

```
  'VendorID'                   = 25
  'Service Number '            = 8
  'Result Block'               = NIL
```

E.4.3 无证实专有传输服务例程

这是一个利用 BACnet 无证实服务的典型专有服务请求。

```
Service                    = UncomfirmedPrivateTransfer
  'VendorID'                   = 18
  'Service Number '            = 12
  'Service Parameters'         = (72.4, X '1649')
```

因为这是无证实服务，所以无响应。

E.4.4 重新初始化设备服务例程

本例说明了**重新初始化设备**服务的使用，用于为在服务中需要密码的设备请求热启动。

```
Service                        = ReinitializeDevice
  'Reinitialize State of Device'      = WARMSTART
  'Password'                           = "AbCdEfGh"
```

如果密码验证成功，那么将返回一个'**Result(+)**'原语，然后设备将重新初始化。否则，将返回一个'**Result(-)**'原语。

E.4.5 有证实文本报文服务例程

本例说明了**有证实文本**报文服务的使用。

```
Service                        = ComfirmedTextMessage
  'Text Message Source Device'    = (Device, Instance 5)
  'Message Priority'                   = NORMAL
  'Message'                            = "P.M. required for PUMP347"
```

如果**有证实文本**报文请求成功收到，那么将返回一个'**Result(+)**'原语，否则，将返回一个'**Result(-)**'原语。

E.4.6 无证实文本报文服务例程

本例说明了**无证实文本报文**服务的使用。

```
Service                        = UncomfirmedTextMessage
  'Text Message Source Device'    = (Device, Instance 5)
  'Message Priority'                   = NORMAL
  'Message'                            = "P.M. required for PUMP347"
```

E.4.7 时间同步服务例程

下面是一个**时间同步**服务的参数使用的例程。

```
Service                        = TimeSynchronization
```

‘Time’:

‘Date’ = 17-Nov-92

‘Time’ = 22:45:30.7

此请求将通告所有的远程设备当前时间是 1992 年 11 月 17 日晚上 10:45:30.7。远程设别将使用此信息更新它们的内部时钟以使所有的设备同步。

E.4.8 Who-Has 和 I-Have 服务例程

下面是用于 I-Have 与 I-Have 服务的参数使用的例程。

例 1：寻找包含已知**对象名称**的对象的设备。

考虑试图寻找对象名为“OA Temp”的对象。

Service = Who-Has

‘Object Name’ = “OA Temp”

假定这里只有一台有上诉对象的设备，那么将收到下面的 I-Have 服务指示。

Service = I-Have

‘Device Identifier’ = (Device, Instance 8)

‘Object Identifier’ = (Analog Input, Instance 3)

‘Object Name’ = “OA Temp”

例 2：寻找包含已知**对象标识符**的对象的设备。

考虑试图寻找**对象标识符**为(Analog Input, Instance 3)的对象。

Service = I-Have

‘Object Identifier’ = (Analog Input, Instance 3)

假定这里只有一台有上诉对象的设备，那么将收到下面的“我有”服务指示。

Service = I-Have

‘Device Identifier’ = (Device, Instance 8)

‘Object Identifier’ = (Analog Input, Instance 3)

‘Object Name’ = “OA Temp”

E.4.9 Who-Is 和 I-Am 服务例程

下面是用于 **Who-Is** 与 **I-Am** 服务的参数使用的例程。

例 1：建立一台设备的网络地址。

我们希望确定另一台 BACnet 设备的网络地址，但只知道该设备的**设备标识符**。

```
Service                    = Who-Is
'Who-Is Device Identifier'         = (Device, Instance 3)
```

假定在网络上有这样一台设备，那么一会儿后它将使用 **I-Am** 服务进行响应：

```
Service                    = I-Am
'I-Am Device Identifier'           = (Device, Instance 3)
'Max APDU Length Accepted'     = 1024
'Segmentation Supported'       = NO_SEGMENTATION
'Vendor Identifier'                = 99
```

MAC 层同网络层源网络 SNET 与源地址 SADR 结合的源地址提供所有的同设备(Device, Instance 3)通信所需要的地址信息。

例 2：找出几乎所有的网络设备。

假设我们希望确定当前在本地网络上有哪些设备。

```
Service                    = Who-Is Service
```

网络上的每一台设备均作回答，导致若干个 **I-Am** 服务请求。

```
Service                    = I-Am
'I-Am Device Identifier'           = (Device, Instance 1)
'Max APDU Length Accepted'     = 480
'Segmentation Supported'       = SEGMENTED_TRANSMIT
'Vendor Identifier'                = 99
```

```
Service                      = I-Am
 ‘I-Am Device Identifier’          = (Device, Instance 2)
 ‘Max APDU Length Accepted’     = 206
 ‘Segmentation Supported’       = SEGMENTED_RECEIVE
 ‘Vendor Identifier’               = 33

Service                      = I-Am
 ‘I-Am Device Identifier’          = (Device, Instance 3)
 ‘Max APDU Length Accepted’     = 1024
 ‘Segmentation Supported’       = NO_SEGMENTATION
 ‘Vendor Identifier’               = 99

Service                      = I-Am
 ‘I-Am Device Identifier’          = (Device, Instance 4)
 ‘Max APDU Length Accepted’     = 128
 ‘Segmentation Supported’       = SEGMENTED_BOTH
 ‘Vendor Identifier’               = 66
```

通过在网络层头部中包含全局网络地址(X‘FFFF’),这个过程可以扩展用来找出BACnet 互联网上几乎所有的设备。

E.5 虚拟终端服务

下面是用于 **VT 开启**，**VT-数据**与 **VT 关闭**服务的参数使用的例程。为了易读，有些例程中包含嵌套的控制字符，如回车。这些字符将在大括号内显示，如“{cr}{lf} text”。

同缺省终端建立通话：

我们希望创建一个同另一台 BACnet 设备中的缺省终端类型的操作员接口的 **VT 会话**。

```
Service                      = VT-Open
 ‘VT-class’                       = ANSI_X3.64
 ‘Local VT Session Identifier’      = 5
```

假定目标机器能够创建新的 **VT 会话**，那么‘**Result(+)**’响应为：

```
 ‘Remote VT Session Identifier’     = 29
```

结果显示目标设备已经接受了请求并创建了一个新的会话。

现在会话建立了，远程设备上的操作员接口程序通过发送 **VT 数据**请求提示我们开始。

```
Service                    = VT-Data
 'VT-session Identifier'          = 5
 'VT-new Data'                = "{cr}{lf} Enter User Name:"
 'VT-data Flag'                   = 0
```

我们的操作员接口显示队列是空的，所以能够接收所有进来的字符。因此我们的 **VT 用户**发出 **'Result(+)'**：

```
 'All New Data Accepted'      = TRUE
```

最后，我们的操作员输入他（她）的名字响应提示，这将发出 **VT 数据**请求：

```
Service                    = VT-Data
 'VT-session Identifier'          = 29
 'VT-new Data'                = "FRED {cr}"
 'VT-data Flag'                   = 0
```

同样地，远程操作员接口接收字符，响应 **'Result(+)'**：

```
 'All New Data Accepted'      = TRUE
```

然后远程操作员接口回应输入的字符，发出另一指示：

```
Service                    = VT-Data
 'VT-session Identifier'          = 5
 'VT-new Data'                = "FRED {cr}{lf} Enter Password"
 'VT-data Flag'                   = 1
```

我们的操作员显示队列是空的，所有能够接收所有进来的字符。因此我们的虚拟终端用户发出 **'Result(+)'**：

‘All New Data Accepted’ = TRUE

出于某种原因，FRED 决定取消这些虚拟终端对话，并通告操作员接口程序。接口程序发出虚拟终端关闭请求：

Service = VT-Close
‘List of Remote VT Session Identifiers’ = (29)

E.6 安全服务

本例显示了通过密钥服务器发送会话密钥给两台设备A和B的请求**密钥**服务和**认证**服务的使用。此过程在 24.2.1 节中详细述论。加密部分用斜体字表示。

设备 A 直接向密钥服务器发出使用其**私有密钥 PK_A**加密的请求密钥请求，开始此过程。假定 **BACnet 有证实请求 PDU** 的‘Invoke ID’为 5：

Service = RequestKey
‘Requesting Device Identifier’ = (Device, Instance 1)
‘Requesting Device Address’ = (2, X‘11’)
‘Remote Device Identifier’ = (Device, Instance 2)
‘Remote Device Address’ = (2, X‘22’)

为了验证请求是否是由设备 A 发出的，密钥服务器发出包括‘**期待的调用 ID**’参数值为 5 的**认证**服务请求。

Service = Authenticate
‘Pseudo Random Number’ = X‘12345678’
‘Expected Invoke ID’ = 15

注意**认证**服务请求同样也是在传输前使用设备 A 的**私有密钥**加密了的，该密钥为存储于密钥服务器中的一个拷贝。

因为事实上是设备 A 发出的请求密钥服务请求，所有**认证**服务请求能够被设备 A 利用某**私有密钥 PK_A**解密。而且‘**期待的调用 ID**’参数值与先前请求密钥请求中的‘Invoke ID’相符。因此正如 24.5.2 节中所描述的，设备 A 使用‘**伪随机数**’产生‘修改的随机数’参数组成的**认证确认**的**复杂确认**。

‘Authenticate-ACK’ = X‘93B5D7F1’

然后密钥服务器产生一个**会话密钥**，使用各设备各自的 PK 加密，将其写入所有使用**添加列表元素**服务的设备 A 和设备 B 中**设备**对象的**会话密钥列表**属性中。每个设备在执行**添加列表元素**服务请求与发出确认之前都必须执行上述的认证过程。最后，如果所有的认证均完成，那么密钥服务器向设备 A 发出一个**简单确认** PDU，表示最初的**请求密钥**服务已经成功完成。

附件 F — APDU 编码例程（资料）

（本附件不是标准的一部分，仅仅作为资料使用）

本附件说明 BACnet 编码准则的应用，给出附件 E 中例程的编码的 APDU。

F.1 报警和事件服务编码例程

F.1.1 例 E.1.1 编码——确认报警服务

例 E.1.1 编码包含于 F.1.3 和 F.1.8。

F.1.2 例 E.1.2 编码——有证实 COV 通告服务

```
X ‘00’          PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0,
                SA=0)
X ‘02’          Maximum APDU Size Accessed = 206 octets
X ‘0F’          Invoke ID = 15
X ‘01’          Service Choice = 1 (ConfirmedCOVNotification-Request)
X ‘09’          SD Context Tag 0 (Subscriber Process Identifier, L=1)
X ‘12’          Subscriber Process Identifier =18
X ‘1C’          SD Context Tag 1 (Initiating Device Identifier, L=4)
X ‘02000004’        Device, Instance 4
X ‘2C’          SD Context Tag 2 (Monitored Object Identifier, L=4)
X ‘0000000A’        Analog Input, Instance Number = 10
X ‘’ 39’            SD Context Tag 3 (Time Remaining, L=1)
X ‘00’          Time Remaining = 0
X ‘4E’          PD Opening Tag 4 (List Of Values)
        X ‘09’          SD Context Tag 0 (Property Identifier, L=1)
        X ‘55’          85 (PRESENT_VALUE)
        X ‘2E’          PD Opening Tag 2 (Value)
            X ‘44’          Application Tag 4 (Real, L=4)
            X ‘42820000’        65.0
        X ‘2F’          PD Closing Tag 2 (Value)
        X ‘09’          SD Context Tag 0 (Property Identifier, L=1)
        X ‘6F’          111 (STATUS_FLAGS)
        X ‘2E’          PD Opening Tag 2 (Value)
```

```
            X ‘82’          Application Tag 8 (Bit String, L=2)
            X ‘0400’            0, 0, 0, 0 (FALSE, FALSE, FALSE, FALSE)
        X ‘2F’          PD Closing Tag 2 (Value)
X ‘4F’          PD Closing Tag 4 (List Of Values)
```

假定服务进程执行正确，那么将返回一个简单确认：

```
X ‘20’          PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘0F’          Invoke ID = 15
X ‘01’          Service ACK Choise = 1 (ConfirmedCOVNotification)
```

F.1.3 例 E.1.3 编码——有证实事件通告服务

```
X ‘00’          PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0,
                SA=0)
X ‘02’          Maximum APDU Size Accessed = 206 octets
X ‘10’          Invoke ID = 16
X ‘02’          Service Choice = 2 (ConfirmedEventNotification-Request)
X ‘09’          SD Context Tag 0 (Process Identifier, L=1)
X ‘01’          Process Identifier =1
X ‘1C’          SD Context Tag 1 (Initiating Device Identifier, L=4)
X ‘02000004’        Device, Instance 4
X ‘2C’          SD Context Tag 2 (Event Object Identifier, L=4)
X ‘00000002’        Analog Input, Instance Number = 2
X ‘3E’          PD Opening Tag 3 (Time Stamp)
        X ‘19’          SD Context Tag 1 (SequenceNumber, L=1)
        X ‘10’          16
X ‘3F’          PD Closing Tag 3 (Time Stamp)
X ‘49’          SD Context Tag 4 (Notification Class, L=1)
X ‘04’          4
X ‘59’          SD Context Tag 5 (Priority, L=1)
X ‘64’          100
X ‘69’          SD Context Tag 6 (Event Type, L=1)
X ‘05’          5 (OUT_OF_RANGE)
X ‘89’          SD Context Tag 8 (Notify Type, L=1)
```

```
X ‘00’          0 (ALARM)
X ‘99’          SD Context Tag 9 (AckedRequired, L=1)
X ‘01’          TRUE
X ‘A9’          SD Context Tag 10 (From State, L=1)
X ‘00’          0 (NORMAL)
X ‘B9’          SD Context Tag 11 (To State, L=1)
X ‘03’          3 (HIGH_LIMIT)
X ‘CE’          PD Opening Tag 12 (Event Values)
        X ‘5E’          PD Opening Tag 5 (BACnetNotificationParameters,
                                          Choice 5=OUT_OF_RANGE)
            X ‘0C’          SD Context Tag 0 (Exceeding Value, L=4)
            X ‘42A03333’        80.1
            X ‘1A’          SD Context Tag 1 (Status Flags, L=2)
            X ‘0480’            1, 0, 0, 0 (TRUE, FALSE, FALSE, FALSE)
            X ‘2C’          SD Context Tag 2 (Deadband, L=4)
            X ‘3F800000’        1.0
            X ‘3C’          SD Context Tag 3 (Exceed Limit, L=4)
            X ‘42A00000’        80.0
        X ‘5F’          PD Closing Tag 5 (BACnetNotificationParameters)
X ‘CF’          PD Closing Tag 12 (Event Values)
```

假定服务进程执行正确，那么将返回一个简单确认：

```
X ‘20’          PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘10’          Invoke ID = 16
X ‘02’          Service ACK Choise = 2 (ConfirmedEventNotification)
```

将来，可以产生如下的确认报警请求：

```
X ‘00’          PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0,
                SA=0)
X ‘02’          Maximum APDU Size Accessed = 206 octets
X ‘07’          Invoke ID = 7
X ‘00’          Service Choice = 0 (AcknowledgeAlarm-Request)
X ‘09’          SD Context Tag 0 (Acknowledging Process Identifier, L=1)
```

```
X ‘01’          Acknowledging Process Identifier =1
X ‘1C’          SD Context Tag 1 (Event Object Identifier, L=4)
X ‘00000002’        Analog Input, Instance Number = 2
X ‘29’          SD Context Tag 2 (Event State Identifier, L=1)
X ‘03’          3 (HIGH_LIMIT)
X ‘3E’          PD Opening Tag 3 (Time Stamp)
        X ‘19’          SD Context Tag 1 (Sequence Number, L=1)
        X ‘10’          16
X ‘3F’          PD Closing Tag 3 (Time Stamp)
X ‘4C’          SD Context Tag 4 (Acknowledgment Source, L=4)
X ‘00’              ANSI X3.4 Encoding
X ‘4D444C’          “MDL”
X ‘5E’          PD Opening Tag 5 (Time Of Acknowledgment)
        X ‘2E’          PD Opening Tag 2 (Date Time)
            X ‘A4’          Application Tag 10 (Date, L=4)
            X ‘5C0615FF’        June 21, 1992
            X ‘B4’          Application Tag 11 (Time, L=4)
            X ‘0D032909’        13:03:41.9
        X ‘2F’          PD Closing Tag 2 (Date Time)
X ‘5F’          PD Closing Tag 5 (Time Of Acknowledgment)
```

假定服务进程执行正确，那么将返回一个简单确认：

```
X ‘20’          PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘07’          Invoke ID = 7
X ‘00’          Service ACK Choise = 0 (AcknowledgeAlarm)
```

F.1.4 例 E.1.4 编码——获得报警摘要服务

```
X ‘00’          PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0,
                SA=0)
X ‘02’          Maximum APDU Size Accessed = 206 octets
X ‘01’          Invoke ID = 1
X ‘03’          Service Choice = 3 (GetAlarmSummary-Request)
```

假定服务进程执行正确，那么将返回一个复杂确认：

X ‘30’	PDU Type = 3 (BACnet-ComplexACK-PDU, SEG=0,MOR=0)
X ‘01’	Invoke ID = 1
X ‘03’	Service ACK Choise = 3 (GetAlarmSummary-ACK)
X ‘C4’	Application Tag 12 (Object Identifier, L=4) (Object Identifier)
X ‘00000002’	Analog Input, Instance Number = 2
X ‘91’	Application Tag 9 (Enumerated, L=1) (Alarm State)
X ‘03’	3 (HIGH_LIMIT)
X ‘82’	Application Tag 8 (Bit String, L=2) (Acknowledged Transitions)
X ‘0560’	0, 1, 1 (FALSE, TRUE, TRUE)
X ‘C4’	Application Tag 12 (Object Identifier, L=4) (Object Identifier)
X ‘00000003’	Analog Input, Instance Number = 3
X ‘91’	Application Tag 9 (Enumerated, L=1) (Alarm State)
X ‘04’	3 (LOW_LIMIT)
X ‘82’	Application Tag 8 (Bit String, L=2) (Acknowledged Transitions)
X ‘05E0’	1, 1, 1 (TRUE, TRUE ,TRUE)

F.1.5 例 E.1.5 编码——获得登记摘要服务

例 1：请求编码

X ‘00’	PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X ‘02’	Maximum APDU Size Accessed = 206 octets
X ‘01’	Invoke ID = 1
X ‘04’	Service Choice = 4 (GetEnrollmentSummary-Request)
X ‘09’	SD Context Tag 0 (Acknowledgment Filter, L=1)
X ‘02’	2 (NOT_ACKED)

假定服务进程执行正确，那么将返回一个包含答复的复杂确认：

X ‘30’ PDU Type = 3 (BACnet-ComplexACK-PDU, SEG=0,MOR=0)
X ‘01’ Invoke ID = 1
X ‘04’ Service ACK Choise = 4 (GetEnrollmentSummary-ACK)
X ‘C4’ Application Tag 12 (Object Identifier, L=4) (Object Identifier)
X ‘00000002’ Analog Input, Instance Number = 2
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event Type)
X ‘05’ 5 (OUT _OF_RANGE)
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event State)
X ‘03’ 3 (HIGH_LIMIT)
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Priority)
X ‘64’ 100
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Notification Class)
X ‘04’ 4
X ‘C4’ Application Tag 12 (Object Identifier, L=4) (Object Identifier)
X ‘02400006’ Event Enrollment, Instance Number = 6
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event Type)
X ‘01’ 1 (CHANGE_OF_STATE)
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event State)
X ‘00’ 0 (NORMAL)
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Priority)
X ‘32’ 50
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Notification Class)
X ‘02’ 2

例 2：请求编码

X ‘00’ PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X ‘02’ Maximum APDU Size Accessed = 206 octets
X ‘02’ Invoke ID = 2

```
X ‘04’          Service Choice = 4 (GetEnrollmentSummary-Request)
X ‘09’          SD Context Tag 0 (Acknowledgment Filter, L=1)
X ‘00’          0 (ALL)
X ‘1E’          PD Opening Tag 1 (Enrollment Filter)
        X ‘0E’          PD Opening Tag 0 (Recipient)
            X ‘0C’          SD Context Tag 0 (Device, L=4)
            X ‘02000011’        Device, Instance Number = 17
        X ‘0F’          PD Closing Tag 0 (Recipient)
        X ‘19’          SD Context Tag 1 (Process Identifier)
        X ‘09’          9
X ‘1F’          PD Closing Tag 1 (Enrollment Filter)
X ‘4E’          PD Opening Tag 4 (Priority Filter)
        X ‘09’          SD Context Tag 0 (MinPriority, L=1)
        X ‘06’          6
        X ‘19’          SD Context Tag 1 (MaxPriority, L=1)
        X ‘0A’          10
X ‘4F’          PD Closing Tag 4 (Priority Filter)
```

假定服务进程执行正确，那么将返回一个包含答复的复杂确认：

```
X ‘30’          PDU Type = 3 (BACnet-ComplexACK-PDU, SEG=0,MOR=0)
X ‘02’          Invoke ID = 2
X ‘04’          Service ACK Choise = 4 (GetEnrollmentSummary-ACK)
X ‘C4’          Application Tag 12 (Object Identifier, L=4) (Object
Identifier)
X ‘00000002’        Analog Input, Instance Number = 2
X ‘91’          Application Tag 9 (Enumerated, L=1) (Event Type)
X ‘05’          5 (OUT _OF_RANGE)
X ‘91’          Application Tag 9 (Enumerated, L=1) (Event State)
X ‘00’          0 (NORMAL)
X ‘21’          Application Tag 2 (Unsigned Integer, L=1) (Priority)
X ‘08’          8
X ‘21’          Application Tag 2 (Unsigned Integer, L=1) (Notification
Class)
X ‘04’          4
```

X ‘C4’ Application Tag 12 (Object Identifier, L=4) (Object Identifier)
X ‘00000003’ Analog Input, Instance Number = 3
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event Type)
X ‘05’ 5 (OUT _OF_RANGE)
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event State)
X ‘00’ 0 (NORMAL)
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Priority)
X ‘08’ 8
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Notification Class)
X ‘02’ 2
X ‘C4’ Application Tag 12 (Object Identifier, L=4) (Object Identifier)
X ‘00000004’ Analog Input, Instance Number = 4
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event Type)
X ‘05’ 5 (OUT _OF_RANGE)
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event State)
X ‘00’ 0 (NORMAL)
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Priority)
X ‘08’ 8
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Notification Class)
X ‘02’ 2
X ‘C4’ Application Tag 12 (Object Identifier, L=4) (Object Identifier)
X ‘02400007’ Even Enrollment, Instance Number = 7
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event Type)
X ‘04’ 4 (FLOATING_LIMIT)
X ‘91’ Application Tag 9 (Enumerated, L=1) (Event State)
X ‘00’ 0 (NORMAL)
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Priority)
X ‘03’ 3
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Notification Class)

X ‘08’ 8

F.1.6 例 E.1.6 编码——预订 COV 服务

X ‘00’ PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X ‘02’ Maximum APDU Size Accessed = 206 octets
X ‘0F’ Invoke ID = 15
X ‘05’ Service Choice = 5 (SubscribeCOV-Request)
X ‘09’ SD Context Tag 0 (Subscriber Process Identifier, L=1)
X ‘12’ 18
X ‘1C’ SD Context Tag 1 (Monitored Object Identifier, L=4)
X ‘0000000A’ Analog Input, Instance Number = 10
X ‘29’ SD Context Tag 2 (Issue Confirmed Notifications, L=1)
X ‘01’ 1 (TRUE)
X ‘39’ SD Context Tag 3 (Lifetime, L=1)
X ‘00’ 0

假定服务进程执行正确，那么将返回一个简单确认：

X ‘20’ PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘0F’ Invoke ID = 15
X ‘05’ Service ACK Choise = 5 (SubscribeCOV)

F.1.7 例 E.1.7 编码——无证实 COV 通告服务

X ‘10’ PDU Type=1 (BACnet-Unconfirmed-Request-PDU)
X ‘02’ Service Choise = 2 (UnconfirmedCOVNotification-Request)
X ‘09’ SD Context Tag 0 (Subscriber Process Identifier, L=1)
X ‘12’ 18
X ‘1C’ SD Context Tag 1 (Initiating Device Identifier, L=4)
X ‘02000004’ Device, Instance Number = 4
X ‘2C’ SD Context Tag 2 (Monitored Object Identifier, L=4)
X ‘0000000A’ Analog Input, Instance Number = 10
X ‘39’ SD Context Tag 3 (Time Remaining, L=1)

```
X ‘00’          0
X ‘4E’          PD Opening Tag 4 (List Of Values)
        X ‘09’          SD Context Tag 0 (Property Identifier, L=1)
        X ‘55’          85 (PRESENT_VALUE)
        X ‘2E’          PD Opening Tag 2 (Value)
            X ‘44’          Application Tag 4 (Real, L=4)
            X ‘42820000’        65.0
        X ‘2F’          PD Closing Tag 2 (Value)
        X ‘09’          SD Context Tag 0 (Property Identifier, L=1)
        X ‘6F’          111 (STATUS_FLAGS)
        X ‘2E’          PD Opening Tag 2 (Value)
            X ‘82’          Application Tag 8 (Bit String, L=2)
            X ‘0400’            0, 0, 0, 0 (FALSE, FALSE, FALSE, FALSE)
        X ‘2F’          PD Closing Tag 2 (Value)
X ‘4F’          PD Closing Tag 4 (List Of Values)
```

F.1.8 例 E.1.8 编码——无证实事件通告服务

```
X ‘10’          PDU Type=1 (BACnet-Unconfirmed-Request-PDU)
X ‘03’          Service Choice = 3 (UnconfirmedEventNotification-Request)
X ‘09’          SD Context Tag 0 (Process Identifier, L=1)
X ‘01’          1
X ‘1C’          SD Context Tag 1 (Initiating Device Identifier, L=4)
X ‘02000009’        Device, Instance 9
X ‘2C’          SD Context Tag 2 (Event Object Identifier, L=4)
X ‘00000002’        Analog Input, Instance Number = 2
X ‘3E’          PD Opening Tag 3 (Time Stamp)
        X ‘19’          SD Context Tag 1 (SequenceNumber, L=1)
        X ‘10’          16
X ‘3F’          PD Closing Tag 3 (Time Stamp)
X ‘49’          SD Context Tag 4 (Notification Class, L=1)
X ‘04’          4
X ‘59’          SD Context Tag 5 (Priority, L=1)
X ‘64’          100
X ‘69’          SD Context Tag 6 (Event Type, L=1)
```

```
X ‘05’          5 (OUT_OF_RANGE)
X ‘89’          SD Context Tag 8 (Notify Type, L=1)
X ‘00’          0 (ALARM)
X ‘99’          SD Context Tag 9 (AckedRequired, L=1)
X ‘01’          TRUE
X ‘A9’          SD Context Tag 10 (From State, L=1)
X ‘00’          0 (NORMAL)
X ‘B9’          SD Context Tag 11 (To State, L=1)
X ‘03’          3 (HIGH_LIMIT)
X ‘CE’          PD Opening Tag 12 (Event Values)
        X ‘5E’          PD Opening Tag 5 (BACnetNotificationParameters,
                                          Choice 5=OUT_OF_RANGE)
            X ‘0C’          SD Context Tag 0 (Exceeding Value, L=4)
            X ‘42A03333’        80.1
            X ‘1A’          SD Context Tag 1 (Status Flags, L=2)
            X ‘0480’            1, 0, 0, 0 (TRUE, FALSE, FALSE, FALSE)
            X ‘2C’          SD Context Tag 2 (Deadband, L=4)
            X ‘3F800000’        1.0
            X ‘3C’          SD Context Tag 3 (Exceed Limit, L=4)
            X ‘42A00000’        80.0
        X ‘5F’          PD Closing Tag 5 (BACnetNotificationParameters)
X ‘CF’          PD Closing Tag 12 (Event Values)
```

将来，可以产生如下的确认报警请求：

```
X ‘00’          PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0,
                SA=0)
X ‘02’          Maximum APDU Size Accessed = 206 octets
X ‘07’          Invoke ID = 7
X ‘00’          Service Choice = 0 (AcknowledgeAlarm-Request)
X ‘09’          SD Context Tag 0 (Acknowledging Process Identifier, L=1)
X ‘01’          Acknowledging Process Identifier =1
X ‘1C’          SD Context Tag 1 (Event Object Identifier, L=4)
X ‘00000002’        Analog Input, Instance Number = 2
X ‘29’          SD Context Tag 2 (Event State Identifier, L=1)
```

X ‘03’ 3 (HIGH_LIMIT)
X ‘3E’ PD Opening Tag 3 (Time Stamp)
X ‘19’ SD Context Tag 1 (Sequence Number, L=1)
X ‘10’ 16
X ‘3F’ PD Closing Tag 3 (Time Stamp)
X ‘4C’ SD Context Tag 4 (Acknowledgment Source, L=4)
X ‘00’ ANSI X3.4 Encoding
X ‘4D444C’ “MDL”
X ‘5E’ PD Opening Tag 5 (Time Of Acknowledgment)
X ‘2E’ PD Opening Tag 2 (Date Time)
X ‘A4’ Application Tag 10 (Date, L=4)
X ‘5C0615FF’ June 21, 1992
X ‘B4’ Application Tag 11 (Time, L=4)
X ‘0D032909’ 13:03:41.9
X ‘2F’ PD Closing Tag 2 (Date Time)
X ‘5F’ PD Closing Tag 5 (Time Of Acknowledgment)

假定服务进程执行正确，那么将返回一个简单确认：

X ‘20’ PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘07’ Invoke ID = 7
X ‘00’ Service ACK Choise = 0 (AcknowledgeAlarm)

F.2 文件访问服务编码例程

F.2.1 例 E.2.1 编码——基本读文件服务

例 1：从文件中读数据

X ‘00’ PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X ‘02’ Maximum APDU Size Accessed = 206 octets
X ‘00’ Invoke ID = 0
X ‘06’ Service Choice = 6 (AtomicReadFile-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L=4) (File Identifier)
X ‘02800001’ File, Instance Number = 1
X ‘0E’ PD Opening Tag 0 (Stream Access)
X ‘31’ Application Tag 3 (Signed Integer, L=1) (File Start Position)
X ‘00’ 0
X ‘21’ PD Opening Tag 0 (Unsigned, L=1) (Requested Octet Count)
X ‘1B’ 27
X ‘0F’ PD Closing Tag 0 (Stream Access)

假定服务进程执行正确，那么将返回一个复杂确认：

X ‘30’ PDU Type=3 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X ‘00’ Invoke ID = 0
X ‘06’ Service Choice = 6 (AtomicReadFile-Request)

X ‘10’ Application Tag 1 (Boolean, 0 (FALSE)) (End Of File)
X ‘0E’ PD Opening Tag 0 (Stream Access)
X ‘31’ Application Tag 3 (Signed Integer, L=1) (File Start Position)
X ‘00’ 0
X ‘65’ Application Tag 6 (Octet String, L > 4)
X ‘1B’ Extended Length = 27
X ‘436869C6C657230312041F6E2D54696D653D342E3320486F757273’ “Chiller01 On-Time = 4.3 Hours”
X ‘0F’ PD Closing Tag 0 (Stream Access)

例 2：从文件中读记录

X ‘00’ PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X ‘02’ Maximum APDU Size Accessed = 206 octets

X ‘12’ Invoke ID = 18
X ‘06’ Service Choice = 6 (AtomicReadFile-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L=4) (File Identifier)
X ‘02800002’ File, Instance Number = 2
X ‘1E’ PD Opening Tag 1 (Record Access)
X ‘31’ Application Tag 3 (Signed Integer, L=1) (File Start Record)
X ‘0E’ 14
X ‘21’ Application Tag 2 (Signed Integer, L=1) (File Start Count)
X ‘03’ 3
X ‘1F’ PD Closing Tag 1 (Record Access)

假定服务进程执行正确，那么将返回一个复杂确认：

X ‘30’ PDU Type=3 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0)
X ‘12’ Invoke ID = 18
X ‘06’ Service Choice = 6 (AtomicReadFile-ACK)

X ‘11’ Application Tag 1 (Boolean, 1 (TRUE)) (End Of File)
X ‘1E’ PD Opening Tag 1 (Record Access)
X ‘31’ Application Tag 3 (Signed Integer, L=1) (File Start Record)
X ‘0E’ 14
X ‘21’ Application Tag 2 (Unsigned, L=1) (Returned Record Count)
X ‘02’ 2
X ‘65’ Application Tag 6 (Octet String, L > 4) (File Record Data)
X ‘0A’ Extended Length = 10
X ‘31323A30302C34352E36’ “12:00, 45.6”
X ‘65’ Application Tag 6 (Octet String, L > 4) (File Record Data)

X ‘0A’ Extended Length = 10

X ‘31323A31352C34342E38’ “12:15, 44.8”

X ‘1F’ PD Closing Tag 1 (Record Access)

F.2.2 例 E.2.2 编码——基本写文件服务

例 1：向文件中写数。

X ‘00’ PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)

X ‘02’ Maximum APDU Size Accessed = 206 octets

X ‘55’ Invoke ID = 85

X ‘07’ Service Choice = 7 (AtomicWriteFile-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L=4) (File Identifier)

X ‘02800001’ File, Instance Number = 1

X ‘0E’ PD Opening Tag 0 (Stream Access)

X ‘31’ Application Tag 3 (Signed Integer, L=1) (File Start Position)

X ‘1E’ 30

X ‘65’ Application Tag 6 (Octet String, L > 4) (File Data)

X ‘1B’ Extended Length = 27

X ‘4368696C6C65723031204F6E2D54696D653D342E3320486F757273’

“12:15, 44.8”

X ‘0F’ PD Closing Tag 0 (Record Access)

假定服务进程执行正确，那么将返回一个复杂确认：

X ‘30’ PDU Type=3 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0)

X ‘55’ Invoke ID = 85

X ‘07’ Service Choice = 7 (AtomicWriteFile-ACK)

X ‘09’ SD Context Tag 0 (File Start Position, L=1)

X ‘1E’ 30

例 2：向文件中追加两个记录

X ‘00’ PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X ‘02’ Maximum APDU Size Accessed = 206 octets
X ‘55’ Invoke ID = 85
X ‘07’ Service Choice = 7 (AtomicWriteFile-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L=4) (File Identifier)
X ‘02800002’ File, Instance Number = 2
X ‘1E’ PD Opening Tag 1 (Record Access)
X ‘31’ Application Tag 3 (Signed Integer, L=1) (File Start Record)
X ‘FF’ -1 (Append to End of File)
X ‘21’ Application Tag 2 (Unsigned Integer, L=1) (Record Count)
X ‘02’ 2
X ‘65’ Application Tag 6 (Octet String, L > 4) (File Record Data)
X ‘0A’ Extended Length = 10
X ‘31323A30302C34352E36’ “12:00, 45.6”
X ‘65’ Application Tag 6 (Octet String, L > 4) (File Record Data)
X ‘0A’ Extended Length = 10
X ‘31323A31352C34342E38’ “12:15, 44.8”
X ‘1F’ PD Closing Tag 1 (Record Access)

假定服务进程执行正确，那么将返回一个复杂确认：

X ‘30’ PDU Type=3 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0)
X ‘55’ Invoke ID = 85
X ‘07’ Service ACK Choice = 7 (AtomicWriteFile-ACK)
X ‘19’ SD Context Tag 1 (File Start Position, L=1)
X ‘0E’ 14

F.3 对象访问服务编码例程

F.3.1 例 E.3.1 编码——添加列表元素服务

```
X ‘00’          PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
                = 0, SA = 0)
X ‘02’          Maximum APDU Size Accepted = 206 octets
X ‘01’          Invoke ID = 1
X ‘08’          Service Choice = 8 (AddListElement-Request)

X ‘0C’          SD Context Tag 0 (Object Identifier, L = 4)
X ‘02C00003’    Group, Instance Number = 3
X ‘19’          SD Context Tag 1 (Property Identifier, L = 1)
X ‘35’          53 (LIST_OF_GROUP_MEMBERS)
X ‘3E’          PD Opening Tag 3 (List Of Elements)
    X ‘0C’          SD Context Tag 0 (Object Identifier, L = 4)
    X ‘0000000F’    85 (PRESENT_VALUE)
    X ‘1E’          PD Opening Tag 1 (List Of Property References)
        X ‘09’          SD Context Tag 0 (Property Identifier, L = 1)
        X ‘55’          85 (PRESENT_VALUE)
        X ‘09’          SD Context Tag 0 (Property Identifier, L = 1)
        X ‘67’          103 (RELIABILITY)
    X ‘1F’          PD Opening Tag 1 (List Of Property References)
X ‘3F’          PD Closing Tag 3 (List Of Elements)
```

假定服务进程执行正确，那么将返回一个简单确认：

```
X ‘20’          PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘01’          Invoke ID = 1
X ‘08’          Service ACK Choice = 8 (AddListElement)
```

F.3.2 例 F.3.2 编码——删除列表元素服务

```
        X  ‘00’              PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                        = 0, SA = 0)
        X  ‘02’              Maximum APDU Size Accepted = 206 octets
        X  ‘34’              Invoke ID = 52
        X  ‘09’              Service Choice = 9 (RemoveListElement-Request)

        X  ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
        X  ‘02C00003’        Group, Instance Number = 3
        X  ‘19’              SD Context Tag 1 (Property Identifier, L = 1)
        X  ‘35’              53 (LIST_OF_GROUP_MEMBERS)
        X  ‘3E’              PD Opening Tag 3 (List Of Elements)
                X  ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
                X  ‘0000000C’        Analog Input, Instance Number = 12
                X  ‘1E’              PD Opening Tag 1 (List Of Property References)
                        X  ‘09’          SD Context Tag 0 (Property Identifier, L
        = 1)
                        X  ‘55’          85 (PRESENT_VALUE)
                        X  ‘09’          SD Context Tag 0 (Property Identifier, L
        = 1)
                        X  ‘67’          103 (RELIABILITY)
                        X  ‘09’          SD Context Tag 0 (Property Identifier, L
        = 1)
                        X  ‘1C’          28 (DESCRIPTION)
                X  ‘1F’              PD Opening Tag 1 (List Of Property References)

                X  ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
                X  ‘0000000D’        Analog Input, Instance Number = 13
                X  ‘1E’              PD Opening Tag 1 (List Of Property References)
                        X  ‘09’          SD Context Tag 0 (Property Identifier, L
        = 1)
```

```
                X  '55'          85 (PRESENT_VALUE)
                X  '09'          SD Context Tag 0 (Property Identifier, L
= 1)
                X  '67'          103 (RELIABILITY)
                X  '09'          SD Context Tag 0 (Property Identifier, L
= 1)
                X  '1C'          28 (DESCRIPTION)
          X  '1F'              PD Closing Tag 1 (List Of Property References)
X  '3F'              PD Closing Tag 3 (List Of Elements)
```

假定服务进程执行正确，那么将返回一个简单确认：

```
X  '20'              PDU Type = 2 (BACnet-SimpleACK-PDU)
X  '34'              Invoke ID = 52
X  '09'              Service ACK Choice = 9 (AddListElement)
```

本例的第二部分重新插入上面删除的三个元素中的两个：

```
X  '00'              PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                = 0, SA = 0)
X  '02'              Maximum APDU Size Accepted = 206 octets
X  '35'              Invoke ID = 53
X  '08'              Service Choice = 8 (AddListElement -Request)

X  '0C'              SD Context Tag 0 (Object Identifier, L = 4)
X  '02C00003'        Group, Instance Number = 3
X  '19'              SD Context Tag 1 (Property Identifier, L = 1)
X  '35'              53 (LIST_OF_GROUP_MEMBERS)
X  '3E'              PD Opening Tag 3 (List Of Elements)
          X  '0C'              SD Context Tag 0 (Object Identifier, L = 4)
          X  '0000000C'        Analog Input, Instance Number = 12
          X  '1E'              PD Opening Tag 1 (List Of Property References)
                X  '09'          SD Context Tag 0 (Property Identifier, L
= 1)
                X  '55'          85 (PRESENT_VALUE)
```

```
                X  '09'          SD Context Tag 0 (Property Identifier, L
= 1)
                X  '67'          103 (RELIABILITY)
          X  '1F'              PD Opening Tag 1 (List Of Property References)

          X  '0C'              SD Context Tag 0 (Object Identifier, L = 4)
          X  '0000000D'        Analog Input, Instance Number = 13
          X  '1E'              PD Opening Tag 1 (List Of Property References)
                X  '09'          SD Context Tag 0 (Property Identifier, L
= 1)
                X  '55'          85 (PRESENT_VALUE)
                X  '09'          SD Context Tag 0 (Property Identifier, L
= 1)
                X  '67'          103 (RELIABILITY)
          X  '1F'              PD Closing Tag 1 (List Of Property References)
     X  '3F'              PD Closing Tag 3 (List Of Elements)
```

假定服务进程执行正确，那么将返回一个简单确认：

```
     X  '20'              PDU Type = 2 (BACnet-SimpleACK-PDU)
     X  '35'              Invoke ID = 53
     X  '08'              Service ACK Choice = 8 (AddListElement)
```

F.3.3 例 F.3.3 编码——创建对象服务

```
     X  '00'              PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                     = 0, SA = 0)
     X  '04'              Maximum APDU Size Accepted = 1024 octets
     X  '56'              Invoke ID = 86
     X  '0A'              Service Choice = 10 (CreateObject -Request)

     X  '0E'              PD Opening Tag 0 (Object Specifier)
          X  '09'              SD Context Tag 0 (Property Identifier, L = 1)
          X  '0A'              10 (File Object)
```

X ‘0F’ PD Closing Tag 0 (Object Specifier)
X ‘1E’ PD Opening Tag 1 (List Of Initial Values)
X ‘09’ SD Context Tag 0 (Property Identifier, L = 1)
X ‘4D’ 77 (OBJECT_NAME)
X ‘2E’ PD Opening Tag 2 (Value)
X ‘75’ Application Tag 7 (Character String, L>4)
X ‘08’ Extended Length = 8
X ‘00’ ANSI X3.4 Encoding
X ‘54726566642031’ “Trend 1”
X ‘2F’ PD Closing Tag 2 (Value)
X ‘09’ SD Context Tag 0 (Property Identifier, L = 1)
X ‘29’ 41 (FILE_ACCESS_METHOD)
X ‘2E’ PD Opening Tag 2 (Value)
X ‘91’ Application Tag 9 (Enumerated, L = 1)
X ‘00’ 0 (RECORD_ACCESS)
X ‘2F’ PD Closing Tag 2 (Value)
X ‘1F’ PD Opening Tag 1 (List Of Initial Values)

假定服务进程执行正确，那么将返回一个传送新建对象标识符的确认：

X ‘30’ PDU Type = 3 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR = 0)
X ‘56’ Invoke ID = 86
X ‘0A’ Service ACK Choice = 10 (CreateObject -ACK)
X ‘C4’ Application Tag 12 (Object Identifier, L = 4)
X ‘0280000D’ File, Instance Number = 13

F.3.4 例 F.3.4 编码——删除对象服务

例 1：成功删除对象

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘04’ Maximum APDU Size Accepted = 1024 octets

X ‘57’ Invoke ID = 87
X ‘0B’ Service Choice = 11 (DeleteObject -Request)
X ‘C4’ Application Tag 12 (Object Identifier, L = 4)
X ‘02C00006’ Group, Instance Number = 6

假定服务进程执行正确，那么将返回一个简单确认：

X ‘20’ PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘57’ Invoke ID = 87
X ‘0B’ Service ACK Choice = 11 (DeleteObject)

例 2：删除对象的不成功尝试。

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘04’ Maximum APDU Size Accepted = 1024 octets
X ‘58’ Invoke ID = 88
X ‘0B’ Service Choice = 11 (DeleteObject -Request)

X ‘C4’ Application Tag 12 (Object Identifier, L = 4)
X ‘02C00007’ Group, Instance Number = 7

在本例中，假定对象被保护，不能被此协议服务删除。服务方将发出如下响应：

X ‘50’ PDU Type = 5 (BACnet-Error-PDU)
X ‘58’ Original Invoke ID = 88
X ‘0B’ Error Choice = 11(DeleteObject)
X ‘91’ Application Tag 9 (Enumerated, L = 1) (Error Class)
X ‘01’ 1 (OBJECT)
X ‘91’ Application Tag 9 (Enumerated, L = 1) (Error Code)
X ‘17’ 23 (OBJECT_DELETION_NOT_PERMITTED)

F.3.5 例 F.3.5 编码——读属性服务

```
X ‘00’              PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                    = 0, SA = 0)
X ‘00’              Maximum APDU Size Accepted = 50 octets
X ‘01’              Invoke ID = 1
X ‘0C’              Service Choice = 12 (ReadProperty -Request)

X ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
X ‘00000005’        Analog Input, Instance Number = 5
X ‘19’              SD Context Tag 1 (Property Identifier, L = 1)
X ‘55’              85 (PRESENT_VALUE)
```

此请求产生如下结果：

```
X ‘30’              PDU Type = 3 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR=
0)
X ‘01’              Invoke ID = 1
X ‘0C’              Service ACK Choice = 12 (ReadProperty -ACK)

X ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
X ‘00000005’        Analog Input, Instance Number = 5
X ‘19’              SD Context Tag 1 (Property Identifier, L = 1)
X ‘55’              85 (PRESENT_VALUE)
X ‘3E’              PD Opening Tag 3 (Property Value)
        X ‘44’              Application Tag 4 (Real, L = 4)
        X ‘4290999A’        72.3
X ‘3F’              PD Closing Tag 3 (Property Value)
```

F.3.6 例 F.3.6 编码——条件读属性服务

例 1：获取所有数字输入对象的对象标识符。

```
X ‘00’              PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                    = 0, SA = 0)
```

```
X  '04'              Maximum APDU Size Accepted = 1024 octets
X  '51'              Invoke ID = 81
X  '0D'              Service Choice = 13 (ReadPropertyConditional Request)

X  '0E'              PD Opening Tag 0 (Object Selection Criteria)
        X  '09'              SD Context Tag 0 (Selection Logic, L = 1)
        X  '00'              0 (AND)
        X  '1E'              PD Opening Tag 1 (List Of Selection Criteria)
                X  '09'          SD Context Tag 0 (Selection Logic, L = 1)
                X  '4F'          79 (OBJECT_TYPE)
                X  '29'          SD Context Tag 2 (Relation Specifier)
                X  '00'          0 (EQUAL)
                X  '3E'          PD Opening Tag 3 (Comparison Value)
                    X  '91'          Application Tag 9 (Enumerated, L
= 1)
                                     (Object Type)
                    X  '03'          3 (BINARY_INPUT)
                X  '3F'          PD Closing Tag 3 (Comparison Value)
        X  '1F'              PD Closing Tag 1 (List Of Selection Criteria)
X  '0F'              PD Closing Tag 0 (Object Selection Criteria)
```

此请求产生如下结果：

```
X  '30'              PDU Type = 3 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR=
0)
X  '51'              Invoke ID = 81
X  '0D'              Service ACK Choice = 13 (ReadPropertyConditional-ACK)

X  '0C'              SD Context Tag 0 (Object Identifier, L = 4)
X  '00C00001'        Binary Input, Instance Number = 1

X  '0C'              SD Context Tag 0 (Object Identifier, L = 4)
X  '00C00002'        Binary Input, Instance Number = 2
```

```
X ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
X ‘00C00003’        Binary Input, Instance Number = 3

X ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
X ‘00C00004’        Binary Input, Instance Number = 4
```

例 2：模拟输入的条件读

```
X ‘00’              PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                    = 0, SA = 0)
X ‘04’              Maximum APDU Size Accepted = 1024 octets
X ‘52’              Invoke ID = 82
X ‘0D’              Service Choice = 13 (ReadPropertyConditional Request)

X ‘0E’              PD Opening Tag 0 (Object Selection Criteria)
        X ‘09’              SD Context Tag 0 (Selection Logic, L = 1)
        X ‘00’              Enumeration 0 (AND)
        X ‘1E’              PD Opening Tag 1 (List Of Selection Criteria)
                X ‘09’          SD Context Tag 0 (Selection Logic, L = 1)
                X ‘4F’          79 (OBJECT_TYPE)
                X ‘29’          SD Context Tag 2 (Relation Specifier)
                X ‘00’          0 (EQUAL)
                X ‘3E’          PD Opening Tag 3 (Comparison Value)
                        X ‘91’          Application Tag 9 (Enumerated, L
= 1)
                                        (Object Type)
                        X ‘00’          0 (ANALOG_INPUT)
                X ‘3F’          PD Closing Tag 3 (Comparison Value)

                X ‘09’          SD Context Tag 0 (Property Identifier, L
= 1)
                X ‘55’          85 (PRESENT_VALUE)
                X ‘29’          SD Context Tag 2 (Relation Specifier)
                X ‘03’          3 (GREATER_THAN)
```

```
                X  '3E'          PD Opening Tag 3 (Comparison Value)
                     X  '44'          Application Tag 4 (Real, L = 4)
                     X  '42C80000'    100.0
                X  '3F'          PD Closing Tag 3 (Comparison Value)
          X  '1F'              PD Closing Tag 1 (List Of Selection Criteria)
     X  '0F'              PD Closing Tag 0 (Object Selection Criteria)
```

此请求产生如下结果：

```
     X  '30'              PDU Type = 3 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR=
0)
     X  '52'              Invoke ID = 82
     X  '0D'              Service ACK Choice = 13 (ReadPropertyConditional-ACK)

     X  '0C'              SD Context Tag 0 (Object Identifier, L = 4)
     X  '00000001'        Analog Input, Instance Number = 1
     X  '1E'              PD Opening Tag 1 (List Of Results)
          X  '29'              SD Context Tag 2 (Property Identifier, L = 1)
          X  '55'              85 (PRESENT_VALUE)
          X  '4E'              PD Opening Tag 4 (Property Value)
               X  '44'          Application Tag 4 (Property Value)
             X  '42CC999A'      102.3
          X  '4F'              PD Closing Tag 4 (Property Value)
     X  '1F'              PD Opening Tag 1 (List Of Results)

     X  '0C'              SD Context Tag 0 (Object Identifier, L = 4)
     X  '00000002'        Analog Input, Instance Number = 2
     X  '1E'              PD Opening Tag 1 (List Of Results)
          X  '29'              SD Context Tag 2 (Property Identifier, L = 1)
          X  '55'              85 (PRESENT_VALUE)
          X  '4E'              PD Opening Tag 4 (Property Value)
               X  '44'          Application Tag 4 (Property Value)
             X  '4334B333'      180.7
          X  '4F'              PD Closing Tag 4 (Property Value)
     X  '1F'              PD Opening Tag 1 (List Of Results)
```

```
X  '0C'              SD Context Tag 0 (Object Identifier, L = 4)
X  '00000003'        Analog Input, Instance Number = 3
X  '1E'              PD Opening Tag 1 (List Of Results)
        X  '29'              SD Context Tag 2 (Property Identifier, L = 1)
        X  '55'              85 (PRESENT_VALUE)
        X  '4E'              PD Opening Tag 4 (Property Value)
                X  '44'          Application Tag 4 (Property Value)
                X  '430E199A'    142.1
        X  '4F'              PD Closing Tag 4 (Property Value)
X  '1F'              PD Opening Tag 1 (List Of Results)
```

例 3：查找不可靠的或脱离服务的对象。

```
    X  '00'              PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                    = 0, SA = 0)
    X  '04'              Maximum APDU Size Accepted = 1024 octets
    X  '53'              Invoke ID = 83
    X  '0D'              Service Choice = 13 (ReadPropertyConditional Request)

    X  '0E'              PD Opening Tag 0 (Object Selection Criteria)
            X  '09'              SD Context Tag 0 (Selection Logic, L = 1)
            X  '01'              1 (OR)
            X  '1E'              PD Opening Tag 1 (List Of Selection Criteria)
                    X  '09'          SD Context Tag 0 (Selection Logic, L = 1)
                    X  '67'          103 (RELIABILITY)
                    X  '29'          SD Context Tag 2 (Relation Specifier)
                    X  '01'          1 (NOT_EQUAL)
                    X  '3E'          PD Opening Tag 3 (Comparison Value)
                            X  '91'          Application Tag 9 (Enumerated, L
    = 1)
                            X  '00'          0 (ANALOG_INPUT)
                    X  '3F'          PD Closing Tag 3 (Comparison Value)
```

```
                X  '09'          SD Context Tag 0 (Property Identifier, L
= 1)
                X  '51'          81 (OUT_OF_SERVICE)
                X  '29'          SD Context Tag 2 (Relation Specifier)
                X  '00'          0 (EQUAL)
                X  '3E'          PD Opening Tag 3 (Comparison Value)
                     X  '11'          Application  Tag  1  (Boolean,
1(TRUE))
                X  '3F'          PD Closing Tag 3 (Comparison Value)
          X  '1F'             PD Closing Tag 1 (List Of Selection Criteria)
X  '0F'             PD Closing Tag 0 (Object Selection Criteria)

X  '1E'             PD Opening Tag 1 (List Of Property References)
          X  '09'             SD Context Tag 0 (Property Identifier, L = 1)
          X  '55'             85 (PRESENT_VALUE)
          X  '09'             SD Context Tag 0 (Property Identifier, L = 1)
          X  '67'             103 (RELIABILITY)
          X  '09'             SD Context Tag 0 (Property Identifier, L = 1)
          X  '51'             81 (OUT_OF_SERVICE)
X  '1F'             PD Closing Tag 1 (Object Property References)
```

假定服务进程执行正确，那么将返回一个包含请求值的复杂确认：

```
X  '30'             PDU Type = 3 (BACnet-ConfirmedACK-PDU, SEG = 0, MOR=
0)
X  '53'             Invoke ID = 83
X  '0D'             Service ACK Choice = 13 (ReadPropertyConditional-ACK)

X  '0C'             SD Context Tag 0 (Object Identifier, L = 4)
X  '00C00001'       Binary Input, Instance Number = 1
X  '1E'             PD Opening Tag 1 (List Of Results)
          X  '29'             SD Context Tag 2 (Property Identifier, L = 1)
          X  '55'             85 (PRESENT_VALUE)
          X  '4E'             PD Opening Tag 4 (Property Value)
                X  '91'          Application Tag 9 (Enumerated, L = 1)
```

```
                X  '01'          1 (ACTIVE)
        X  '4F'              PD Closing Tag 4 (Property Value)

        X  '29'              SD Context Tag 2 (Property Identifier, L = 1)
        X  '67'              103 (RELIABILTY)
        X  '4E'              PD Opening Tag 4 (Property Value)
                X  '91'          Application Tag 9 (Enumerated, L = 1)
                X  '00'          0 (NO_FAULT_DETECTED)
        X  '4F'              PD Closing Tag 4 (Property Value)

        X  '29'              SD Context Tag 2 (Property Identifier, L = 1)
        X  '51'              81 (OUT_OF_SERVICE)
        X  '4E'              PD Opening Tag 4 (Property Value)
                X  '11'          Application Tag 1 (TRUE)
        X  '4F'              PD Closing Tag 4 (Property Value)
X  '1F'              PD Opening Tag 1 (List Of Results)

X  '0C'              SD Context Tag 0 (Object Identifier, L = 4)
X  '00000002'        Analog Input, Instance Number = 2
X  '1E'              PD Opening Tag 1 (List Of Results)
        X  '29'              SD Context Tag 2 (Property Identifier, L = 1)
        X  '55'              85 (PRESENT_VALUE)
        X  '4E'              PD Opening Tag 4 (Property Value)
                X  '44'          Application Tag 4 (Real, L = 4)
               X  '437A0000'     250.0
        X  '4F'              PD Closing Tag 4 (Property Value)

        X  '29'              SD Context Tag 2 (Property Identifier, L = 1)
        X  '67'              103 (RELIABILITY)
        X  '4E'              PD Opening Tag 4 (Property Value)
                X  '91'          Application Tag 9 (Enumerated, L = 1)
                X  '07'          7 (UNRELIABLE_OTHER)
        X  '4F'              PD Closing Tag 4 (Property Value)

        X  '29'              SD Context Tag 2 (Property Identifier, L = 1)
```

```
        X '51'          81 (OUT_OF_SERVICE)
        X '4E'          PD Opening Tag 4 (Property Value)
            X '10'          Application Tag 1 (Boolean, 0 (FALSE))
        X '4F'          PD Closing Tag 4 (Property Value)
X '1F'          PD Opening Tag 1 (List Of Results)
```

例 4：寻找所有 Object_Names 与比较值“C* Pressure”和“AC? Supply Temp”相符的对象的标识符。

```
X '00'          PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                = 0, SA = 0)
X '04'          Maximum APDU Size Accepted = 1024 octets
X '54'          Invoke ID = 84
X '0D'          Service Choice = 13 (ReadPropertyConditional Request)

X '0E'          PD Opening Tag 0 (Object Selection Criteria)
        X '09'          SD Context Tag 0 (Selection Logic, L = 1)
        X '01'          1 (OR)

        X '1E'          PD Opening Tag 1 (List Of Selection Criteria)
            X '09'          SD Context Tag 0 (Selection Logic, L
= 1)
            X '4D'          77 (OBJECT_NAME)
            X '29'          SD Context Tag 2 (Relation Specifier)
            X '00'          0 (EQUAL)
            X '3E'          PD Opening Tag 3 (Comparison Value)
                X '7D'          Application Tag 7 (Character
String, L>4)
                X '0C'          Extended Length = 12
                X '00'          ANSI X3.4 Encoding
                X '432A205073657373757265'      “C* Pressure”
            X '3F'          PD Closing Tag 3 (Comparison Value)

            X '09'          SD Context Tag 0 (Property Identifier,
```

L = 1)

X ‘4D’ 77 (OBJECT_NAME)

X ‘29’ SD Context Tag 2 (Relation Specifier)

X ‘00’ 0 (EQUAL)

X ‘3E’ PD Opening Tag 3 (Comparison Value)

X ‘7D’ Application Tag 7 (Character String, L>4)

X ‘10’ Extended Length = 16

X ‘00’ ANSI X3.4 Encoding

X ‘41433F20537570706C792054656D70’ “AC? Supply Temp”

X ‘3F’ PD Closing Tag 3 (Comparison Value)

X ‘1F’ PD Closing Tag 1 (List Of Selection Criteria)

X ‘0F’ PD Closing Tag 0 (Object Selection Criteria)

X ‘1E’ PD Opening Tag 1 (List Of Property References)

X ‘09’ SD Context Tag 0 (Property Identifier, L = 1)

X ‘4D’ 77 (OBJECT_NAME)

X ‘1F’ PD Closing Tag 1 (List Of Property References)

假定服务进程执行正确，那么将返回一个复杂确认：

X ‘30’ PDU Type = 3 (BACnet-ComplexACK-PDU, SEG = 0, MOR= 0)

X ‘54’ Invoke ID = 84

X ‘0D’ Service ACK Choice = 13 (ReadPropertyConditional-ACK)

X ‘0C’ SD Context Tag 0 (Object Identifier, L = 4)

X ‘00000004’ Analog Input, Instance Number = 4

X ‘1E’ PD Opening Tag 1 (List Of Results)

X ‘29’ SD Context Tag 2 (Property Identifier, L = 1)

X ‘4D’ 77 (OBJECT_NAME)

X ‘4E’ PD Opening Tag 4 (Property Value)

X ‘75’ Application Tag 7 (Character String, L>4)

X ‘10’ Extended Length = 16

X ‘00’ ANSI X3.4 Encoding

```
                X  '41433120537570706C792054656D70'      "AC1 Supply Temp"
        X  '4F'             PD Closing Tag 4 (Property Value)
X  '1F'             PD Closing Tag 1 (List Of Results)

X  '0C'             SD Context Tag 0 (Object Identifier, L = 4)
X  '00000007'       Analog Input, Instance Number = 7
X  '1E'             PD Opening Tag 1 (List Of Results)
        X  '29'             SD Context Tag 2 (Property Identifier, L = 1)
        X  '4D'             77 (OBJECT_NAME)
        X  '4E'             PD Opening Tag 4 (Property Value)
                X  '75'         Application Tag 7 (Character String, L>4)
                X  '0E'         Extended Length = 14
                X  '00'         ANSI X3.4 Encoding
                X  '435750312050726573737572657'         "CWP1 Pressure"
        X  '4F'             PD Closing Tag 4 (Property Value)
X  '1F'             PD Closing Tag 1 (List Of Results)

X  '0C'             SD Context Tag 0 (Object Identifier, L = 4)
X  '00000008'       Analog Input, Instance Number = 8
X  '1E'             PD Opening Tag 1 (List Of Results)
        X  '29'             SD Context Tag 2 (Property Identifier, L = 1)
        X  '4D'             77 (OBJECT_NAME)
        X  '4E'             PD Opening Tag 4 (Property Value)
                X  '75'         Application Tag 7 (Character String, L>4)
                X  '19'         Extended Length = 25
                X  '00'         ANSI X3.4 Encoding
                X  '43686960C6572203120467265 6F6E20507265737375726572657' "Chiller 1
                                                                    Freon
Pressure"
        X  '4F'             PD Closing Tag 4 (Property Value)
X  '1F'             PD Closing Tag 1 (List Of Results)

X  '0C'             SD Context Tag 0 (Object Identifier, L = 4)
X  '0000000A'       Analog Input, Instance Number = 10
X  '1E'             PD Opening Tag 1 (List Of Results)
```

```
            X  '29'             SD Context Tag 2 (Property Identifier, L = 1)
            X  '4D'             77 (OBJECT_NAME)
            X  '4E'             PD Opening Tag 4 (Property Value)
                    X  '75'         Application Tag 7 (Character String, L>4)
                    X  '10'         Extended Length = 16
                    X  '00'         ANSI X3.4 Encoding
                    X  '41433220537570706C792054656D70'        "AC2 Supply Temp"
            X  '4F'             PD Closing Tag 4 (Property Value)
      X  '1F'             PD Closing Tag 1 (List Of Results)

      X  '0C'             SD Context Tag 0 (Object Identifier, L = 4)
      X  '0000000C'       Analog Input, Instance Number = 12
      X  '1E'             PD Opening Tag 1 (List Of Results)
            X  '29'             SD Context Tag 2 (Property Identifier, L = 1)
            X  '4D'             77 (OBJECT_NAME)
            X  '4E'             PD Opening Tag 4 (Property Value)
                    X  '75'         Application Tag 7 (Character String, L>4)
                    X  '10'         Extended Length = 16
                    X  '00'         ANSI X3.4 Encoding
                    X  '41433220537570706C792054656D70'        "AC3 Supply Temp"
            X  '4F'             PD Closing Tag 4 (Property Value)
      X  '1F'             PD Closing Tag 1 (List Of Results)
```

F.3.7 例 F.3.7 编码——读多个属性服务

例 1：读单一对象中的多个属性。

```
      X  '00'             PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                    = 0, SA = 0)
      X  '04'             Maximum APDU Size Accepted = 1024 octets
      X  'F1'             Invoke ID = 241
      X  '0E'             Service Choice = 14 (ReadPropertyMulitiple-Request)

      X  '0C'             SD Context Tag 0 (Object Identifier, L = 4)
```

```
X  ‘00000010’       Analog Input, Instance Number = 16
X  ‘1E’             PD Opening Tag 1 (List Of Property Reference)
       X  ‘09’             SD Context Tag 0 (Property Identifier, L = 1)
       X  ‘55’             85 (PRESENT_VALUE)

       X  ‘09’             SD Context Tag 0 (Property Identifier, L = 1)
       X  ‘67’             103 (RELIABILITY)
X  ‘1F’             PD Closing Tag 1 (List Of Property Reference)
```

假定服务进程执行正确，那么将返回一个复杂确认：

```
X  ‘30’             PDU Type = 3 (BACnet-ComplexACK-PDU, SEG = 0, MOR= 0)
X  ‘F1’             Invoke ID = 241
X  ‘0E’             Service ACK Choice = 14 (ReadPropertyMultiple-ACK)

X  ‘0C’             SD Context Tag 0 (Object Identifier, L = 4)
X  ‘00000010’       Analog Input, Instance Number = 16
X  ‘1E’             PD Opening Tag 1 (List Of Results)
       X  ‘29’             SD Context Tag 2 (Property Identifier, L = 1)
       X  ‘55’             85 (PRESENT_VALUE)
       X  ‘4E’             PD Opening Tag 4 (Property Value)
              X  ‘44’      Application Tag 4 (Real, L = 4)
              X  ‘4290999’     72.3
       X  ‘4F’             PD Closing Tag 4 (Property Value)

       X  ‘29’             SD Context Tag 2 (Property Identifier, L = 1)
       X  ‘67’             103 (RELIABILITY)
       X  ‘4E’             PD Opening Tag 4 (Property Value)
              X  ‘91’          Application Tag 9 (Enumerated, L = 1)
              X  ‘00’          0 (NO_FAULT_DETECTED)
       X  ‘4F’             PD Closing Tag 4 (Property Value)
X  ‘1F’             PD Closing Tag 1 (List Of Results)
```

例 2：读若干个对象的属性。

```
X ‘00’          PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0,
MOR
                = 0, SA = 0)
X ‘04’          Maximum APDU Size Accepted = 1024 octets
X ‘02’          Invoke ID = 2
X ‘0E’          Service Choice = 14 (ReadPropertyMulitiple-Request)

X ‘0C’          SD Context Tag 0 (Object Identifier, L = 4)
X ‘00000021’    Analog Input, Instance Number = 33
X ‘1E’          PD Opening Tag 1 (List Of Property Reference)
      X ‘09’          SD Context Tag 0 (Property Identifier, L = 1)
      X ‘55’          85 (PRESENT_VALUE)
X ‘1F’          PD Closing Tag 1 (List Of Property Reference)

X ‘0C’          SD Context Tag 0 (Object Identifier, L = 4)
X ‘00000032’    Analog Input, Instance Number = 50
X ‘1E’          PD Opening Tag 1 (List Of Property Reference)
      X ‘09’          SD Context Tag 0 (Property Identifier, L = 1)
      X ‘55’          85 (PRESENT_VALUE)
X ‘1F’          PD Closing Tag 1 (List Of Property Reference)

X ‘0C’          SD Context Tag 0 (Object Identifier, L = 4)
X ‘00000023’    Analog Input, Instance Number = 35
X ‘1E’          PD Opening Tag 1 (List Of Property Reference)
      X ‘09’          SD Context Tag 0 (Property Identifier, L = 1)
      X ‘55’          85 (PRESENT_VALUE)
X ‘1F’          PD Closing Tag 1 (List Of Property Reference)
```

假定服务进程执行正确，那么将返回一个复杂确认：

```
X ‘30’          PDU Type = 3 (BACnet-ComplexACK-PDU, SEG = 0, MOR= 0)
X ‘02’          Invoke ID = 2
X ‘0E’          Service ACK Choice = 14 (ReadPropertyMultiple-ACK)
```

X ‘0C’ SD Context Tag 0 (Object Identifier, L = 4)
X ‘00000021’ Analog Input, Instance Number = 33
X ‘1E’ PD Opening Tag 1 (List Of Results)
X ‘29’ SD Context Tag 2 (Property Identifier, L = 1)
X ‘55’ 85 (PRESENT_VALUE)
X ‘4E’ PD Opening Tag 4 (Property Value)
X ‘44’ Application Tag 4 (Real, L = 4)
X ‘42293333’ 42.3
X ‘4F’ PD Closing Tag 4 (Property Value)
X ‘1F’ PD Closing Tag 1 (List Of Results)

X ‘0C’ SD Context Tag 0 (Object Identifier, L = 4)
X ‘00000032’ Analog Input, Instance Number = 50
X ‘1E’ PD Opening Tag 1 (List Of Results)
X ‘29’ SD Context Tag 2 (Property Identifier, L = 1)
X ‘55’ 85 (PRESENT_VALUE)
X ‘5E’ PD Opening Tag 5 (Property Access Error)
X ‘91’ Application Tag 9 (Enumerated, L = 1) (Error Class)
X ‘01’ 1 (OBJECT)
X ‘91’ Application Tag 9 (Enumerated, L = 1) (Error Class)
X ‘1F’ 31 (UNKNOWN_OBJECT)
X ‘5F’ PD Closing Tag 5 (Property Access Error)
X ‘1F’ PD Closing Tag 1 (List Of Results)

X ‘0C’ SD Context Tag 0 (Object Identifier, L = 4)
X ‘00000023’ Analog Input, Instance Number = 35
X ‘1E’ PD Opening Tag 1 (List Of Results)
X ‘29’ SD Context Tag 2 (Property Identifier, L = 1)
X ‘55’ 85 (PRESENT_VALUE)
X ‘4E’ PD Opening Tag 4 (Property Value)
X ‘44’ Application Tag 4 (Real, L = 4)
X ‘43D9D99A’ 435.7
X ‘4F’ PD Closing Tag 4 (Property Value)

X ‘1F’ PD Closing Tag 1 (List Of Results)

F.3.8 例 F.3.8 编码——写属性服务

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘04’ Maximum APDU Size Accepted = 1024 octets
X ‘59’ Invoke ID = 89
X ‘0F’ Service Choice = 15 (WriteProperty-Request)

X ‘0C’ SD Context Tag 0 (Object Identifier, L = 4)
X ‘00800001’ Analog Input, Instance Number = 1
X ‘19’ SD Context Tag 1 (Property Identifier, L = 1)
X ‘55’ 85 (PRESENT_VALUE)
X ‘3E’ PD Opening Tag 3 (Property Value)
X ‘44’ Application Tag 4 (Real, L = 4)
X ‘43340000’ Property Value = 180.0
X ‘3F’ PD Closing Tag 3 (Property Value)

假定服务进程执行正确，那么将返回一个简单确认：

X ‘20’ PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘59’ Invoke ID = 89
X ‘0F’ Service ACK Choice = 15 (WriteProperty)

F.3.9 例 F.3.9 编码——写多个属性服务

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘04’ Maximum APDU Size Accepted = 1024 octets
X ‘01’ Invoke ID = 1
X ‘10’ Service Choice = 16 (WritePropertyMulitiple-Request)

```
X  ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
X  ‘00800005’        Analog Input, Instance Number = 5
X  ‘1E’              PD Opening Tag 1 (List Of Properties)
        X  ‘09’              SD Context Tag 0 (Property Identifier, L = 1)
        X  ‘55’              85 (PRESENT_VALUE)
        X  ‘1E’              PD Opening Tag 1 (List Of Properties)
                X  ‘44’      Application Tag 4 (Real, L = 4)
                X  ‘42860000’     67.0
        X  ‘2F’              PD Closing Tag 2 (Property Value)
X  ‘1F’              PD Closing Tag 1 (List Of Properties)

X  ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
X  ‘00800006’        Analog Input, Instance Number = 6
X  ‘1E’              PD Opening Tag 1 (List Of Results)
        X  ‘09’              SD Context Tag 0 (Property Identifier, L = 1)
        X  ‘55’              85 (PRESENT_VALUE)
        X  ‘2E’              PD Opening Tag 2 (Property Value)
                X  ‘44’      Application Tag 4 (Real, L = 4)
                X  ‘42860000’     67.0
        X  ‘2F’              PD Closing Tag 2 (Property Value)
X  ‘1F’              PD Closing Tag 1 (List Of Properties)

X  ‘0C’              SD Context Tag 0 (Object Identifier, L = 4)
X  ‘00800007’        Analog Input, Instance Number = 7
X  ‘1E’              PD Opening Tag 1 (List Of Results)
        X  ‘09’              SD Context Tag 0 (Property Identifier, L = 1)
        X  ‘55’              85 (PRESENT_VALUE)
        X  ‘2E’              PD Opening Tag 2 (Property Value)
                X  ‘44’      Application Tag 4 (Real, L = 4)
                X  ‘42900000’     72.0
        X  ‘2F’              PD Closing Tag 2 (Property Value)
X  ‘1F’              PD Closing Tag 1 (List Of Properties)
```

假定服务进程执行正确，那么将返回一个简单确认：

X ‘20’ PDU Type = 2 (BACnet-SimpleACK-PDU)

X ‘01’ Invoke ID = 1

X ‘10’ Service ACK Choice = 16 (WritePropertyMultiple)

F.4 远程设备管理服务编码例程

F.4.1 例 F.4.1 编码——设备通信控制服务

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR = 0, SA = 0)

X ‘04’ Maximum APDU Size Accepted = 1024 octets

X ‘05’ Invoke ID = 5

X ‘11’ Service Choice = 17 (DeviceCommunicationControl-Request)

X ‘09’ SD Context Tag 0 (Time Duration, L = 1)

X ‘05’ 5

X ‘19’ SD Context Tag 1 (Enable-Disable, L = 1)

X ‘01’ 1 (DISABLE)

X ‘2D’ SD Contended Tag 2 (Password, L>4)

X ‘08’ Extended Length = 8

X ‘00’ ANSI X3.4 Encoding

X ‘23656762646621’ “#egbdf!”

假定服务进程执行正确，那么将返回一个简单确认：

X ‘20’ PDU Type = 2 (BACnet-SimpleACK-PDU)

X ‘05’ Invoke ID = 5

X ‘11’ Simple ACK Choice = 17 (DeviceCommunicationControl)

F.4.2 例 F.4.2 编码——有证实专有传输服务

X ‘00’ PDU Type = 0 ((BACnet-Confirmed-Request-PDU) SEG = 0, MOR = 0, SA = 0)

X ‘04’ Maximum APDU Size Accepted = 1024 octets
X ‘55’ Invoke ID = 85
X ‘12’ Service Choice = 18 (ConfirmedPrivateTransfer)

X ‘09’ SD Context Tag 0 (Vendor ID, L = 1)
X ‘19’ 25(XYZ Controls Company Limited)
X ‘19’ SD Context Tag 1 (Service Number, L = 1)
X ‘08’ 8 (XYZ Proprietary Service #8)
X ‘2E’ PD Opening Tag 2 (Service Parameters)
X ‘44’ Application Tag 4 (Real, L = 4)
X ‘4290CCCD’ 72.4
X ‘62’ Application Tag 6 (Octet String, L = 2)
X ‘1649’ X ‘1649’
X ‘2F’ PD Closing Tag 2 (Service Parameters)

假定服务成功执行，但不需要返回请求方 BACnet 用户结果，那么将使用 BACnet 复杂应答 PDU 返回一‘Result(+)’原语。

X ‘30’ PDU Type = 3 (BACnet-ComplexACK-PDU, SEG = 0, MOR = 0)
X ‘55’ Invoke ID = 85
X ‘12’ Service Choice = 18 (ConfirmedPrivateTransfer-ACK)

X ‘09’ SD Context Tag 0 (Vendor ID, L = 1)
X ‘19’ 25
X ‘19’ SD Context Tag 1 (Service Number, L = 1)
X ‘08’ 8

F.4.3 例 F.4.3 编码——无证实专有传输服务

X ‘10’ PDU Type = 1 (BACnet-Unconfirmed-Request-PDU)
X ‘04’ Service Choice = 4 (UnconfirmedPrivateTransfer-Request)

X ‘09’ SD Context Tag 0 (Vendor ID, L = 1)

X ‘19’ 25
X ‘19’ SD Context Tag 1 (Service Number, L = 1)
X ‘08’ 8
X ‘2E’ PD Opening Tag 2 (Service Parameters)
X ‘44’ Application Tag 4 (Real, L = 4)
X ‘4290CCCD’ 72.4
X ‘62’ Application Tag 6 (Octet String, L = 2)
X ‘1649’ X ‘1649’
X ‘2F’ PD Closing Tag 2 (Service Parameters)

F4.4 例 F.4.4 编码——重新初始化设备服务

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘01’ Maximum APDU Size Accepted = 128 octets
X ‘02’ Invoke ID = 2
X ‘14’ Service Choice = 20 (ReinitializeDevice-Request)

X ‘09’ SD Context Tag 0 (Reinitialized State Of Device, L = 1)
X ‘01’ 1(WARMSTART)
X ‘1D’ SD Context Tag 1 (Password, L>4)
X ‘09’ Extended Length = 9
X ‘00’ ANSI X3.4 Encoding
X ‘4162436445664768’ “AbCdEfGh”

假定服务进程执行正确，那么将返回一个简单确认：

X ‘20’ PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘02’ Invoke ID = 2
X ‘14’ Service ACK Choice = 20 (ReinitializeDevice)

F.4.5 例 F.4.5 编码——有证实文本报文服务

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘01’ Maximum APDU Size Accepted = 128 octets
X ‘03’ Invoke ID = 3
X ‘13’ Service Choice = 19 (ConfirmedTextMessage-Request)

X ‘0C’ SD Context Tag 0 (Text Message Source Device, L = 4)
X ‘02000005’ Device, Instance Number = 5
X ‘29’ SD Context Tag 2 (Message Priority, L = 1)
X ‘00’ 0(NORMAL)
X ‘3D’ SD Context Tag 3 (Message, L>4)
X ‘18’ Extended Length = 24
X ‘00’ ANSI X3.4 Encoding
X ‘504D20726571756972656420666F722050554D50333437’ “PM required for PUMP347”

假定服务进程执行正确，那么将返回一个简单确认：

X ‘20’ PDU Type = 2 (BACnetSimpleACK-PDU)
X ‘03’ Invoke ID = 3
X ‘13’ Service Choice = 19 (ConfirmedTextMessage)

F.4.6 例 F.4.6 编码——无证实文本报文服务

X ‘10’ PDU Type = 1 (BACnet-Unconfirmed-Request-PDU)
X ‘05’ Service Choice = 5 (UnconfirmedPrivateTransfer-Request)

X ‘0C’ SD Context Tag 0 (Text Message Source Device, L = 4)
X ‘02000005’ Device, Instance Number = 5
X ‘29’ SD Context Tag 2 (Message Priority, L = 1)
X ‘00’ 0(NORMAL)
X ‘3D’ SD Context Tag 3 (Message, L>4)
X ‘18’ Extended Length = 24

X ‘00’ ANSI X3.4 Encoding

X ‘504D20726571756972656420666F722050554D50333437’ “PM required for PUMP347”

注意此报文不需要响应，因其是无证实型的。

F.4.7 例 F.4.7 编码——时间同步服务

X ‘10’ PDU Type = 1 (BACnet-Unconfirmed-Request-PDU)

X ‘06’ Service Choice = 6 (TimeSynchronization-Request)

X ‘A4’ Application Tag 10 (Date, L = 4)

X ‘5C0B11FF’ November 17, 1992 (Day of Week Unspecified)

X ‘B4’ Application Tag 11 (Time, L = 4)

X ‘162D1E46’ 22:45:30.70

F.4.8 例 F.4.8 编码——Who-Has 和 I-Have 服务

例 1：寻找包含已知对象名的对象的设备。

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)

X ‘07’ Service Choice = 7 (Who-Has-Request)

X ‘3D’ SD Context Tag 3 (Object Name, L>4)

X ‘07’ Extended Length = 7

X ‘00’ ANSI X3.4 Encoding

X ‘4F4154656D70’ “OA Temp”

假定只有一台设备有上述对象，那么将收到下面的 I-Have 服务指示。

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)

X ‘01’ Service Choice = 1 (I-Have-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (Device Identifier)
X ‘02000008’ Device, Instance Number = 8
X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (Device Identifier)
X ‘00000003’ Analog Input, Instance Number = 3
X ‘75’ Application Tag 7 (Character String, L>4) (Object Name)
X ‘07’ Extended Length = 7
X ‘00’ ANSI X3.4 Encoding
X ‘4F4154656D70’ “OA Temp”

例 2：寻找包含已知对象标识符的对象的设备。

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)
X ‘07’ Service Choice = 7 (Who-Has-Request)

X ‘2C’ SD Context Tag 2 (Object Identifier, L = 4)
X ‘00000003’ Analog Input, Instance Number = 3

假定只有一台设备有上述对象，那么将收到下面的 I-Have 服务指示。

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)
X ‘01’ Service Choice = 1 (I-Have-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (Device Identifier)
X ‘02000008’ Device, Instance Number = 8
X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (Device Identifier)
X ‘00000003’ Analog Input, Instance Number = 3
X ‘75’ Application Tag 7 (Character String, L>4) (Object Name)
X ‘07’ Extended Length = 7

X ‘00’ ANSI X3.4 Encoding
X ‘4F4154656D70’ “OA Temp”

F.4.9 例 F.4.9 编码——Who-Is 和 I-Am 服务

例 1：建立一台已知设备对象标识符如实例号的设备的网络地址。

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)
X ‘08’ Service Choice = 8 (Who-Is-Request)

X ‘09’ SD Context Tag 0 (Device Instance Range Low Limit, L = 1)
X ‘03’ 3
X ‘19’ SD Context Tag 1 (Device Instance Range High Limit, L = 1)
X ‘03’ 3

假定在网络上有这样一台设备，那么一会儿后它将使用 I-Am 服务进行响应：

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)
X ‘00’ Service Choice = 0 (I-Am-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (I-Am Device Identifier)
X ‘02000008’ Device, Instance Number = 8
X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (Object Identifier)
X ‘00000003’ Analog Input, Instance Number = 3
X ‘75’ Application Tag 7 (Character String, L>4) (Object Name)
X ‘07’ Extended Length = 7
X ‘00’ ANSI X3.4 Encoding
X ‘4F4154656D70’ “OA Temp”

例 2：找出几乎所有的网络设备。

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)
X ‘08’ Service Choice = 8 (Who-Is-Request)

网络上的每一台设备均使用 I-Am 服务进行响应：

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)
X ‘00’ Service Choice = 0 (I-Am-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (I-Am Device Identifier)
X ‘02000001’ Device, Instance Number = 1
X ‘22’ Application Tag 2 (Unsigned Integer, L = 2) (Max APDU Length Accepted)
X ‘01E0’ 480
X ‘91’ Application Tag 9 (Enumerated, L = 1) (Segmentation Supported)
X ‘01’ 1 (SEGMENTED_TRANSMIT)
X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (vendor ID)
X ‘63’ 99

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)
X ‘00’ Service Choice = 0 (I-Am-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (I-Am Device Identifier)

X ‘02000002’ Device, Instance Number = 2
X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (Max APDU Length Accepted)
X ‘CE’ 206
X ‘91’ Application Tag 9 (Enumerated, L = 1) (Segmentation Supported)
X ‘02’ 2 (SEGMENTED_RECEIVE)
X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (vendor ID)

X ‘21’ 33

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)
X ‘00’ Service Choice = 0 (I-Am-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (I-Am Device Identifier)
X ‘02000003’ Device, Instance Number = 3
X ‘22’ Application Tag 2 (Unsigned Integer, L = 2) (Max APDU Length Accepted)
X ‘0400’ 1024
X ‘91’ Application Tag 9 (Enumerated, L = 1) (Segmentation Supported)
X ‘03’ 3 (NO_SEGMENTATION)
X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (vendor ID)
X ‘63’ 99

X ‘10’ PDU Type = 1 (Unconfirmed-Service-Request-PDU)
X ‘00’ Service Choice = 0 (I-Am-Request)

X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (I-Am Device Identifier)
X ‘02000004’ Device, Instance Number = 4
X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (Max APDU Length Accepted)
X ‘80’ 128
X ‘91’ Application Tag 9 (Enumerated, L = 1) (Segmentation Supported)
X ‘00’ 0 (SEGMENTED_BOTH)
X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (vendor ID)
X ‘42’ 66

F.5 例 F.5 编码——虚拟终端服务

建立虚拟终端会话（VT-Session）：

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘01’ Maximum APDU Size Accepted = 128 octets
X ‘50’ Invoke ID = 80
X ‘15’ Service Choice = 21 (VT-Open-Request)

X ‘91’ Application Tag 9 (Enumerated, L = 1) (VT Class)
X ‘01’ 1 (ANSI_X3.64)
X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (Local VT Session Identifier)
X ‘05’ 5

假定目标设备能够创建新的虚拟终端会话，那么将返回复杂响应：

X ‘30’ PDU Type = 3 (BACnet-ComplexACK-PDU, SEG = 0, MOR = 0
X ‘50’ Invoke ID = 80
X ‘15’ Service Choice = 21 (VT-Open-ACK)

X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (Remote VT Session Identifier)
X ‘1D’ 29

终端开始（Sign-on）。目标设备发送提示：

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘01’ Maximum APDU Size Accepted = 128 octets
X ‘51’ Invoke ID = 81
X ‘17’ Service Choice = 23 (VT-Data-Request)

X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (VT Session Identifier)

X ‘05’ 5

X ‘65’ Application Tag 6 (Octet String, L>1) (VT New Data)

X ‘12’ Extended Length = 18

X ‘0D0A456E746572205573657220456E616D653A’ “{cr}{lf}Enter User Name:”

X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (VT Data Flag)

X ‘00’ 0

假定操作员接口设备正确接受数据，那么将返回一个复杂确认：

X ‘30’ PDU Type = 3 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR= 0)

X ‘51’ Invoke ID = 81

X ‘17’ Service Choice = 23 (VT-Data-ACK)

X ‘09’ SD Context Tag 0 (All New Data Accepted, L = 1)

X ‘01’ 1 (TRUE)

输入用户名：

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR

= 0, SA = 0)

X ‘01’ Maximum APDU Size Accepted = 128 octets

X ‘52’ Invoke ID = 82

X ‘17’ Service Choice = 23 (VT-Data-Request)

X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (VT Session Identifier)

X ‘1D’ 29

X ‘65’ Application Tag 6 (Octet String, L>4) (VT New Data)

X ‘05’ Extended Length = 5

X ‘46524544440D’ “FRED {cr}”

X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (VT Data Flag)

X ‘00’ 0

目标设备响应：

X ‘30’ PDU Type = 3 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR= 0)

X ‘52’ Invoke ID = 82

X ‘17’ Service Choice = 23 (VT-Data-ACK)

X ‘09’ SD Context Tag 0 (All New Data Accepted, L = 1)

X ‘01’ 1 (TRUE)

输入密码：

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR

= 0, SA = 0)

X ‘01’ Maximum APDU Size Accepted = 128 octets

X ‘53’ Invoke ID = 83

X ‘17’ Service Choice = 23 (VT-Data-Request)

X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (VT Session Identifier)

X ‘05’ 5

X ‘65’ Application Tag 6 (Octet String, L>4) (VT New Data)

X ‘15’ Extended Length = 21

X ‘46524544D0A456E746572206061737376F72643A’ “FRED {cr}{lf} Enter Password”

X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (VT Data Flag)

X ‘01’ 1

目标设备响应：

X ‘30’ PDU Type = 3 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR= 0)

X ‘53’ Invoke ID = 83

X ‘17’ Service Choice = 23 (VT-Data-ACK)

X ‘09’ SD Context Tag 0 (All New Data Accepted, L = 1)

X ‘01’ 1 (TRUE)

终端关闭（Sign-off）：

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘01’ Maximum APDU Size Accepted = 128 octets
X ‘54’ Invoke ID = 84
X ‘16’ Service Choice = 22 (VT-Close-Request)

X ‘21’ Application Tag 2 (Unsigned Integer, L = 1) (List Of Remote VT
SessionIdentifier)
X ‘1D’ 29

响应：

X ‘20’ PDU Type = 2 (BACnet-SimpleACK-PDU)
X ‘54’ Invoke ID = 84
X ‘16’ Service ACK Choice = 23 (VT-Close)

F.6 例 F.6 编码——安全服务

设备 A 向密钥服务器发送请求密钥服务请求：

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR
= 0, SA = 0)
X ‘05’ Maximum APDU Size Accepted = 1470 octets
X ‘0F’ Invoke ID = 15
X ‘19’ Service Choice = 25 (RequestKey-Request)

注意 APDU 中从此处到结束的所有字节均加密。

X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (Requesting Device Identifier)

X ‘02000001’ Device, Instance Number = 1

X ‘22’ Application Tag 2 (Unsigned Integer, L = 2) (Network Number)

X ‘0002’ 2

X ‘61’ Application Tag 6 (Octet String, L = 1) (MAC Address)

X ‘11’ 17

X ‘C4’ Application Tag 12 (Object Identifier, L = 4) (Remote Device Identifier)

X ‘02000002’ Device, Instance Number = 2

X ‘22’ Application Tag 2 (Unsigned Integer, L = 2) (Network Number)

X ‘0002’ 2

X ‘61’ Application Tag 6 (Octet String, L = 1) (MAC Address)

X ‘22’ 34

密钥服务器查询验证请求密钥请求：

X ‘00’ PDU Type = 0 (BACnet-Confirmed-Request-PDU, SEG = 0, MOR = 0, SA = 0)

X ‘05’ Maximum APDU Size Accepted = 1470 octets

X ‘01’ Invoke ID = 1

X ‘18’ Service Choice = 24 (Authenticate-Request)

注意 APDU 中从此处到结束的所有字节均加密。

X ‘0C’ SD Context Tag 0 (Pseudo Random Number, L = 4)

X ‘12345678’ 305, 419, 896

X ‘19’ SD Context Tag 1 (Expected Invoke ID, L = 1)

X ‘0F’ 15

请求密钥服务请求的始发者用复杂确认响应验证请求：

X ‘30’ PDU Type = 2 (BACnet-ComplexACK-PDU, SEG = 0, MOR = 0)

X ‘01’ Invoke ID = 01

X ‘18’ Service Choice = 24 (Authenticate-ACK)

注意 APDU 中从此处到结束的所有字节均加密。

X ‘24’ Application Tag 2 (Unsigned Integer, L = 4) (Modified Random Number)

X ‘93B5D7F1’ 2, 478, 168, 049

附件 G — 循环冗余校验（CRC）计算（资料）

（本附件不是标准的一部分，仅仅作为资料使用）

由于历史的原因，CRC 生成器一般采用带有异或反馈的移位寄存器实现。这是一种廉价的用硬件使用串行比特流加工 CRC 信息的方法。因为商用的通用异步收发报机（UART）不提供 CRC 的硬件实现，所以那些在第 9 节和第 10 节描述的基于 UART 的协议必须用软件的方法实现这些运算。这可以模拟带有异或反馈的移位寄存器的工作形式一位一位地计算来实现，也可以用一种更有效的算法将字节作为一个整体计算实现。本附件给出使用这种方式来计算 CRC。下面举出的算法用作例子，并不限制生产商关于 CRC 计算的实现。

G.1 报头 CRC 计算

我们从如图 G-1 所示的硬件 CRC 生成器的框图开始。其中所使用的多项式为：

$$X^8 + X^7 + 1$$

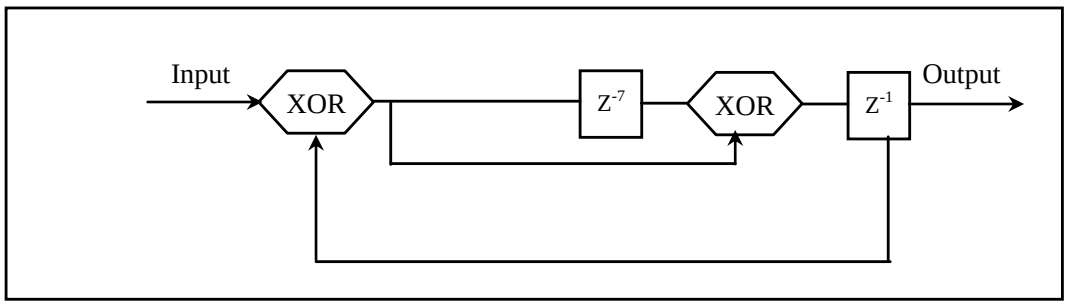


图 G-1. 硬件报头 CRC 生成器

硬件实现是在一个串行比特流上完成的，但我们的计算必须在整个字节上完成。为此，我们参照八位比特数据的电路实现。CRC 移位寄存器的输入端初始化为 X0，输出端初始化为 X7，输入数据从 D0 到 D7（低位先行，如同 UART 的发送和接收）。下面的每一块中，每一位（项）按照纵向异或运算。

```
输入                寄存器内容
    |-------------------------------------------------|
D0  |   X0  X1  X2  X3  X4  X5  X6  X7  |
    |-------------------------------------------------|
D1  |       X0  X1  X2  X3  X4  X5  X6  |
    |   D0                          D0  |
    |   X7                          X7  |
    |-------------------------------------------------|
D2  |           X0  X1  X2  X3  X4  X5  |
```

```
    |   D1  D0                        D1  |
    |   X6  X7                        X6  |
    |   D0                            D0  |
    |   X7                            X7  |
    |-----------------------------------------------|
D3  |               X0  X1  X2  X3    X4  |
    |   D2  D1  D0                    D2  |
    |   X5  X6  X7                    X5  |
    |   D1  D0                        D1  |
    |   X6  X7                        X6  |
    |   D0                            D0  |
    |   X7                            X7  |
    |-----------------------------------------------|
D4  |                   X0  X1  X2    X3  |
    |   D3  D2  D1  D0                    |
    |   X4  X5  X6  X7                X4  |
    |   D2  D1  D0                    D2  |
    |   X5  X6  X7                    X5  |
    |   D1  D0                        D1  |
    |   X6  X7                        X6  |
    |   D0                            D0  |
    |   X7                            X7  |
    |-----------------------------------------------|
D5  |                       X0  X1    X2  |
    |   D4  D3  D2  D1  D0            D4  |
    |   X3  X4  X5  X6  X7            X3  |
    |   D3  D2  D1  D0                D3  |
    |   X4  X5  X6  X7                X4  |
    |   D2  D1  D0                    D2  |
    |   X5  X6  X7                    X5  |
    |   D1  D0                        D1  |
    |   X6  X7                        X6  |
    |   D0                            D0  |
    |   X7                            X7  |
    |-----------------------------------------------|
```

```
D6  |                                 X0  X1  |
    |   D5  D4  D3  D2  D1  D0            D5  |
    |   X2  X3  X4  X5  X6  X7            X2  |
    |   D4  D3  D2  D1  D0                D4  |
    |   X3  X4  X5  X6  X7                X3  |
    |   D3  D2  D1  D0                    D3  |
    |   X4  X5  X6  X7                    X4  |
    |   D2  D1  D0                        D2  |
    |   X5  X6  X7                        X5  |
    |   D1  D0                            D1  |
    |   X6  X7                            X6  |
    |   D0                                D0  |
    |   X7                                X7  |
    |-------------------------------------------------|
D7  |                                     X0  |
    |   D6  D5  D4  D3  D2  D1  D0        D6  |
    |   X1  X2  X3  X4  X5  X6  X7        X1  |
    |   D5  D4  D3  D2  D1  D0            D5  |
    |   X2  X3  X4  X5  X6  X7            X2  |
    |   D4  D3  D2  D1  D0                D4  |
    |   X3  X4  X5  X6  X7                X3  |
    |   D3  D2  D1  D0                    D3  |
    |   X4  X5  X6  X7                    X4  |
    |   D2  D1  D0                        D2  |
    |   X5  X6  X7                        X5  |
    |   D1  D0                            D1  |
    |   X6  X7                            X6  |
    |   D0                                D0  |
    |   X7                                X7  |
    |-------------------------------------------------|
    |                                     D0  |
    |                                     X7  |
    |   D7  D6  D5  D4  D3  D2  D1        D7  |
    |   X0  X1  X2  X3  X4  X5  X6        X0  |
    |   D6  D5  D4  D3  D2  D1  D0        D6  |
```

X1	X2	X3	X4	X5	X6	X7	X1
D5	D4	D3	D2	D1	D0		D5
X2	X3	X4	X5	X6	X7		X2
D4	D3	D2	D1	D0			D4
X3	X4	X5	X6	X7			X3
D3	D2	D1	D0				D3
X4	X5	X6	X7				X4
D2	D1	D0					D2
X5	X6	X7					X5
D1	D0						D1
X6	X7						X6
D0							D0
X7							X7

上述最后一块是一字节数据通过 CRC 生成器移位的结果。所以一个字节可以通过执行指定的异或操作处理成 CRC。为了简化需要的移位操作，我们定义 X0 为并行字节的最高位，X7 为最低位。因为大多数处理器的字节表示与此相反，所以我们重新命名以符合标准处理器的比特命名法。令 C7 = X0，C6 = X1，等等。则上面表单中最后的结果变为：

X0	X1	X2	X3	X4	X5	X6	X7
C7	C6	C5	C4	C3	C2	C1	C0
							D0
							C0
D7	D6	D5	D4	D3	D2	D1	D7
C7	C6	C5	C4	C3	C2	C1	C7
D6	D5	D4	D3	D2	D1	D0	D6
C6	C5	C4	C3	C2	C1	C0	C6
D5	D4	D3	D2	D1	D0		D5
C5	C4	C3	C2	C1	C0		C5
D4	D3	D2	D1	D0			D4
C4	C3	C2	C1	C0			C4
D3	D2	D1	D0				D3
C3	C2	C1	C0				C3

```
|   D2  D1  D0                  D2  |
|   C2  C1  C0                  C2  |
|   D1  D0                      D1  |
|   C1  C0                      C1  |
|   D0                          D0  |
|   C0                          C0  |
|-------------------------------------------------|
```

或者，纵向重新排列使得运算最小化（因为异或是可交换的），并规定任何项与其自身异或应被排除：

```
|-------------------------------------------------|
|   D7  D6  D5  D4  D3  D2  D1  D7  |
|   C7  C6  C5  C4  C3  C2  C1  C7  |
|   D6  D5  D4  D3  D2  D1  D0  D6  |
|   C6  C5  C4  C3  C2  C1  C0  C6  |
|   D5  D4  D3  D2  D1  D0      D5  |
|   C5  C4  C3  C2  C1  C0      C5  |
|   D4  D3  D2  D1  D0          D4  |
|   C4  C3  C2  C1  C0          C4  |
|   D3  D2  D1  D0              D3  |
|   C3  C2  C1  C0              C3  |
|   D2  D1  D0                  D2  |
|   C2  C1  C0                  C2  |
|   D1  D0                      D1  |
|   C1  C0                      C1  |
|   D0                          D0  |
|   C0                          C0  |
|-------------------------------------------------|
```

在运算中，CRC 生成器的 C7-C0 都初始化为 1。在发送过程中，帧类型、目的地址、源地址和长度在发送前都将经过计算。最后的 CRC 寄存器的值的补码将作为结果发送。在接收端，帧类型、目的地址、源地址、长度和接收到的 CRC 字节将经过计算。如果所有字节都正确收到，那么接收方 CRC 寄存器最后的值将是 X’55’。

作为一个应用举例，考虑一个从节点 X’05’到节点 X’10’的令牌帧。如下：

Description	Value	CRC Register After Octet is Processed
Preamble 1,not included in CRC	X‘55’	
Preamble 2,not included in CRC	X‘FF’	

frame type = TOKEN	X ‘00’	X ‘55’
destination address	X ‘10’	X ‘C2’
source address	X ‘05’	X ‘BC’
data length MSB = 0	X ‘00’	X ‘95’
data length LSB = 0	X ‘00’	X ‘73’ ones complement is X ‘8C’
Header CRC	X ‘8C’	X ‘55’ final result at receiver

因此，发送方将对五个字节 X’00’、X’10’、X’05’、X’00’和 X’00’计算 CRC，结果为 X’73’。其补码为 X’8C’，这将追加在帧的后面。

接收方将对六个字节 X’00’、X’10’、X’05’、X’00’、X’00’和 X’8C’计算 CRC，结果为 X’55’，这表示正确接收到帧。

G.1.1 用 C 语言实现报头 CRC 算法举例

本小节提供了一个用 C 语言实现报头 CRC 算法的例子。输入是将被处理的和累加 CRC 的字节。函数返回值就是更新后的 CRC 值。

为了最小化移位运算，我们使用了非循环的比特左移。另外，第 8 位将同最低位字节异或。

期望的结果：

C7 C6 C5 C4 C3 C2 C1 C0	
C6 C5 C4 C3 C2 C1 C0 C7	
C5 C4 C3 C2 C1 C0 C6	
C4 C3 C2 C1 C0 C5	
C3 C2 C1 C0 C4	
C2 C1 C0 C3	
C1 C0 C2	
C0 C1	

移 16 位字：

	C7 C6 C5 C4 C3 C2 C1	v
C7	C6 C5 C4 C3 C2 C1 C0	v<<1
C7 C6	C5 C4 C3 C2 C1 C0	v<<2
C7 C6 C5	C4 C3 C2 C1 C0	v<<3
C7 C6 C5 C4	C3 C2 C1 C0	v<<4

```
       C7 C6 C5 C4 C3   |   C2 C1 C0              |   v<<5
    C7 C6 C5 C4 C3 C2   |   C1 C0                 |   v<<6
 C7 C6 C5 C4 C3 C2 C1   |   C0                    |   v<<7

/*  Accumulate “dataValue” into the CRC in crcValue.
/   Return value is updated CRC
/
/   Assumes that “unsigned char” is equivalent to one octet.
/   Assumes that “unsigned int” is 16 bits.
/   the  ^ operator means exclusive OR.
*/
unsigned char CalcHeaderCRC ( unsigned char dataValue, unsigned char crcValue)
{
    Unsigned int crc;
    crc = crcValue ^ dataValue; /* XOR C7 . . C0 with D7 . . D0 */
    /* Exclusive or the terms in the table (top down) */
    crc = crc ^ (crc << 1) ^ (crc << 2) ^ (crc << 3)
            ^ (crc << 4) ^ (crc << 5) ^ (crc << 6) ^ (crc << 7);
    /* Combine bits shifted out left hand end */
    return (crc & oxfe) ^ ((crc >> 8) & 1);
}
```

G.1.2 用汇编语言实现报头 CRC 算法举例

本小节提供了一个用 68HC11（一种精简指令集的八位处理器）汇编语言实现报头 CRC 算法的例子。程序使用变量 CRCLO 累加 CRC。T1 是一个临时存储变量。大多数指令对累加器 A 或 B 进行操作。

```
HEADERCRC:                  ; ACCUMULATE THE OCTET (0, X) INTO THE DATA CRC
        LDAB 0, X           ; FETCH DATA OCTET (INDEXED OPERATION)
        EORB    CRCLO           ; D7-D0 EXCLUSIVE OR C7-C0
        STAB    CRCLO           ; SAVE RESULT
        CLRA                    ; CLEAR REGISTER A
        ASLD                    ; SHIFT (A, B) LEFT AS A 16 BIT VALUE
        ; A = (-   -   -   -   -   -   -   7)  B = (6  5   4   3   2   1   0
    -)
```

```
    EORB    CRCLO
    ASLD
    ; A = (-    -   -   -   -   -   7   6)  B = (5  4   3   2   1   0   -
-)
    ;     (- -   -   -   -   -   -   7)  B = (6  5   4   3   2   1   0   -)
    EORB    CRCLO
    ASLD
    ; A = (-    -   -   -   -   7   6   5)  B = (4  3   2   1   0   -   -
-)
    ; A = (-    -   -   -   -   -   7   6)  B = (5  4   3   2   1   0   -
-)
    ;     (- -   -   -   -   -   -   7)  B = (6  5   4   3   2   1   0   -)
    EORB    CRCLO
    ASLD
    ; A = (-    -   -   -   7   6   5   4)  B = (3  2   1   0   -   -   -
-)
    ; A = (-    -   -   -   -   7   6   5)  B = (4  3   2   1   0   -   -
-)
    ; A = (-    -   -   -   -   -   7   6)  B = (5  4   3   2   1   0   -
-)
    ;     (- -   -   -   -   -   -   7)  B = (6  5   4   3   2   1   0   -)
    EORB    CRCLO
    ASLD
    ; A = (-    -   -   7   6   5   4   3)  B = (2  1   0   -   -   -   -
-)
    ; A = (-    -   -   -   7   6   5   4)  B = (3  2   1   0   -   -   -
-)
    ; A = (-    -   -   -   -   7   6   5)  B = (4  3   2   1   0   -   -
-)
    ; A = (-    -   -   -   -   -   7   6)  B = (5  4   3   2   1   0   -
-)
    ;     (- -   -   -   -   -   -   7)  B = (6  5   4   3   2   1   0   -)
    EORB    CRCLO
    ASLD
    ; A = (-    -   7   6   5   4   3   2)  B = (1  0   -   -   -   -   -
```

```
-)
    ; A = (-    -   -   7   6   5   4   3)  B = (2  1   0   -   -   -   -
-)
    ; A = (-    -   -   -   7   6   5   4)  B = (3  2   1   0   -   -   -
-)
    ; A = (-    -   -   -   -   7   6   5)  B = (4  3   2   1   0   -   -
-)
    ; A = (-    -   -   -   -   -   7   6)  B = (5  4   3   2   1   0   -
-)
    ;     (- -   -   -   -   -   -   7)  B = (6  5   4   3   2   1   0  -)
    EORB    CRCLO
    ASLD
    ; A = (-    7   6   5   4   3   2   1)  B = (0  -   -   -   -   -   -
-)
    ; A = (-    -   7   6   5   4   3   2)  B = (1  0   -   -   -   -   -
-)
    ; A = (-    -   -   7   6   5   4   3)  B = (2  1   0   -   -   -   -
-)
    ; A = (-    -   -   -   7   6   5   4)  B = (3  2   1   0   -   -   -
-)
    ; A = (-    -   -   -   -   7   6   5)  B = (4  3   2   1   0   -   -
-)
    ; A = (-    -   -   -   -   -   7   6)  B = (5  4   3   2   1   0   -
-)
    ;     (- -   -   -   -   -   -   7)  B = (6  5   4   3   2   1   0  -)
    EORB    CRCLO
    ANDB    #0FEH           ; CLEAR LSB OF BOTTOM HALF
    ANDB    #01H            ; CLEAR ALL BUT LSB OF HIGH HALF
    STAA    T1
    ; A = (-    -   -   -   -   -   -   1)  B = (0  -   -   -   -   -   -
-)
    ;     (- -   -   -   -   -   -   2)  B = (1  0   -   -   -   -   -  -)
    ;     (- -   -   -   -   -   -   3)  B = (2  1   0   -   -   -   -  -)
    ;     (- -   -   -   -   -   -   4)  B = (3  2   1   0   -   -   -  -)
    ;     (- -   -   -   -   -   -   5)  B = (4  3   2   1   0   -   -  -)
```

```
;     (- -   -   -   -   -   -   6)  B = (5  4   3   2   1   0   -   -)
;     (- -   -   -   -   -   -   7)  B = (6  5   4   3   2   1   0   -)
;     (- -   -   -   -   -   -   -)  B = (7  6   5   4   3   2   1   -)
EORB    T1              ; COMBINE LSB OF HIGH HALF
STAB    CRCLO
RTS
```

G.1.3 报头 CRC 算法的其它实现

报头 CRC 算法也可以有其它的实现方法。可以看出，新的 CRC 值是先前的 CRC 值和数据值的异或函数。因此，我们可以利用先前的 CRC 与数据的异或值作为索引，使用一个有 256 项的查找表单来快速求出新的CRC 值。表单的内容可以用 G.1.1 或 G.1.2 中介绍的方法计算。

G.2 数据 CRC 的计算

我们从如图 G-2 所示的硬件 CRC 生成器的框图开始。其中所使用的多项式为 CRC-CCITT：

$$X^{16}+X^{12}+X^{5}+1$$

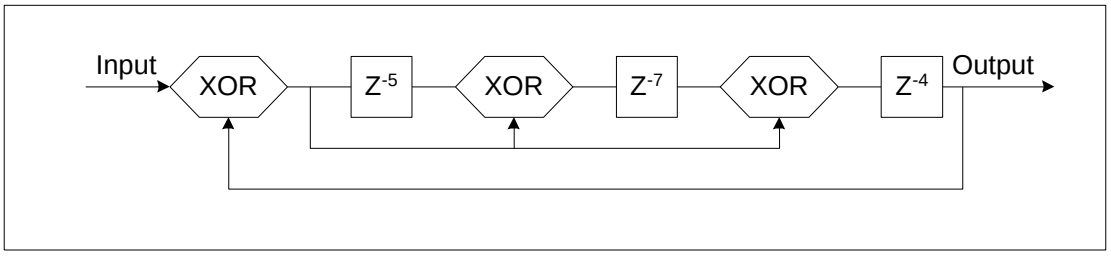


图 G-2 硬件数据 CRC 生成器

硬件实现是在一串行比特流上完成的，但我们的计算必须在整个字节上完成。为此，我们参照八位比特数据的电路实现。CRC 移位寄存器的输入端初始化为 X0，输出端初始化为 X15，输入数据从 D0 到 D7（低位先行，如同 UART 的发送和接收）。下面的每一块中，项纵向异或。

输入　　寄存器内容

```
    |-------------------------------------------------------------------------
-----------------------------|
D0  |   X0  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 X14 X15 |
    |-------------------------------------------------------------------------
-----------------------------|
D1  |       X0  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 X14 |
    |   D0                  D0                          D0              |
    |   X15                 X15                         X15             |
    |-------------------------------------------------------------------------
-----------------------------|
D2  |           X0  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 |
    |   D1  D0              D1  D0                      D1  D0          |
    |   X14 X15             X14 X15                     X14 X15         |
    |-------------------------------------------------------------------------
-----------------------------|
D3  |               X0  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 |
    |   D2  D1  D0          D2  D1  D0                  D2  D1  D0      |
    |   X13 X14 X15         X13 X14 X15                 X13 X14 X15     |
    |-------------------------------------------------------------------------
-----------------------------|
D4  |                   X0  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 |
    |   D3  D2  D1  D0      D3  D2  D1  D0              D3  D2  D1  D0  |
    |   X12 X13 X14 X15     X12 X13 X14 X15             X12 X13 X14 X15 |
    |-------------------------------------------------------------------------
-----------------------------|
D5  |                       X0  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 |
    |       D3  D2  D1  D0  D3  D2  D1  D0                  D3  D2  D1  |
    |       X12 X13 X14 X15     X12 X13 X14 X15             X12 X13 X14 |
    |   D4                  D4                  D4                      |
    |   X11                 X11                 X11                     |
    |   D0                  D0                  D0                      |
    |   X15                 X15                 X15                     |
    |-------------------------------------------------------------------------
-----------------------------|
D6  |                           X0  X1  X2  X3  X4  X5  X6  X7  X8  X9  |
```

```
    |           D3  D2  D1  D0      D3  D2  D1  D0              D3  D2  |
    |           X12 X13 X14 X15     X12 X13 X14 X15             X12 X13 |
    |   D5  D4              D5  D4                      D5  D4          |
    |   X10 X11             X10 X11                     X10 X11         |
    |   D1  D0              D1  D0                      D1  D0          |
    |   X14 X15             X14 X15                     X14 X15         |
    |-------------------------------------------------------------------------
----------------------------|
D7  |                           X0  X1  X2  X3  X4  X5  X6  X7  X8  |
    |               D3  D2  D1  D0      D3  D2  D1  D0              D3  |
    |               X12 X13 X14 X15     X12 X13 X14 X15             X12 |
    |   D6  D5  D4          D6  D5  D4                  D6  D5  D4      |
    |   X9  X10 X11         X9  X10 X11                 X9  X10 X11     |
    |   D2  D1  D0          D2  D1  D0                  D2  D1  D0      |
    |   X13 X14 X15         X13 X14 X15                 X13 X14 X15     |
    |-------------------------------------------------------------------------
----------------------------|
    |                               X0  X1  X2  X3  X4  X5  X6  X7  |
    |                   D3  D2  D1  D0      D3  D2  D1  D0              |
    |                   X12 X13 X14 X15     X12 X13 X14 X15            |
    |   D7  D6  D5  D4      D7  D6  D5  D4              D7  D6  D5  D4  |
    |   X8  X9  X10 X11     X8  X9  X10 X11             X8  X9  X10 X11 |
    |   D3  D2  D1  D0      D3  D2  D1  D0              D3  D2  D1  D0  |
    |   X12 X13 X14 X15     X12 X13 X14 X15             X12 X13 X14 X15 |
    |-------------------------------------------------------------------------
----------------------------|
```

上述最后一块是一字节数据通过 CRC 生成器移位的结果。所以一个字节可以通过执行指定的异或操作处理成 CRC。为了简化需要的移位操作，我们定义 X0 为并行字节的最高位，X15 为最低位。因为大多数处理器的字节表示与此相反，所以我们重新命名以符合标准处理器的比特命名法。令 C15=X0，C14=X1，等等。则上面表单中最后的结果变为：

```
    |-------------------------------------------------------------------------
----------------------------|
    |   X0  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 X14 X15 |
    |-----------------------------------------------+---------------------
```

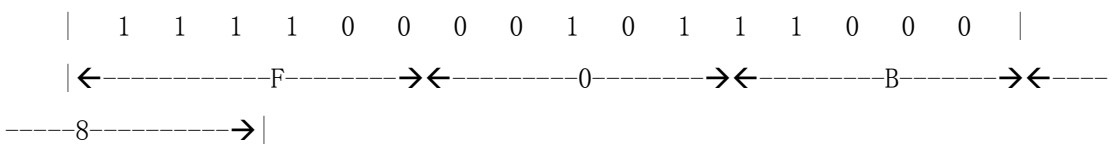
```
--------------------------|
    |   C15 C14 C13 C12 C11 C10 C9  C8  C7  C6  C5  C4  C3  C2  C1  C0  |
    |-------------------------------------------------+-------------------
--------------------------|
    |                                 C15 C14 C13 C12 C11 C10 C9  C8  |
    |                 D3  D2  D1  D0      D3  D2  D1  D0              |
    |                 C3  C2  C1  C0      C3  C2  C1  C0              |
    |   D7  D6  D5  D4      D7  D6  D5  D4              D7  D6  D5  D4  |
    |   C7  C6  C5  C4      C7  C6  C5  C4              C7  C6  C5  C4  |
    |   D3  D2  D1  D0      D3  D2  D1  D0              D3  D2  D1  D0  |
    |   C3  C2  C1  C0      C3  C2  C1  C0              C3  C2  C1  C0  |
    |-------------------------------------------------+-------------------
--------------------------|
```

或者，纵向重新排列以最小化运算（因为异或是可交换的）：

```
    |-------------------------------------------------+-------------------
--------------------------|
    |                                 C15 C14 C13 C12 C11 C10 C9  C8  |
    |   D7  D6  D5  D4  D3  D2  D1  D0                                |
    |   C7  C6  C5  C4  C3  C2  C1  C0                                |
    |                   D7  D6  D5  D4  D3  D2  D1  D0                 |
    |                   C7  C6  C5  C4  C3  C2  C1  C0                 |
    |   D3  D2  D1  D0                              D7  D6  D5  D4    |
    |   C3  C2  C1  C0                              C7  C6  C5  C4    |
    |                                               D3  D2  D1  D0    |
    |                                               C3  C2  C1  C0    |
    |                   D3  D2  D1  D0                                |
    |                   C3  C2  C1  C0                                |
    |-------------------------------------------------+-------------------
--------------------------|
```

在运算中，CRC 生成器的 C15-C0 都初始化为 1。在发送过程中，数据在发送前都将经过计算。最后的 CRC 寄存器的值的补码将作为结果以低位字节（C7-C0）优先的顺序发送。在接收端，所有的数据包括接收到的 CRC 字节将经过计算。如果所有字节都正确收到，那么接

收方 CRC 寄存器最后的值将是 X’F0B8’。



作为一个应用举例，考虑一串数据 X’01’、X’22’和 X’30’。

Description	Value	CRC Register After Octet is Processed
		X‘FFFF’ (initial value)
frame data octet	X‘01’	X‘1E0E’
second data octet	X‘22’	X‘EB70’
third data octet	X‘30’	X‘42EF’ ones complement is X‘BD10’
CRC1 (least significant octet)	X‘10’	X‘0F3A’
CRC2 (most significant octet)	X‘BD’	X‘F0B8’ final result at receiver

因此，发送方将对三个字节 X’01’、X’22’和 X’30’计算 CRC，结果为 X’42EF’。其补码为 X’BD10’，这将按照低位先行即 X’10’、X’BD’的顺序追加在帧的后面。

接收方将对五个字节 X’01’、X’22’、X’30’、X’10’和 X’BD’计算 CRC，结果为 X’F0B8’，这表示正确接收到帧。

G.2.1 用 C 语言实现数据 CRC 算法举例

本小节提供了一个用 C 语言实现数据 CRC 算法的例子。输入是将被处理的和累加 CRC 的字节。函数返回值就是更新后的 CRC 值。

```
/*  Accumulate “dataValue” into the CRC in crcValue.
/   Return value is updated CRC
/
/   Assumes that “unsigned char” is equivalent to one octet.
/   Assumes that “unsigned int” is 16 bits.
/   the ^ operator means exclusive OR.
*/
```

```
unsigned char CalcDataCRC ( unsigned char dataValue, unsigned char crcValue)
{
    Unsigned int crcLow;
    crcLow = (crcValue & 0xff) ^ dataValue; /* XOR C7 . . C0 with D7 . . D0 */
    /* Exclusive or the terms in the table (top down) */
    return (crcValue >> 8) ^ (crcLow << 8) ^ (crcLow << 3)
                    ^ (crcLow << 12) ^ (crcLow << 4)
                    ^ (crcValue & 0xff) ^ ((crcValue & 0xff) << 7);
}
```

G.2.2 用汇编语言实现数据 CRC 算法举例

本小节提供了一个用 68HC11（一种精简指令集的八位处理器）汇编语言实现数据 CRC 算法的例子。程序使用变量 CRCLO 和 CRCHI 累加 CRC。T1 和 T2 是临时存储变量。大多数指令对累加器 A 或 B 进行操作。

```
DATACRC:                        ; ACCUMULATE THE OCTET (0, X) INTO THE DATA CRC
        LDAA 0, X               ; FETCH DATA OCTET (INDEXED OPERATION)
        EORA    CRCLO           ; D7-D0 EXCLUSIVE OR C7-C0
        STAA    CRCLO           ; SAVE RESULT
        CLRB                    ; CLEAR REGISTER B
        LSRA                    ; SHIFT A RIGHT, 0 INTO MSB, LSB INTO CARRY
        RORB                    ; ROTATE B RIGHT, CARRY INTO MSB
        LSRA
        RORB
        LSRA
        RORB
        LSRA
        RORB
        ; A = (-    -   -   -   7   6   5   4)  B = (3  2   1   0   -   -   -
    -)
        ;
        EORB    CRCLO
        ANDA    #0FH            ; MASK OFF ALL BUT LS 4 BITS
        ; A = (-    -   -   -   7   6   5   4)  B = (3  2   1   0   -   -   -
    -)
```

```
;     (- -   -   -   3   2   1   0)
;
STAA    T1              ; SAVE TEMP RESULT (NOT USED EXCEPT AS TEMP)
STABT2
LSRA                ; SHIFT RIGHT 1 BIT
RORB
; A = (-    -   -   -   -   7   6   5)  B = (4  3   2   1   0   -   -
-)
;     (- -   -   -   -   3   2   1)      (0  -   -   -   -   -   -   -)
;
EORA    CRCLO           ; COMBINE PARTAL TERMS
EORA    T2
EORB    CRCHI           ; C8 - C15
EORB    T1
; A = (-    -   -   -   -   7   6   5)  B = (4  3   2   1   0   -   -
-)
;     (- -   -   -   -   3   2   1)      (0  -   -   -   -   -   -   -)
;     (7 6   5   4   3   2   1   0)      (15 14  13  12  11  10  9   8)
;     (3 2   1   0   -   -   -   -)      (-  -   -   -   7   6   5   4)
;                                        (-  -   -   -   3   2   1   0)
;
STAA    CRCHI           ; SAVE RESULT
STAB    CRCLO
RTS
```

G.2.3 数据 CRC 算法的其它实现

数据 CRC 算法也可以有其它的实现方法。可以看出，新的 CRC 值是由两项异或得出的。一项是先前 CRC 值的高位字节。另一项是先前 CRC 值的低位字节和数据值的异或函数。因此，我们可以利用先前 CRC 值的低位字节与数据的异或值作为索引，使用一个有 256 项的查找表单来快速求出第二项的值。然后与先前 CRC 值的高位字节作异或运算以求出新的 CRC 值。表单的内容可以用 G.2.1 或 G.2.2 中介绍的方法计算。

附件 H — 组合 BACnet 网络与 NON-BACnet 网络（规范）

（本附件是标准的一部分，要求实现。）

H.1 将 Non-BACnet 网络映射给 BACnet 路由器

BACnet 路由器除了可以提供互联多个 BACnet 网络的方法之外，也可以用来为 Non-BACnet 网络提供网关功能。Non-BACnet 网络在报文结构，处理过程和介质访问控制技术的使用方面与本标准包含的内容是不同的。使用扩展路由表概念实现将 Non-BACnet 网络映射给 BACnet 路由器，其方法是使用 BACnet NPCI 寻址 Non-BACnet 设备。为此，每个 Non-BACnet 网络分配一个唯一的双字节网络编号，该 Non-BACnet 网络上的每台设备由一个 MAC 地址代表，而这个 MAC 地址可能与该设备在外部网络的实际 MAC 地址字节一致，也可能不一致。因为从定义上说，与 Non-BACnet 网络上的设备通信并没有标准化，所以这样的路由器网关如何进行从 BACnet 端口到 Non-BACnet 端口的报文的解释、翻译以及传送操作的过程超出了本标准的范围。

H.2 在单各个物理设备中的多个“虚拟”BACnet 设备

一个 BACnet 设备定义为拥有一个 BACnet **设备**对象、使用本标准规范的过程进行通信的设备。然而，在有些实例中，有必要用多个 BACnet 设备来模拟一个物理的楼宇自动控制设备的动作。这种设备被称为“虚拟 BACnet 设备”。实现的方法是将这个物理设备配置成为一个或者多个“虚拟网络”的路由器。也就是说，每个虚拟 BACnet 设备与一个唯一的目标网络（DNET）和目标地址（NADR）对相关联，这也是唯一的 BACnet 地址。实际的物理设备所执行的功能与一个路由器的功能相同，这个路由器的作用是在下列两个网络之间进行路由，一个网络是虚拟 BACnet 设备之间传送报文的网络，另一个网络是具有与分配给这些虚拟 BACnet 设备的网络编号相同网络编号的实际 BACnet 网络。

H.3 BACnet 与 IP 路由

本节规范 BACnet 报文使用由美国国防部（DoD）高级计划研究局（DARPA）研制的协议进行传输的过程。这些协议统称为因特网协议族。本附件规范的方法包括 BACnet LSDU 的封装与解封装，以及将这些报文通过 IP 路由器进行传输的技术，这种技术称之为“隧道技术”。虽然本附件规范的技术称为 BACnet/Internet 协议分组组装/分解器，但是关键功能可以直接内建到 BACnet 节点设备之中。

H.3.1 BACnet/Internet 协议分组组装/分解器（B/IP PAD）

B/IP PAD 是一种设备，其中实现了本标准第 6 节规范的 BACnet 网络层功能和因特网的 UDP 协议功能及 IP 协议功能。

当 B/IP PAD 从本地网络接收到一个 BACnet 分组时，检查报文的 NPCI 域，看是否含有

一个 DNET 网络编号。如果有一个网络编号，B/IP PAD 查询一个内部表单，这个表单的表目是网络编号与在 BACnet 互联网上的所有的 B/IP PAD 的 IP 地址、以及在本地网上表示 IP 数据报下一跳的 IP 路由器的 IP 地址的映射。如果搜寻到一个合适的表目，则 B/IP PAD 把 BACnet 报文的 LSDU 部分封装进 UDP 分组中，这里，LSDU 成为 UDP 分组的数据部分，参见图 H-1。UDP 的源端口和目标端口号都设置为 X’BAC0’。然后 B/IP PAD 将这个 UDP 封装进入一个 IP 数据报并且发送给本地的 IP 路由器。这个 IP 数据报的源地址是 B/IP PAD 自己的 IP 地址，目标地址是包含的 BACnet 报文的 DNET 对应的那个 B/IP PAD 的 IP 地址。如果包含的 BACnet 报文中的 DNET 是一个全局广播地址,那么 B/IP PAD 将向表单中的所有 B/IP PAD 传输这个 BACnet LSPU。B/IP PADs 之间的分组的传输遵循 IP 协议标准的规范。

当 B/IP PAD 从一个 IP 路由器接收到一个 IP 数据报时，查找关于 UDP 端口 X’BAC0’的进程，这个进程使用 6.3 节所定义的过程把 UDP 数据报的数据部分转换为 BACnet 报文，然后在本地网络上传输。

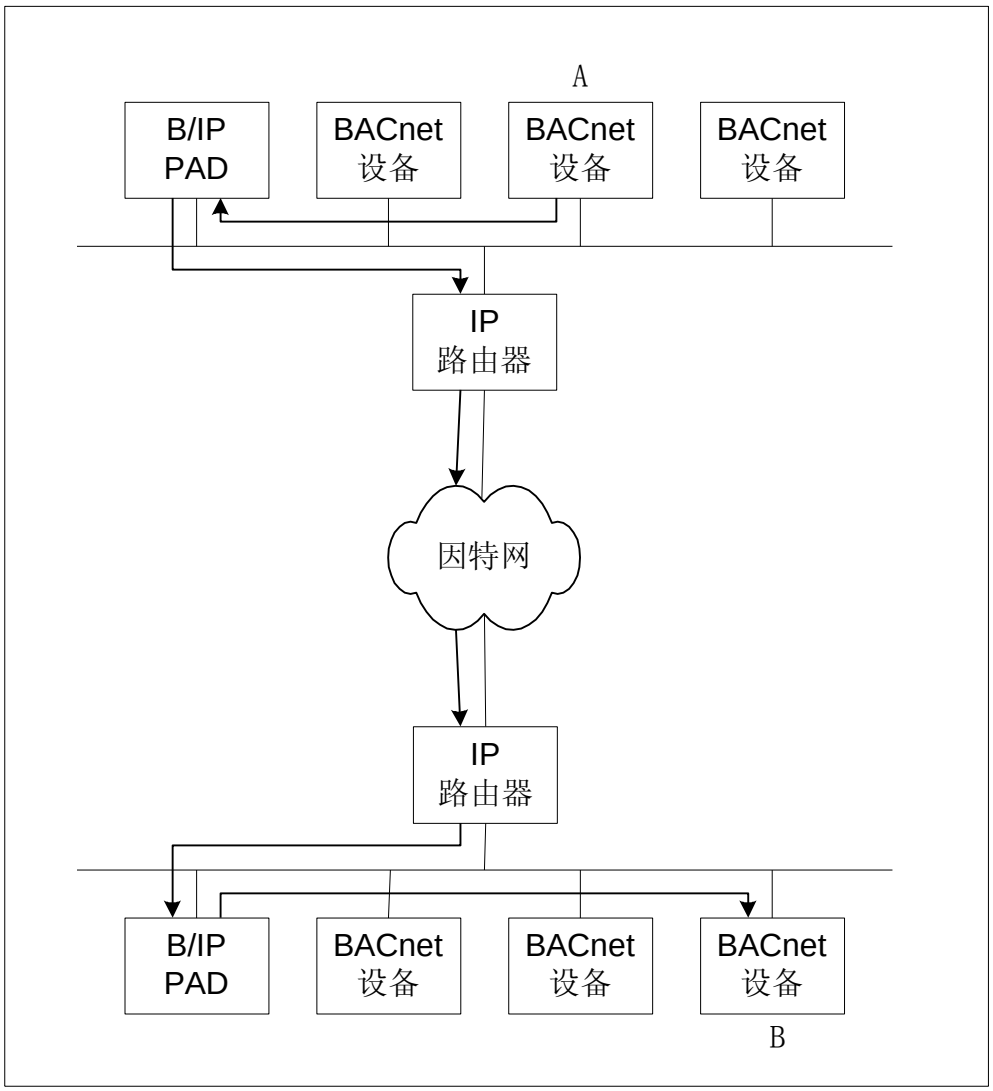
H.3.2 实现注解

图 H-1 表示在 8802-3 局域网上的实现配置。使用包含在 LLC 头部的 LSAP 可以将 BACnet 分组与 IP 分组及其它协议分组区分开。IP 分组使用的 LSAP 标号为 X’06’，而 BACnet 网络层分组使用的 LSAP 标号为 X’82’。在图中所表示的实例中，每个分组在网络上实际出现了两次，一次是作为 BACnet 报文，另一次是作为 IP 报文。B/IP PAD 也构造成为同时完成 BACnet 组装/分解功能和 IP 分组的 IP 路由功能。这样的设备就是 B/IP PAD/Router。参见图 H-2。

关于 DoD 协议的进一步内容，请查阅 25 节列出的 DDN Protocol Handbook。

H.4 BACnet 与 IPX 路由

本节规范 BACnet 报文使用由 Novell 公司研制的互联网数据报协议，即 IPX 协议，传输的过程。这个协议层是基于施乐网络系统（XNS）协议中使用分组交换协议（PEP）的一个子集。本附件规范的方法包括 BACnet LSDU 的封装和解封装，以及它们通过使用 IPX 路由器的功能在 IPX 互联网络上的传输，这种技术称为“IPX 隧道技术”。虽然本附件规范的技术称为 BACnet/Internet 协议分组组装/分解器,但是关键功能可以直接内建到 BACnet 节点设备之中。



图H-1. 使用一个B/IP PAD实现IP隧道

H.4.1 BACnet/IPX 协议分组组装/分解器（B/IPX PAD）

B/IPX PAD 是一种设备，其中实现了本标准第 6 节规范的 BACnet 网络层功能和 IPX 网络层功能。

当 B/IPX PAD 从本地网络接收到一个 BACnet 包时，检查报文的 NPCI 域，看是否含有一个 DNET 网络编号。如果有一个网络编号，B/IPX PAD 查询一个内部表单，这个表单的表目是网络编号与 IPX 网络编号、MAC 地址的映射。IPX 网络编号与 MAC 地址一起表示所有的在 BACnet 互联网中的远程 B/IPX PAD。如果搜寻到一个合适的表目，则 B/IPX PAD 把 BACnet 报文的 LSDU 部分封装进一个 IPX 隧道分组中（参见图 H-3）。这里，IPX 隧道分组的目标网络编号和 MAC 层地址是对应的 B/IPX PAD 的目标网络编号和 MAC 层地址，目标套接字编号设置成为 BACnet IPX 套接字编号 X’87C1’。 然后，B/IPX PAD 把这个隧道分组封装进入一个 IPX 数据报，并发送给本地的 IPX 路由器。这个 IPX 数据报具有发送方 B/IPX PAD 自己的 IPX 网络编号、MAC 层地址和 BACnet IPX 套接字编号作为源地址。如果 BACnet 报文中的 DNET

为全局广播地址，那么 B/IPX PAD 向表单中的所有 B/IPX PAD 传输这个 BACnet LSPU。B/IPX PAD 之间分组的传输遵循 IPX 路由标准规范的过程。

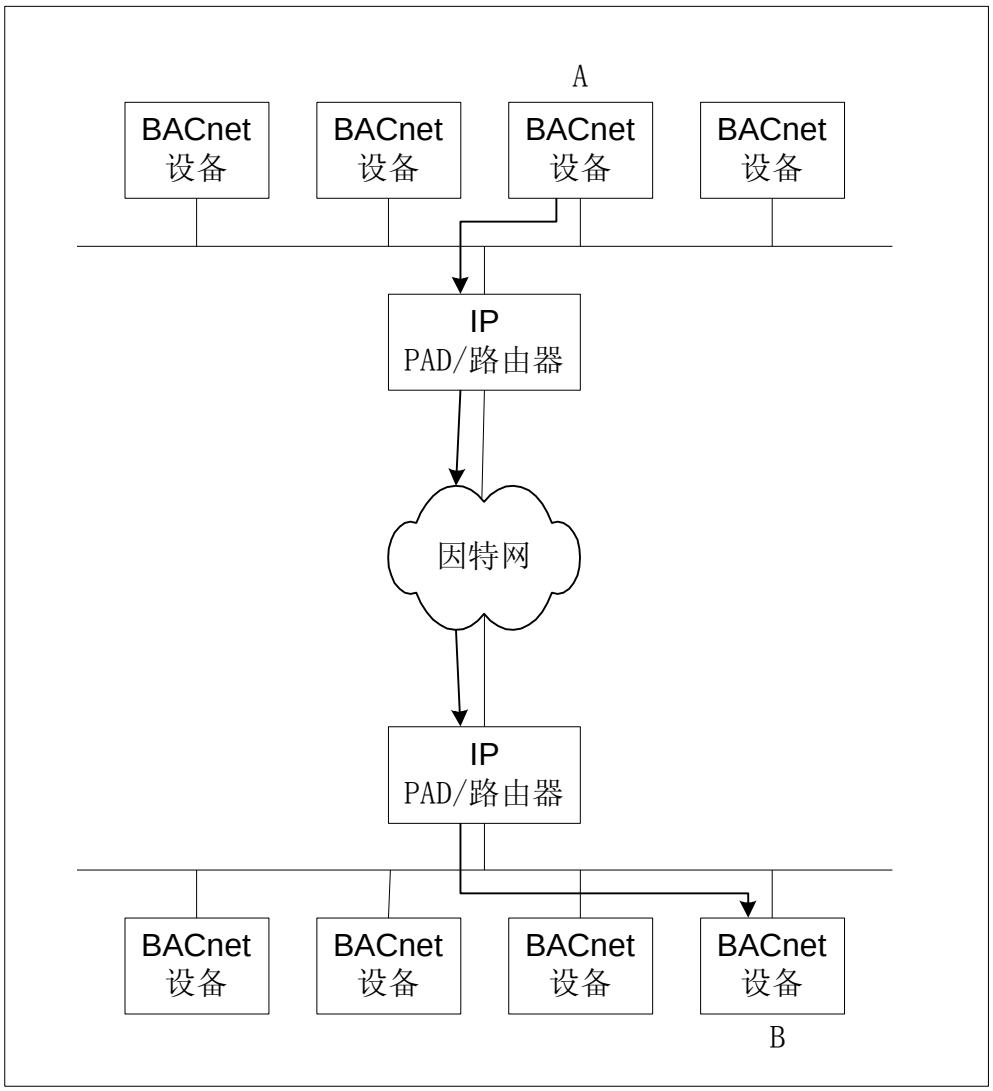
当 B/IPX PAD 使用 BACnet 套接字接收到一个 IPX 数据报时，利用 6.3 节中所定义的过程把 BACnet IPX 隧道分组的数据部分转换为 BACnet 报文在本地网络上传输。

H.4.2 实现注解

图 H-1 表示在 8802-3 局域网上的实现配置。使用包含在 LLC 头部的 LSAP 可以将 BACnet 分组与 IPX 分组及其它协议分组区分开。IPX 分组使用的 LSAP 标号为 X’E0’，而 BACnet 网络层分组使用的 LSAP 标号为 X’82’。在图中所表示的实例中，每个分组在网络上实际出现了两次，一次是作为 BACnet 报文，另一次是作为 IPX 报文。B/IPX PAD 也构造成为同时完成 BACnet 组装/分解功能和 IP 分组的 IPX 路由功能。这样的设备就是 B/IPX PAD/Router。参见图 H-2。

参考文献

NetWare® System Interface Technical Overview
Addison-Wesley Publishing Company, Reading Massachusetts
ISDN 0-201-57027-0



图H-2. 使用一个B/IP PAD路由器实现IP隧道

```
;IPX Header (30 octets)
        dw          X'FFFF'             ;XNS checksum (not used by IPX)
        dw          ?                   ;packet length (set by IPX)
        db          ?                   ;routers crossed (set by IPX)
        db          4                   ;type 4=Packet Exchange Packet
        dd          ?                   ;32 bit dest network number
        db          ?,?,?,?,?,?         ;dest MAC layer address
        dw          X'87C1'             ;dest BACnet socket
        dd          ?                   ;32 bit source network number
        db          ?,?,?,?,?,?         ;source MAC layer address
        dw          X'87C1'             ;source BACnet socket
        db 564 dup ?                    ;BACnet LSDU
```

图H-3. 一个BACnet IPX 隧道分组的结构

附件 I — 具有活动最小值和非活动最小值的可命令属性

（本附件不是标准的一部分，仅仅作为资料使用）

本附件是具有活动最小值和非活动最小值的可命令属性的应用举例。

假设有一个二进制输出对象，其**活动最小值**属性为 600 秒（10 分钟），**非活动最小值**属性为 1200 秒（20 分钟），**极性**属性为**正向**。对象的**优先值数组**属性均为空，由**释放缺省**属性决定的**当前值**属性为**非活动**（off），此状态已经几个小时了。参见图 I-1（a）。

在图 I-1（b）中，优先级为 9 的**当前值**设定为**活动**（on）。请求服务器的内部逻辑将把其写入**优先值数组**中的表目 9。因为**当前值**为 off 已经超过了 20 分钟，所以状态将立即改变。因此，**当前值**的值将为**活动**，而**状态改变时间**属性也将设置为相应的控制时间。

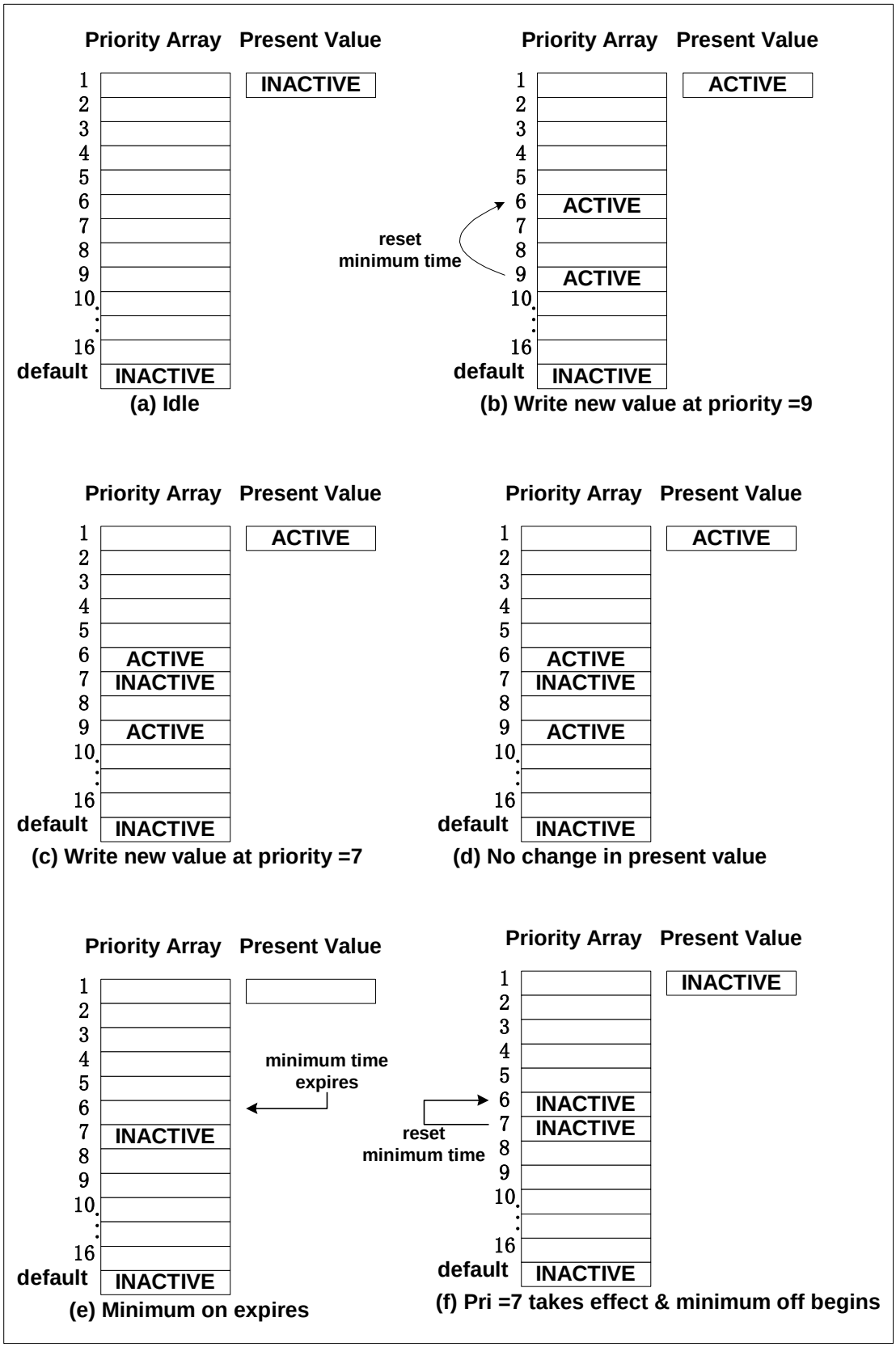
因为**当前值**的状态已经改变，所以最小活动和非活动时间维持实体将把新状态**活动**写入**优先值数组**中的表目 6，以执行最小时间。随后的进入**优先值数组**的优先级低于 6 的**当前值**将不会起作用，这是因为目前状态**活动**的优先级为 6。如图 I-1（c）所示，优先级为 7 的**当前值**的**非活动**值将进入**优先值数组**，但不会起作用。

优先级高于 6 的**当前值**将进入**优先值数组**，并可能引起状态的改变，这是因为其优先级更高的缘故。这是为紧急事件和火灾控制考虑的。

在图 I-1（e）中，当最小活动时间耗尽时即 10 分钟后，设备中的最小活动和最小非活动时间维持实体将发出优先级 6 的释放（NULL）。然后将检查**优先值数组**中剩下的表目以决定新的**当前值**。如果没有状态的变化，那么就不会有进一步的动作发生。在这个例子中，如果没有优先级高于 9 的**非活动**请求，就是这种情况。

在这个例子中，有一个优先级为 7 的**非活动**状态。因此，如图 I-1（f），当最小活动时间耗尽时，优先级为 6 的**活动**释放，优先级为 7 的值将接管控制。**当前值**改变状态进入**非活动**。象以前一样，因为当前值的状态发生了改变，所以最小活动和最小非活动时间维持实体将把新状态**非活动**写入**优先值数组**中的表目 6，以执行最小时间。

注意优先级高于 6 的状态的写入将不受最小活动和最小非活动时间的限制，如果这种写入引起了状态的改变，那么服务器仍将把新值写入**优先值数组**中的表目 6。因此，如果在最小时间内高优先级的请求被释放了，那么在最小时间耗尽前，任何低优先级的请求将不会引起状态的改变。



图I-1. 活动最小值和非活动最小值举例

ASHRAE 关注其成员的开发工作对室内室外环境的影响。ASHRAE 的成员将尽力尽责地努力降低开发的系统和部件对室内和室外环境可能造成的任何有害影响，同时将系统提供的对环境的有益影响最大地发挥出来。

ASHRAE 的短期目标是保证标准范围内的系统和部件对于室内室外环境的影响不超过由它和其它负责机构建立的标准所规定的程度。

ASHRAE 的进一步的目标是继续通过其标准委员会和下属的技术委员会组织制定出更新的恰当的标准，采纳、推荐和促进那些由其它负责机构开发的新的和修正的标准。

在本标准中，相应的章节将会包含更新的标准和设计考虑，这些将作为系统的修订版本发布。

ASHRAE 将引导进行环境信息的研究和宣传，也将从其它相关的负责组织中挑选和传播相关的信息，作为更新标准的指南。

对设备和系统进行设计和选择的影响将被考虑到系统的正面和负面功效的范围。有害材料的处理也将被考虑进来。

ASHRAE 主要关注设备在 ASHRAE 要求的运行空间范围内对环境的影响。然而，能源的选择和由能源与能量传输而可能带来的环境影响也将在必要的情况下加以考虑。关于能源选择的推荐将由其成员进行。