ICS 35.240.20
L 79



# 中华人民共和国国家标准

GB/T 16284.3—1996
idt ISO/IEC 10021-3:1990

# 信息技术　文本通信
# 面向信报的文本交换系统
# 第3部分：抽象服务定义约定

**Information technology—Text communication —**
**Message-Oriented Text Interchange System**
**(MOTIS)—Part 3: Abstrach service definition conventions**



1996-04-10发布　　1996-12-01实施

国家技术监督局　发布

# 前　言

本标准等同采用国际标准 ISO/IEC 10021-3:1990《信息技术——文本通信——面向信报的文本交换系统——第 3 部分:抽象服务定义约定》及其 ISO/IEC 10021-3:1990/Cor.1:1992《技术修改 1》。

本标准正文和附录中引用其他标准时,用我国的标准编号代替相应的国际标准编号,其对应关系是:

GB/T 16262—1996 代替 ISO/IEC 8824:1990;

GB/T 16263—1996 代替 ISO/IEC 8825:1990。

根据编写国家标准的实际情况,不采用 ISO/IEC 10021-3 的附录 E 为本国家标准的一部分。

GB/T 16284 在《信息技术　文本通信　面向信报的文本交换系统》总标题下,目前包括以下 7 个部分:

第 1 部分(即 GB/T 16284.1):系统和服务概论;

第 2 部分(即 GB/T 16284.2):总体结构;

第 3 部分(即 GB/T 16284.3):抽象服务定义约定;

第 4 部分(即 GB/T 16284.4):信报传送系统:抽象服务定义和规程;

第 5 部分(即 GB/T 16284.5):信报存储器:抽象服务定义;

第 6 部分(即 GB/T 16284.6):协议规范;

第 7 部分(即 GB/T 16284.7):人际信报系统。

本标准的附录 B、附录 C 是标准的附录;附录 A 和附录 D 都是提示的附录。

本标准由中华人民共和国电子工业部提出。

本标准由电子工业部标准化研究所归口。

本标准起草单位:清华大学计算机系。

本标准主要起草人:邱建、史美林、李韵琴、徐明伟、张志豪。

# ISO/IEC 前　言

ISO(国际标准化组织)是由各个国家标准化机构(ISO 的成员体)联合组成的一个世界性组织。该组织通过其各个技术委员会进行国际标准的制定工作。凡是对于已设有技术委员会的某一专业感兴趣的每一个成员体,都有权参加该技术委员会。与 ISO 有联系的官方和非官方国际组织也可参与国际标准的制定工作。ISO 与国际电工委员会(IEC)在电子技术标准化的所有方面都进行密切合作。

各个技术委员会提出的国际标准草案,须先分发给各成员体表决通过后,再由 ISO 理事会批准为国际标准。根据 ISO 工作导则,国际标准至少需要投票成员体的 75%赞成。

国际标准 ISO/IEC 10021-3 是由 ISO/IEC JTC1 信息技术　第一联合技术委员会制定的。

目前,ISO/IEC 10021 在《信息技术——文本通信——面向信报的文本交换系统》总标题下,包括以下 7 个部分:

——第 1 部分:系统和服务概论;

——第 2 部分:总体结构;

——第 3 部分:抽象服务定义约定;

——第 4 部分:信报传送系统:抽象服务定义和规程;

——第 5 部分:信报存储器:抽象服务定义;

——第 6 部分:协议规范;

——第 7 部分:人际信报系统。

本标准的附录 B 和附录 C 是标准的组成部分。附录 A 和附录 D 仅提供参考信息。

# 引　言

本标准是一组面向信报的文本交换系统(MOTIS)国家标准之一，这组标准对包含任意多个协同操作开放系统的信报处理提供了综合说明。

本标准使信报处理系统作为一种复合的分布式信息处理任务得以成为可能，它们的组合成分具有这些特性。

本标准规定了信报处理中分布式信息任务的定义约定，也可用于其他应用。

本标准由 CCITT 和 ISO 合作完成。与 CCITT 的 X.407 技术上是一致的。

# 目　　次

中华人民共和国国家标准

# 信息技术　文本通信
# 面向信报的文本交换系统
# 第3部分：抽象服务定义约定

GB/T 16284.3—1996
idt ISO/IEC 10021-3:1990

**Information technology—Text communication —**
**Message-Oriented Text Interchange System(MOTIS)—**
**Part 3:Abstrach service definition conventions**

## 第一篇　引　　言

### 1　范围

本标准说明了用来描述在信报处理中出现的分布式信息处理任务的约定。本标准结构如下：第一篇是引言；第二篇说明了为抽象定义一个分布式信息处理任务而采用的约定；第三篇给出了具体实现这些任务的通信方面的原则，例如根据开放系统互连(OSI)协议实现的原则。附录提供了重要的补充信息。

本标准没有一致性的要求。

### 2　引用标准

下列标准所包含的条文，通过在本标准中引用而构成为本标准的条文。本标准出版时，所示版本均为有效。所有标准都会被修订，使用本标准的各方应探讨使用下列标准最新版本的可能性。

GB 9387—88　信息处理系统　开放系统互连　基本参考模型(idt ISO 7498:1984)

GB/T 16262—1996　信息处理系统　开放系统互连　抽象语法记法一(ASN.1)规范(idt ISO/IEC 8824:1990)

GB/T 16263—1996　信息处理系统　开放系统互连　抽象语法记法一(ASN.1)基本编码规则规范(idt ISO/IEC 8825:1990)

ISO 8649:1988　信息处理系统——开放系统互连——联系控制服务元素的服务定义

ISO/IEC 9072-1:1989　信息处理系统——文本通信——远程操作——第1部分：模型、记法与服务定义

### 3　定义

下述定义适用于本标准。

本标准以GB 9387提出的概念为基础并使用其中定义的下列术语：

a）**抽象语法**；

b）**应用层**；

c）**应用协议数据单元(APDU)**；

d）**应用协议**；

e）**应用服务元素(ASE)**；

国家技术监督局1996-04-10批准　　1996-12-01实施

f) **具体传送语法**；

g) **分布式信息处理任务**；

h) **层服务**；

i) **层**；

j) **开放系统**；

k) **开放系统互连(OSI)**；

l) **实开放系统**。

本标准使用 GB/T 16262 定义的下列术语：

a) **抽象语法记法一(ASN.1)**；

b) **(数据)类型**；

c) **(数据)值**；

d) **输入**；

e) **整数**；

f) **宏**；

g) **模块**；

h) **客体标识符**；

i) **标记**。

本标准使用 GB/T 16263 定义的下列术语：

a) **基本编码规则**。

本标准使用 ISO 8649 定义的下列术语：

a) **应用上下文(AC)**。

本标准使用 ISO 9072-1 定义的下列术语：

a) **结合操作**；

b) **差错**；

c) **已链接的**；

d) **操作**；

e) **远程操作服务(ROS)**；

f) **远程操作**；

g) **离合操作**。

## 4 缩略语

本章无条文。

## 5 约定

本标准使用下列描述性约定：

### 5.1 ASN.1

本标准使用下列基于 ASN.1 表的描述性约定：

a) 使用 GB/T 16262 的 ASN.1 宏记法来定义宏 OBJECT,PORT 和 REFINE。

b) 使用 ISO/IEC 9072-1 的宏定义 BIND、UNBIND、OPERATION 和 ERROR 来定义宏 ABSTRACT-BIND、-UNBIND、-OPERATION 和-ERROR。

c) 使用 ASN.1 本身来说明在附录 A 的例子中出现的信息客体的抽象语法。

d) 使用第 7 章的宏 OBJECT、PORT 和 REFINE 来说明在附录 A 例子中的各种抽象模型。

e) 使用第 8 章的宏 ABSTRACT-BIND,-OPERATION 和-ERROR 来说明在附录 A 例子中的各

种抽象服务。

ASN.1 出现在两个地方：在本标准的正文中用来作为说明；此外，为用户参考，也大量重复地在附录中出现。假如在这两处发现差异，则指示一个规范差错。

注：整个附录中的 ASN.1 模块的 ASN.1 标记是隐式的；就这方面而言，这些模块是确定的。

5.2 术语

在标准中，术语在定义时用黑体字表示，在其他情况下无特殊表示。

## 第二篇 抽象服务定义约定

### 6 概述

面对描述与说明一个复杂分布式信息处理任务的工作，聪明的方法是先抽象地说明任务，而不是用具体术语来说明。这种方法能保证在独立于任务的具体实现情况下，列出任务的功能需求。这种分离法很重要，因为在不同前提下，任务的每一个方面都容许有几个具体实现。例如，在一个包含三个信报传送代理的信报传送系统中，第一个与第二个代理可能采用 OSI 通信来相互作用，而第二个与第三个可能靠专用的方式来交换。

本篇从宏观与微观两方面说明了抽象地描述一个分布式信息处理任务而采用的约定。宏观描述称为抽象模型，微观描述称为抽象服务。

在本篇定义了说明抽象模型与服务的各种形式工具。在附录 A 中给出了一个综合运用的例子。阅读本篇时，读者可能希望参考一下该附录的实例。

本篇包括下列题目：

a) 抽象模型

b) 抽象服务

注：上面提到的形式工具既不是一个形式描述语言，也不是一个置换。它们只是支持本篇定义的非正式描述约定的简单 ASN.1 记法。

### 7 抽象模型

一个分布式信息处理任务的宏观描述称为此任务与其实现环境的抽象模型(模型)。它是以抽象客体、端口、服务和细分等概念为基础的。(抽象服务的概念在第 8 章中有更完整的说明。)

7.1 抽象客体

一个抽象客体(客体)是一个功能实体，是可能的几个两两之间存在交互作用的实体中的一个。客体有不同的类型，这些类型决定了它们的功能与行为。例如，某种类型的一个客体可能表示一个系统，而另一种类型的多个客体表示它的用户。客体之间通过抽象端口交互作用。

客体类型是通过宏 OBJECT 来说明的。此说明列出了对这个客体提供访问的抽象端口的类型。对于非对称的端口类型，规范指示出这种类型端口是消费者端口还是提供者端口。

```
OBJECT MACRO            ::=
BEGIN
TYPE NOTATION           ::="PORTS" "{"PORTLIST"}" | empty
VALUE NOTATION          ::=value(VALUE OBJECT IDENTIFIER)
Portlist                ::=Port","Portlist | Port
Port                    ::=value(PORT)PortType
PortType                ::=Symmetric | Asymmetric
Symmetric               ::=empty
Asymmetric              ::=Consumer | Supplier
```

```
Consumer                    ::="[C]"
Supplier                    ::="[S]"
END
```

类型 OBJECT 的数据值是一个唯一且无歧义地标识被说明客体的客体类型标识符。

注：关键字"OBJECT"是 ASN.1 的保留字。在当前上下文中，合适替换者的选择留待进一步研究。

## 7.2 抽象端口

一个抽象端口(端口)是一个点，在这点上一个抽象客体与另一个抽象客体交互作用。端口有不同类型，这些类型决定了它们能够允许的相互作用种类。例如，某种类型的端口可能表示访问一个目录系统的方式，而另一类端口则表示管理此系统的方法。

端口类型有以下两种：

a) 对称的：对称端口类型的所有实例全是同一的。

b) 非对称的：非对称端口类型的每个实例必是供应者与消费者两种之一。

注：术语"提供者"和"消费者"的特定分配常常是凭直觉进行的。例如，在一个文件系统中，很自然地会考虑将提供者端口分配给系统用户和管理者。但严格地讲，这两个术语的分配是任意的。

只要两个客体的两端口相接触或已结合，则客体间可通过其端口相互作用。对于一个或多个端口对，能使其状态为"启动"的动作称为结合，能使其状态为"关闭"的动作称为离合。

只有当两个端口匹配时才能结合。任意两个相同的、对称类型的端口可以相匹配。两个相同的非对称类型的端口相匹配当且仅当一个是供应者，另一个是消费者。

端口类型是用宏 PORT 来说明的。此规范标识了抽象操作，该操作表示当两个端口已结合时可能进行的交互作用。假如未列出抽象操作，则认为是未说明。

```
PORT MACRO                  ::=
BEGIN
TYPE NOTATION               ::=Operations | empty
VALUE NOTATION              ::=value(VALUE OBJECT IDENTIFIER)
Operations                  ::=Symmetrical | Asymmetrical
Symmetrical                 ::="ABSTRACT" "OPERATIONS" "{"operationlist"}"
Asymmetrical                ::=Onesided | Twosided
OneSided                    ::=Consumer | Supplier
TwoSided                    ::=Consumer Supplier | Supplier consumer
Consumer                    ::="CONSUMER" "INVOKES" "{"operationlist"}"
Supplier                    ::="SUPPLIER" "INVOKES" "{"operationlist"}"
Operationlist               ::=Operation","Operationlist | Operation
Operation                   ::=value(ABSTRACT-OPERATION) |
--数据值标识抽象操作
TYPE
--数据类型标识抽象操作
END
```

如果端口类型是对称的，则两个客体可提供所有列出的抽象操作。若端口类型是非对称的，则具有消费者端口的客体与拥有供应者端口的客体提供的抽象操作是有区别的。

PORT 类型的数据值是一个客体标识符，它唯一且无歧义地标识了说明的端口类型。

## 7.3 抽象服务

抽象服务是一个客体，通过的一个或多个端口向另一个客体提供的一个功能集合。前一个客体称为抽象服务提供者(提供者)，后一个客体称为抽象服务用户(用户)。上述的每一个端口都是对称的或非对

称的。若为后者，则必为消费者与供应者二者之一。

一个抽象服务可有任意多个用户和提供者。每当提供者的抽象服务端口与用户的匹配端口相结合时，则认为在这两上客体之间存在一个抽象联系(或联系)。

抽象服务在第8章中给予说明。

注：应用层抽象服务的服务目的与OSI低层服务的服务目的很多是相同的。

### 7.4 抽象细分

在不同时间可以用不同方式来考察客体。在某些情况下，适宜将客体视为原子。例如，当描述一个客体如何与其外部客体交互作用，即当说明客体的抽象服务时，应将此客体视为原子。在另外一些情况下，又适宜将客体视为合成的，即由其他若干个客体构成。例如，当描述如何实现一个客体时就是这种情况。

成分客体同样具有端口。一些端口是在结构的客体"表面"可见的，另一些则使成分客体之间交互作用。因此，在成分客体中支持提供与使用较少的抽象服务，而成分客体协作的结果提供了结构客体的全部抽象服务。

把一个客体功能分成几个较小的客体的功能称为这个客体的抽象细分(细分)。可以递归地使用细分技术。可以通过提练成分客体本身来展示其内部结构，一直这样进行下去，直到可以将分解得到的新的成分客体认为是原子的为止。

细分是用宏REFINE来说明的。它标识了内部结构正被展示的客体与构成它的成分的客体。每一个成分客体的特征是唯一的或递归的。宏定义还指出了哪些成分客体的端口与其他成分客体的端口相结合，以及哪些端口在合成客体的表面是可见的。

```
REFINE MACRO          ::=
BEGIN
TYPE NOTATION         ::=Object "AS" Componentlist
VALUE NOTATION        ::=value(VALUE OBJECT IDENTIFIER)
Componentlist         ::=Component ComponentList | Component
Component             ::=ObjectSpecPorts
ObjectSpec            ::=Object | Object" RECURRING"
Ports                 ::=PortSpecList | empty
PortSpeclist          ::=PortSpec PortSpecList | PortSpec
PortSpec              ::=value(PORT)PortSide PortStatus
PortSide              ::=Consumer | Supplier | empty
Consumer              ::="[C]"
Supplier              ::="[S]"
PortStatus            ::="VISIBLE" | "PAIRED" "WITH" ObjectList
ObjectList            ::=Object","ObjectList | object
Object                ::=value(OBJECT)
END
```

类型REFINE的数据值是一个客体标识符。

注：象客体一样，原则上在不同时间也可以用不同方式来考察端口。在某些情况下，适宜将一个端口(对)视为原子。但是，另一些情况下，可以想象用提练端口自身的方法来检测提供此类通信的机制。由此观点，可以认为一个端口对是由一个客体集支持的。这种看法将加强说明通信性能的能力。在本标准版本中不再进一步探讨"端口提炼"的概念。

## 8 抽象服务

一个分布式信息处理任务的微观描述是抽象服务的规范，它定义了如何启动、控制和中止一个任

务。不明确提出微观描述，是以抽象的结合操作、离合操作、操作和差错以及抽象规程的使能概念为基础的。

注：下面定义的宏定义隐含使用 ASN.1 来说明自变量、结果和参数。例如，在说明过程中赋值的任何具体上下文标记，虽然在上下文中无意义，但在一个抽象服务的 ROS 实现过程中却起着重要作用。

## 8.1 抽象规程

一个抽象规程（规程）是一个客体应另一个客体的请求而执行的任务。发出请求与执行任务分别称为规程的调用和执行。发出请求的客体与按请求动作的客体分别称为调用者与执行者。

规程可以（但不必须）要求调用者在调用的基础上，向执行者提供一个单一的已规定类型的信息客体，该信息客体称为规程的自变量。

每一个规程的每次执行都有一个成功或失败的结果。一个规程如果完全执行则认为是成功的，如过早终止则为失败。

一个规程可以（但不必须）要求执行者将成功通知调用者。

它还可以（但不必须）进一步要求，当报告失败时提供某种信息。

注：以下各条中，规定 ASN.1 明确指出作为说明规程的自变量与结果的抽象语法（对抽象错误的参数也一样）。ASN.1 的这些使用并不暗示在开放系统之间必须传送这些信息客体。特别是，信息客体由于用 ASN.1 进行描述，以及 ASN.1 的基本编码规则使得信息客体具有具体的传送语法这个事实在当前上下文中并不重要。ASN.1只是一个形式描述信息客体抽象语法的方便工具。

## 8.2 抽象结合操作

一个抽象结合操作是一个规程，它的成功执行将一对或多对抽象端口结合在一起。调用抽象结合操作的客体叫做发起者，执行这个操作的客体叫响应。

抽象结合操作适于将发起者的一个特别集与响应者的一个匹配相结合。对集合中一个或多个端口是非对称的情况，抽象结合操作可能只适于结合到消费者一边，或只适于到供应者一边，或二者任选其一。

除了在失败时传给调用者的信息只限于一个称作差错信息的单一信息客体这种情况之外，抽象结合操作完全是一个一般的规程。

一个抽象结合操作是靠宏 ABSTRACT-BIND 来说明的，定义如下：

```
ABSTRACT-BIND MACRO          ::=
BEGIN
TYPE NOTATION                ::=Ports Bind
VALUE NOTATION               ::=value(VALUE BindType)
Ports                        ::="TO" "{"PortList"}" | empty
PortList                     ::=PORT","PortList | Port
Port                         ::=value(PORT)PortSide
PortSide                     ::=Consumer | SuppLier | empty
Consumer                     ::="[C]"
Supplier                     ::="[S]"
Bind                         ::=type (BindType)--必须是 BIND 类型
                                  |empty<BindType  ::=Bind>
END
```

由关键字"TO"引导的"PORTS"条列出了此抽象结合操作要结合的响应者的端口，假如在该列有一个非对称端口并未标"[S]"或"[C]"，这表示此抽象结合操作适用于在任一方向结合这样一个端口。

注意自变量、结果、与/或差错信息的说明是靠 ISO/IEC 9072-1 中定义的远程操作的宏 BIND（嵌入的）来完成的，而且它是宏返回的这种类型的一个值。若未提供返回值，则返回默认值"BIND"。

注：ABSTRACT-BIND 与 BIND 之间的联系可有助于具体化抽象服务的 ROS 实现。见 10.1 条。

一个抽象服务对于它提供的每种类型端口全典型地包含一个抽象结合操作。当存在几种端口类型时，它们的抽象结合操作可能但并不一定有区别。

8.3 抽象离合操作

一个抽象离合操作是一个规程，它的执行无论成功与否都将使两端口分离。它由调用相应的抽象结合的客体（即发起者）调用，由响应者执行。

一个抽象离合操作适用于将发起者的端口特定集与响应者的匹配集处分离。当在集合中一个或多个端口是非对称时，抽象离合操作只适用于从消费者方面分离，或只适于从供应者方面分离，或两者任选其一。

除了在失败时，传送给调用者的信息只限于称为差错信息的单一信息客体这种情况外，抽象离合操作是一个完全一般的规程。

一个抽象离合操作是靠宏 ABSTRACT-UNBIND 来说明的，定义如下：

```
ABSTRACT-UNBIND MACRO        ::=
BEGIN
TYPE NOTATION                ::=Ports Unbind
VALUE NOTATION               ::=value(VALUE UnbindType)
Ports                        ::="FROM" "{"PortList"}" | empty
PortList                     ::=Port","PortList|Port
Port                         ::=value(PORT)PortSide
PortSide                     ::=Consumer | Supplier | empty
Consumer                     ::="[C]"
Supplier                     ::="[S]"
Unbind                       ::=type(UnbindType) |
                                --必须是 UNBIND 类型
                                empty <U nbindType   ::=UNBIND>
END
```

由关键字"FROM"引导的"PORTS"条列出了此抽象离合操作要离合的响应者端口，假若在那有一个未标"[S]"或"[C]"的非对称端口，则表明此抽象离合操作适用于在任一方向离合这种端口（虽然真正的方向所决定的）。

注意自变量、结果、与/或差错信息的说明是由 ISO/IEC 9072-1 中定义的远程操作的宏 UNBIND（嵌入的）来完成的，而且它是宏返回的这种类型的一个值。若未提供返回值，则返回缺省值"UNBIND"。

注：ABSTRACT-UNBIND 与 UNBIND 之间的联系可有助于具体化抽象服务的 ROS 实现。见 10.1 条。

一个抽象服务对于它提供的每种类型端口全典型地包含一个抽象离合操作。当存在几种端口类型时，它们的抽象离合操作可能但并不一定有区别。

8.4 抽象操作

抽象操作是一个规程，它可以在两个已结合端口的上下文中被调用。它的失败对于结合无影响。若端口是非对称的，则端口预先规定了调用者含有消费者端口的客延体还是含有供应者端口的客体，或二者之一。若端口是对称的，则调用者可以是二者中任一客体，无论端口对称与否，剩下的一种客体即为执行者。

除了在失败时向调用者传送信息之外，抽象操作可以说是一个完全一般的规程。一个抽象操作当遇到抽象差错时失败，并且传送的信息只限于抽象差错要求的通告那些信息类型。每个抽象操作全都预先规定了差错是否要报告及若报告时，会遇到哪种的抽象差错。

一个抽象操作是靠宏 ABSTRACT-OPERATION 来说明的。它的定义与 ISO/IEC 9072-1 中说明

的远程操作宏 OPERATION 定义是同一的。

ABSTRACT-OPERATION MACRO ::=OPERATION

一个抽象服务对于它提供的每种端口类型全包含零或多个抽象操作。当包含几种端口类型时，它们可能但不一定具有相同的抽象操作。

注：ABSTRACT-OPERATION 与 OPERATION 的等价性有助于具体化抽象服务的 ROS 实现。见 10.1 条。

### 8.5 抽象差错

一个抽象差错是在抽象操作执行过程中可能出现的一种例外情况，它造成执行失败。当一个抽象差错被报告时，执行者向调用者传送抽象差错的标识，可能还有一个称为参数的单一信息客体。对每一个抽象差错全都预先规定了是否要返回一个参数及若返回参数时，其为何种类型。

抽象差错是由宏 ABSTRACT-ERROR 来说明的。它的定义与 ISO/IEC 9072-1 中说明的远程操作的宏 ERROR 定义同一。

ABSTRACT-ERROR MACRO ::=ERROR

一个抽象服务包含它的抽象操作报告的零个或多个抽象错误。

注：ABSTRACT-ERROR 与 ERROR 的等价性有助于具体化抽象服务的 ROS 实现，见 10.1 条。

## 第三篇 抽象服务实现

## 9 概述

一旦用抽象术语描述与说明了一个分布式信息处理任务，就必须规定具体实现任务每一方面的方式。象前面所建议的，每一方面都允许有几种具体的实现。

本篇说明了具体实现抽象模型与服务的原则。一个实 X 是计算机进程或系统，或者是具体实现类型 X 的抽象客体的实开放系统。

本篇包含以下题目：

a) OSI 实现

b) 专有实现

注：对于抽象模型的诸方面，在这里强调的是抽象端口与它们的结合。这是因为抽象端口不仅在抽象客体之间而且在具体实现这些抽象客体的物理系统之间标明界限。因此，抽象端口与其结合是抽象模型的一部分，假如要实现开放系统互连，则此抽象模型一定是已用 OSI 工具构成或可用 OSI 工具构造的。

## 10 OSI 实现

CCITT 建议与国家标准的基本目的是要说明几个协作实开放系统如何来实现分布式信息处理任务。

在 OSI 环境中，客体是由应用进程来实现的。通常客体与应用进程之间存在多对多的映射。在不同开放系统中由应用进程实现的客体之间的通信是根据 OSI 应用协议(包括应用上下文)完成的。一个应用上下文实现一些端口对的结合，使用与离合。

应用上下文的规范是以一些应用服务元素的协操作为根据的。假如定义应用服务元素与支持通信的每个端口相对应，则可以特别直接简单地对实现进行说明。

下面根据 ASEs 与 ACs 对抽象端口与结合的实现进行说明。并要考虑 ROS 与 NON-ROS 两个方面的实现。

### 10.1 ROS 实现

由远程操作完成的端口与结合的具体实现通常并不重要。

首先这是因为在有一个功能上等同的基于 ROS 的应用协议的情况下，对抽象服务进行定义是件直接的简单的事情。此外，又是因为抽象服务的规范框架与基于 ROS 的应用协议规范框架同构。在表 1 中

列出了同构中的对应关系。

表 1　抽象服务和基于 ROS 的协议的对应关系

| 抽象服务方面 | 基于 ROS 的协议方面 |
|---|---|
| 抽象结合操作 | 结合操作 |
| 抽象离合操作 | 离合操作 |
| 抽象操作 | 操作 |
| 抽象差错 | 差错 |

用联系密切或等价的宏的形式说明对应的方面，从而得出表中的对应关系，正如表 2 所示：

表 2　等价抽象服务和 ROS 宏

| 抽象服务宏 | ROS 宏 |
|---|---|
| 抽象结合 | 结合 |
| 抽象离合 | 离合 |
| 抽象操作 | 操作 |
| 抽象差错 | 差错 |

在附录 A 中将通过例子，探讨具体实现抽象端口的基于 ROS 的 ASE 和 AC 的定义。

为细化实现，很有必要存在一个可将所有必须成对的端口结合的抽象结合操作。

注：若在抽象服务中含有多于一个的端口(对)，则要求针对含有的特定端口设计抽象结合操作。当前未被提供适合抽象结合的自动合成，这些抽象结合可假设为以单独端口抽象结合操作的定义为基础。

### 10.2　NON-ROS 实现

用非远程操作的方法来完成端口与结合的具体实现需要更大量的工作，并且毫无通用的一般方法可言。

尽管如此，下面的两个论点是恰当的：

a) 使用 ASN.1 来定义 APDU 将大大简化作为应用协议的抽象服务的具体实现。这是因为协议规范可以简单地从抽象服务规范输入相应的类型和值。

b) 抽象操作不通告其结果的抽象服务的具体实现，这在概念上是简单的。因为每个这样的抽象操作表示一个包含单一 APDU 的交互。从所有可能的交互作用中最简单的出发，可以构造出任意复杂的交互。

## 11　专用实现

CCITT 建议和国家标准的第二个目标是要保证在用专用方式实现部分分布式信息处理任务的同时，不影响系统预计的整体功能。

下面将简要地讨论用专用方法实现抽象端口与结合的问题。并要考虑分布式与非分布式两个方面的实现。

### 11.1　分布式实现

用专用的计算机通信协议来具体实现端口与结合是一个局部问题。体现在抽象服务中的可见功能规范为专用实现的实现者提供了指导。因此，在适于使用这种实现的地方，它们可能在总体任务中起适当的作用。

### 11.2　非分布式实现

以单一计算机内部机制来具体实现端口与结合是一个局部问题。如同在 11.1 条考虑的情况下，作为实现者指导的抽象服务规范保证专用实现仍然能在总体任务中起到适当的作用。

# 附 录 A
（提示的附录）
## 抽象服务记法使用举例

本附录通过一个实例说明了抽象模型和服务记法的使用。实例包括两个系统，黄色系统和绿色系统以及它们的环境，黄色和绿色环境。

它使用抽象模型记法来分别描述环境(A2 和 A4)并且显示了它们的系统是如何相关的：一个是由另一个构成的(A6)。它使用抽象服务记法来描述每个系统的能力(A3 和 A5)。最后，使用能适合 OSI 通信的 ISO/IEC 9072-1 的 ROS 记法实现了系统的端口，比如 AC 和 ASE(A7 和 A8)，从而结束了本例。

### A1 客体标识符的赋予

在本附录中定义的 ASN.1 模块需要赋予不同的客体标识符。下面用 ASN.1 定义了所有模块。除了 ASN.1 模块和应用服务定义约定的主题本身以外，所有的赋值都是正确的。前者的确定赋值出现在模块本身；对它们的其他参考在 IMPORT 条中。后者是固定的。

```
ExampleObjectIdentifiers{joint-iso-ccitt
    mhs-motis(6)asdc(2)example(1)modules(0)object-identifiers(0)}
DEFINITIONS IMPLICIT TAGS      ::=
BEGIN
--序言
--输出任何东西
输入--无--；
                                ID::=OBJECT IDENTIFIER
--抽象服务定义的约定举例(不确定的)
id-asdc-ex                      ID::={joint-iso-ccitt mhs(6)asdc(2) example(1)}
    --不确定的
--种类
id-mod                          ID::={id-asdc-ex 0}--模块；不确定的
id-ot                           ID::={id-asdc-ex 1}--客体类型
id-pt                           ID::={id-asdc-ex 2}--端口类型
id-ref                          ID::={id-asdc-ex 3}--提炼
id-ac                           ID::={id-asdc-ex 4}--应用上下文
id-ase                          ID::={id-asdc-ex 5}--应用服务元素
id-aa                           ID::={id-asdc-ex 6}--抽象语法
--模块
id-mod-object-identifiers       ID::={id-mod 0}--不确定的
id-mod-ye-refinement            ID::={id-mod 1}--不确定的
id-mod-y-abstract-service       ID::={id-mod 2}--不确定的
id-mod-ge-refinement            ID::={id-mod 3}--不确定的
id-mod-g-abstract-service       ID::={id-mod 4}--不确定的
id-mod-ys-refinement            ID::={id-mod 5}--不确定的
id-mod-ys-realization           ID::={id-mod 6}--不确定的
id-mod-gs-realization           ID::={id-mod 7}--不确定的
```

```
--客体类型
id-ot-y-environment                ID::={id-ot 0}
id-ot-y-user                       ID::={id-ot 1}
id-ot-y-system                     ID::={id-ot 2}
id-ot-g-environment                ID::={id-ot 3}
id-ot-g-user                       ID::={id-ot 4}
id-ot-g-manager                    ID::={id-ot 5}
id-ot-g-system                     ID::={id-ot 6}
id-ot-g-agent                      ID::={id-ot 7}
--端口类型
id-pt-y-use                        ID::={id-pt 0}
id-pt-g-use                        ID::={id-pt 1}
id-pt-g-management                 ID::={id-pt 2}
--提炼
id-ref-y-environment               ID::={id-ref 0}
id-ref-g-environment               ID::={id-ref 1}
id-ref-y-system                    ID::={id-ref 2}
--应用上下文
id-ac-y-use                        ID::={id-ac 0}
id-ac-g-use                        ID::={id-ac 1}
id-ac-y-management                 ID::={id-ac 2}
--应用服务元素
id-asc-y-use                       ID::={id-ase 0}
id-asc-g-use                       ID::={id-ase 1}
id-asc-g-management                ID::={id-ase 2}
--抽象语法
id-as-y-use                        ID::={id-as 0}
id-as-g-use                        ID::={id-as 1}
id-pt-g-management                 ID::={id-as 2}
END--of ExampleObjectIdentifiers
```

## A2 黄色环境细分

图 A1 中描述的黄色环境用宏 OBJECT 和宏 REFINE 形式细分如下：

正如图 A1 中指示和下面 ASN.1 规范证实的那样，黄色环境可被模型化为一个客体，此客体可拆成一个中心客体——黄色系统和任意多个其他外围客体——黄色用户(Yellow-user)。黄色系统和黄色用户通过其黄色应用端口交互。