

## 第12章 电子商务和XML

尽管WWW页面在初始创立时是为了交换和显示静态的、人可以读的文档，但它却迅速发展成为全球各类组织进行通信、广告和消费的渠道。

具有买卖功能的Web页面成为电子商务的一个部分，或叫做电子商务。由于电子商务已经成为一个被滥用并经常用错的一个词条，下面的解释是我个人对这个词条的理解：

电子商务是两个或多个实体通过Web进行交换货物或服务以获得货币的行为。

Web从交换科技文献到进行买卖商品和服务是一个巨大的发展，如何建立一个电子商务网站可以写成一本书，但本章并不打算提供创建一个电子商务网站的解决方案。由于在电子商务中对XML有许多错误的概念，本章将解释XML在电子商务中的一些关键问题。不幸的是电子商务相当复杂，人力资源敏感和昂贵，从这个市场上获得真正的回报还有一段漫长的路要走。XML语言的出现并不是为了使电子商务的应用易于开发，虽然毫无疑问它改变了通向电子商务应用之路的面貌，而这也是我们关注它的原因。

本章将分成两个部分。第一部分是理论部分，将介绍一些传统的EDI到电子商务的理论，表明通过使用XML将产生怎样的解决方案，事实上它很像现存的EDI应用——虽然XML体现了电子商务的优点，而这也是大量使用XML语言的原因。

我们将会发现，一些关键的词条正在移入到XML表达式中，同时也产生了一些错误的概念。第一部分将以XML在电子商务领域的发展前瞻作为结束，并说明它将带来比EDI标准更多的东西。在本章的第二部分将会有许多实例。这个部分关注一些现存的为横向领域而开发的DTD模式以及一些更为复杂的消息。接下来是这个新的电子商务领域实现方案，包括Microsoft的BizTalk模式。

在本章我们将会看到：

- 现存的EDI电子商务解决方案是如何工作的。
- XML如何提供一种用于定义商务词汇的机制，使我们得以在不同的应用甚至不同的平台之间共享商业数据。
- 在电子商务中使用XML的一些错误的概念、领会和动机。
- 电子商务未来发展的方向。
- XML在纵向行业领域的实现例子。
- XML在横向行业领域的实现例子。
- 对Microsoft的BizTalk模式的介绍。

人们想知道的是XML如何运用在供应商中，怎样处理当前的事务以及技术如何发展。本章介绍的就是在实现全球电子商务解决方案中所包含的目标、趋势、错误的概念、范围以及初始的工作。

## 12.1 什么是电子商务

如果我们脱下电子商务的外衣，将会发现电子商务只是两方或多方通过电子渠道交换一些种类的商业信息。它并不在意卖些什么，从光盘到汽车部件到智力财富，目的只是一方按双方预先规定的格式从另一方获得某些类型的信息。

大的公司和银行为了支持信用卡商业服务、ATM机、帐户明细交换等业务，卷入电子交易已经有几十年的历史。通常，公司为了这些服务，必须使用增值网络（Value Added Networks, VAN）以共享数据。但从1994年开始，由于Internet的流行导致新的商业模式的出现，该方式产生了值得重视的变化。电子信息的共享不再局限于大的公司，许多小的和中等规模的企业（简称SME）也加入了进来，甚至包括个人。

大公司是电子信息交换的先锋。这些公司传统的通信模式，及提出的解决方案叫做EDI（电子数据交换）。但是由于这些系统出奇的昂贵并难以实现，实施的障碍很多。所以在本章中我们将大概介绍EDI交换是如何产生的，接下来介绍Internet和XML怎样改变了通信的模式。

我们刚刚才开始了解通过Internet传送的XML能够改变不同人之间的商业交互。人们希望XML的发展趋势是扩展和使用可以利用的技术以获得信息技术时代的竞争利益。此外，另一个主要的发展趋势是共享其他企业和组织不断发展和变化的信息，这也是新的电子商务应用所定义和实现的领域。电子商务和大量消息的超集就叫做eBusiness。

历史成就了今天，发展才有更好的开始。现在是学习和使用这些工具的时候了，我们将通过面向世界范围的商业听众，用这新的商业语言获得更多的利益。许多这样的公司对处理自动化感兴趣，但对于表述他们的语言或是维护、研究和驱动他们的开发、流程或交换商业文件没有有效的资源和工具。现在，他们有了。

一提起电子商务，我们中的许多人马上就会想到向公众销售产品和服务的Web站点。但是许多大的站点的背后都有通过EDI或特定协议和格式进行的定单自动处理程序。另外，还有大量的通过供应链进行的电子商务交易。（供应链是资产、服务、信息和操作的联合体，以满足最终客户的需求，它可以包括其他自治公司。）同时部门间共享更多的数据也成为快速增长的趋势。让我们看看在电子商务中的三种业务模式：

- 对客户直接销售。
- 企业对企业交易。
- 信息共享和内容联合。

在这些模式上有许多变化，但它们给出了XML在电子商务领域中运用的基础。

### 12.1.1 对客户直接销售

这是我们都熟悉的一种模式，也是许多人在谈到电子商务时最先想到的模式。通常这种模式的例子是Amazon.com和Dell直销。通过使用一个标准的浏览器我们可以访问他们的网站浏览他们的产品。如果发现了喜爱的东西，我们可以通过使用信用卡

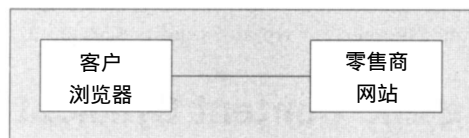


图 12-1

进行购买，并在自家门口得到它（参见图 12-1）。

这个模式不仅适合于大型公司，它同样为向小的潜在市场生产或销售特定产品的许多小公司提供了生命线。虽然这种模式被 Microsoft 和 IBM 等公司提倡，但很少有小公司将 Internet 作为一个直接销售渠道。如同我们在本章后面将要看到的那样，新的工具将帮助 SME 利用 Internet。

传统制造业将他们的利益直接同潜在的上网客户挂钩，通过弃用中间商以寻找更高的回报，这叫做直销（disintermediation）。这些企业面临的挑战包括直接客户营销、客户反馈、小型定单处理，这些都同当前的销售渠道不同。但是一些企业修订了他们的直销策略，使用了在线分销商，将他们看成发布和实现中心（参见图 12-2）。

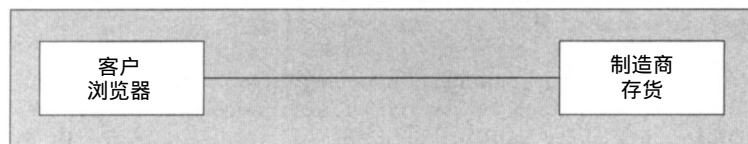


图 12-2

### 12.1.2 企业对企业交易

企业对企业交易（B2B）的含义同对客户直接销售类似，但交易发生在贸易伙伴之间。但是这些大的公司在下定单时不会每次都使用他们的信用卡，而是同他们的供应商进行帐户上的资金划转，并且他们的交易经常使用预定义的共享语言。

使用这种模式对公司的关键利益在于少量的纸质文书处理，这样定货将会更快捷。一些交易甚至可以通过在线自动处理，这样对实时定货和撤消定货提供了便利。如果自动处理实现了当日定货，运行经理就可以注意于一些处理流程的异常情况（参见图 12-3）。

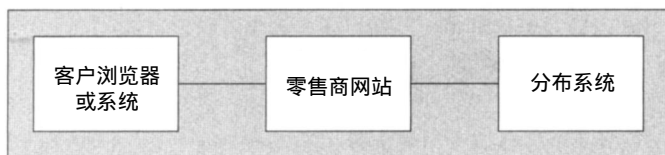


图 12-3

通常，当公司自动处理他们的交易时，将对客户的需求进行选择或响应。这只会对交易产生微小的影响。修改自己商务惯例的公司将获得真正的成本节约，但自动化会带来成本，集成要求大量的前期投资、裁减职位并要求维护。结果一些公司转向 rip and read 处理。它是指一旦接到一个交易，将打印出来并通过纸质文件的流程进行处理或重新输入到公司的计算机中。这个流程对小的企业没有限制，许多大型企业也会在它的某个部门使用这个方式进行日常处理。XML 承诺之一就是向这些组织提供实现自动化的工具，并有更少的技术复杂性，取消对 rip and read 处理的依赖。

### 12.1.3 信息共享和内容联合

我们看到的最后一种模式是信息共享。它有许多形式，它可能是通过卫星向总部上报的销

售数据的汇总，也可能是智力产品的联合——如音乐、影像、新闻或证券市场公司销售的股票市场分析报告。供给链的比喻不仅可以运用在不同地点的货物上还可以使用在服务产品上（参见图12-4）。

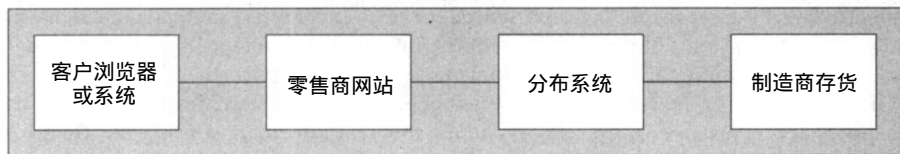


图 12-4

如果把90年代看成是企业执行企业资源规划的十年（ERP：集成的应用软件套件，应用在制造、分销和财务管理，使企业优化他们的商业流程，进行必需的管理分析和目标市场营销），那么接下来的十年必然注重企业的扩展和企业内部的处理。信息共享自动化将给企业带来另一种利益并创造新的商业模式。

信息共享还将会产生信息中间商，他们通过向多个供应商收集资本、价格、能力甚至在线拍卖等信息以向企业提供信息服务（参见图12-5）。

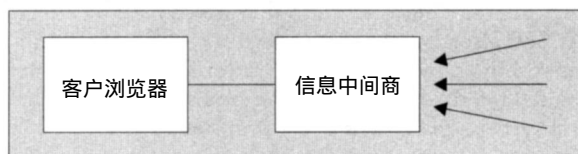


图 12-5

为了了解XML如何改变了电子商务的外貌，我们应该先看看 EDI是如何为电子商务做准备的。

#### 12.1.4 EDI——电子数据交换

EDI在过去的30年间更换了几种格式。它是电子商务的铺路人。在 Web和Internet被人所知以前，公司通过使用各种行业交易标准进行电子数据交换。例如在汽车、分销和电子行业领域，通过使用EDI节省了成本，提高了商业处理能力。在 80年代早期，ANSI就发现了制定电子商务标准的优势，并成立了一个正式的EDI标准委员会，代号为X12。

没有人知道实现EDI的组织有多少个，但可能的数量有100 000或更多。虽然我们将看到，新的EDI开始注意同XML的互操作性。

在北美早在70年代初期，EDI标准就已经被大的公司和政府组织用来交换金融、制造、库存、技术和运输信息。最被人所知的是X12标准，这个标准现在由一个叫做数据交换标准协会（DISA）的非赢利组织维护（<http://www.disa.org>）。

其他一些组织和机构执行的是用于管理、商业和运输的联合国电子数据交换标准(UN/EDIFACT)。该标准由联合国欧洲经济委员会(UNECE)下属的管理、商业和运输便利中心(CEFACT)制定。要了解更多的关于UN/EDIFACT的信息请参阅<http://www.unece.org/trade/untdid/welcome.htm>。

同时存在的还有一些用于 B2B 交换的标准,如银行间的金融协议(“有线传输”),主要用于控制 ATM 国际网络和信用卡验证终端。许多国际财团将 X12 和 UN/EDIFACT 作为基本的 EDI 行业标准。例如许多世界知名的医疗和保险公司通过使用健康等级 7 (HL7) 作为医药信息交换标准。

EDI 技术初步预言了 Internet 的商业应用,EDI 不仅包括了特定的信息格式,还定义了通信协议,甚至一些硬件要求。实现这个系统的费用限制了该技术发展的市场,许多资金都浪费在购买系统特定的软件和硬件上,但常常将 EDI 同公司使用的其他系统混合。仅从实际角度来说,只有政府机构和大的公司才有完全实现 EDI 能力的资源。但是 90 年代在 Internet 上的进行的商务探索,产生了更现代和更简单实现电子商务的标准的的需求。所以几乎所有当前的 EDI 标准都被修订以使用 XML 作为信息和传输的基础。

但是 EDI 是怎样开拓了通向新的电子商务领域之路的呢?这是一个十分重要的问题。我们需要知道在 XML 之前 EDI 可以提供什么,这样才能知道 XML 的立足点。我们应该知道 EDI 已经提供了通过电子手段交换信息的方法,这样允许公司通过电子方式共享信息——在 XML 想到以前。由于在传统的 VAN 上的带宽限制,这些消息使用压缩后的一种难懂的形式通过特定的协议进行传输。但是 EDI 系统却提供了成熟的传输处理、贸易规则管理、日志和备份、商务和法律请求处理以及错误处理等一些特征。在这之上是应用集成的 API、版本控制和异常报告。我们关注的并不是 EDI 使用的软件和硬件,而是要看看在一个 EDI 交易或“商业会话”中交换的是什么。

### 1. 商业会话

EDI 语言要求交换商业语义。如同企业可以通过共享通用的 XML DTD 或模式以保证数据可以共享,EDI 文档提供共享商业信息的标准。

现存的 EDI 标准基于交易的概念,它将包含在消息中发布,预先定义和周知格式,并使用预定义的协议进行传输。协议一般是静态的,并基本使用硬件实现,消息格式是动态的,必须在多个组织间共享,使用通常接受的模式并一般存储在容器中。模式详细描述了数据对象格式并被交易各方所认可。许多 EDI 容器被非赢利组织或国际财团进行管理,以避免任何单一的公司过分影响这些重要交互数据的设计和实现。

如果你认为这个同 XML 中行业 DTD 协议或其他用于共享数据的模式概念相似,那无疑是正确的。消息和模式的集合同 XML 文档相似。事实上,在本章的后面我们将会看到,许多 XML 容器建立起来以提供共享 XML 模式的服务。下面我们看看在 EDI 交易中会发生什么。

EDI 提供了一个大的标准消息集合用于交易。图 12-6 表示了一个使用 X12 协议在买方和卖方进行交易的流程。数字代表交换的消息类型:

这样 EDI 提供了标准的消息类型,通过数字编号或代码进行表示。你可以看到,卖方先发送一个贸易伙伴特性文件 (838), 和一个分类目录 (832)。这样买方就知道可以获得的产品和服务以及怎样同卖方交易,于是发送一个询价请求或 RFQ (840) 消息给卖方。卖方将产生一个响应消息。在这个例子中,将产生一个购买定单 (850), 交易将继续进行直到买方收到发票 (810)。

在 X12 或 EDIFACT 系统中有上百个具有不同目的的消息,并都具有不同的代码以方便交易



的双方进行通信。其中的一些用于特殊的行业领域，如用于汽车销售、健康保险、运输、抵押和学生贷款等。但是，这些消息的开发和说明相当昂贵，有较大的使用障碍。



图 12-6

你可能会猜测，当企业对这些标准消息进行客户化时经常会应合作伙伴的要求添加一些附加信息，这样会有更高的开发要求。可以通过使用客户化内容（同合作伙伴商定）或 EDI 允许的临时定义来实现。理论上，当使用这些扩展时，应使用标准的提交给标准机构以备将来的版本采纳。不过，你可能想象得到，一旦按临时代码实现交易，那么几乎不可能。

下面我们来看看 EDI 消息是什么样子的。我们将看看在 X12 标准中两个购买定单的例子。

## 2. EDI 消息结构

一个 EDI 消息包括数据段和需要翻译的元素。我们已经注意到，制定数据段顺序的机制已经成为制定 XML DTD 功能的经验。数据段分成永久的、可选的或有条件的，并可以重复预定的次数。每个数据段中的 EDI 元素同样在数据标准中制定，包括数据种类属性以及最小、最大值。

这里我们可以看到，购买定单被封装在一个信封中，包括开始的报头 and 最后的报尾。报头包括路由信息、使用标准的版本号、控制计数器、时间戳以及一个标志，该标志表明该交易是否应被正常处理或仅仅是一个测试交易。在信封内的是控制信封，用于定义功能组（交易类型）以及文档的起末位置。报尾包括功能组数量以及交易定义标记，它将同报头相匹配。

在这个消息中有两个购买定单，每一个要求有 4 项（参见图 12-7）。

除非你对 X12 很熟悉，否则该消息非常难懂——你没法说出这是一个购买定单，更不知道它是要买些什么。花费时间去学习它的结构和内容就像在学习一门外语。但是最终用户不会看到这种形式，他们拥有一个具有良好用户界面的应用程序，所以它对我们并不重要，我们只是用

它来做一个表达式的例子。由于使用 EDI系统的公司同外面的世界共享数据经常使用同公司内部不同的语言，这样在 EDI系统中就有对照表工具用于将外界的标准消息转成公司内部使用的格式。

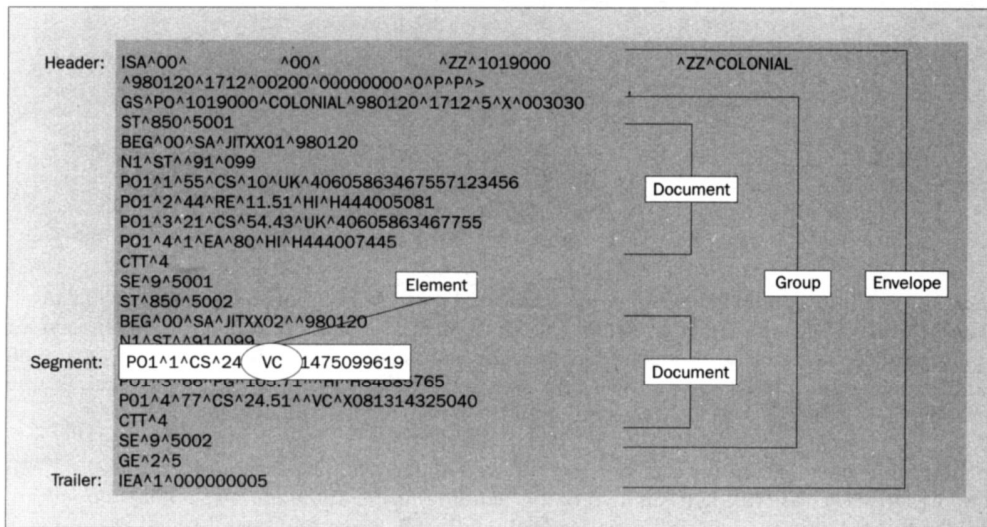


图 12-7

### 3. 将EDI表达式转换成XML

我们仔细看看该消息的一个数据段：交换控制报头。看看我们该如何使用 XML实现X12方法。交换控制报头（通常称为 ISA）用于启动并定义一个交换。我们选择它的原因是它在转成XML时可以强化。这个例子中使用控制号作为发送者指定的 ISA值。ISA是唯一具有固定长度的数据段，具有指定位置的描述，该描述在 X12中是用户定义的（参见图 12-8）。

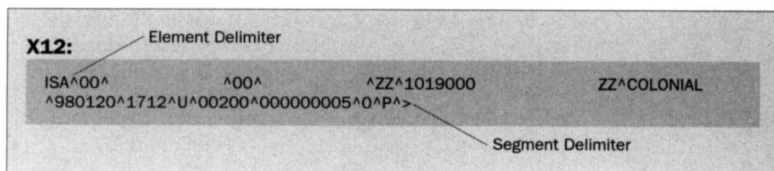


图 12-8

下面是使用XML实现的例子：

```
<ISA AuthorizationQual='00' Authorization=' ' SecurityQual='00' Security=' '
SenderQual='ZZ' Sender='1019000' ReceiverQual='ZZ' Receiver='COLONIAL'
XchgDate='980120' XchgTime='1712' StdAgency='U' StdVersion='00200'
AckReq='0' Usage='P' >000000005</ISA>
```

你可以看到，X12表达式相当紧凑，这是由于低可用、高成本的带宽。但是，当带宽提高时，成本下降了，就不再需要这种复杂的考虑。你可以发现 XML版本十分易于阅读和理解，这样同X12相比有更简单的词汇表。

在XML中，具有NULL值的属性不是必须的，它们在这里出现的原因只是利于说明——利于比较X12和XML表达式。

表12-1是XML中使用的属性的含义，它同 X12语法中定义的长名相对应。

表 12-1

属 性	长 名	目 的
AuthorizationQual	Authorization Information Qualifier	定义授权信息中的信息类型
Authorization	Authorization Information	发送方附加的定义或授权数据，或交换中的交换数据
SecurityQual	Security Information Qualifier	定义安全信息的信息类型
Security	Security Information	定义发送方的安全信息或交换数据
SenderQual	Interchange ID Qualifier	指定的系统/方法代码结构用于指定的保留的发送方 ID 元素
Sender	Interchange Sender ID	发送方定义的代码作为接收方 ID 以向其路由数据
ReceiverQual	Interchange ID Qualifier	指定的系统/方法代码结构用于指定的保留的接收方 ID 元素
Receiver	Interchange Receiver	数据接收方定义的代码
XchgDate	Interchange Date	交换日期
XchgTime	Interchange Time	交换时间
StdAgency	Interchange Control Standards Identifier	用于消息控制标准的响应代理的指定码，附加在交换的报头和报尾
StdVersion	Interchange Control Version Identifier	交换控制数据段的版本号
AckReq	Acknowledgment Requested	发送方发送的代码，请求接收方的交换确认
Usage	Usage Indicator	指示在交换信封中的数据是否是测试数据，还是产品或信息

并不建议现在的 EDI 标准被直接映射成 XML 来实现，重要的目的是抽取丰富的语义并应用在基于 XML 的应用计算技术中。

当使用 XML 实现这种电子商务解决方案时，这种融合叫做 XML/EDI。

无论你是否理解 EDI 语法，我能肯定你一定确信使用 XML 会使获取过程的会话更易读懂。但是还有一个更有力的原因在 EDI 风格的消息中使用 XML，事实上，XML 允许我们使用标记连接程序的某个部分（脚本、组件等等），这样就能按照我们的商业规则对处理流程进行有力的控制。

XML 标记语法非常冗长，但简化了消息的内聚，允许出现更智能的系统。它允许在处理中使用附加的信息作为对数据本身的补充。

但是什么是简化语法的重要性呢？在 XML/EDI 中我们在寻找一种方法标记信息，这样就不需要在内部格式和信息交换标准中进行转化。

使用 XML 可以封装词汇表，这样就可以同时在组织内部和外部使用，从工作流到查询数据库，直到同贸易伙伴交换信息。



同使用对照表不同的是，通过使用 XML，我们可以创造丰富的词汇表用于组织内部和同外部贸易伙伴的信息交换。这样，我们的 XML/EDI 格式就不需要转化了。

如果我们考虑到这一点，可能许多现存的 XML/EDI 表现形式将成为全球的一般机制模式，能将不同的 EDI 和工作流标准融合到 XML 语法中。在接下来的几年中，必须要下定决心，必须对同一张“虚拟”的表进行全方位的审视并开发单一的报头和封装结构、容器交换等等。达到单一全球模式处理电子商务信息和组件这个目标的希望时间是在今后几年，但需要不断努力。

## 12.2 在电子商务中应用XML

如同我们已经看到的，XML 提供了一种标记数据的标准语法，并允许我们在消息中加入附加的信息。这样我们能够将脚本同商业规则相联系。这使我们的表达式和消息结构完全可用。为了了解如何使用这些长处以及 XML 如何应用在电子商务中，让我们看看另一个使用 XML 的电子商务消息的例子。

即使你从没有研究过在电子商务中使用 XML，你也应该猜出在客户和供应商之间运转的是 XML。但 XML 中应包括什么呢？我们将在贸易伙伴中传递的 XML 作为消息，虽然这里这个词是独立于信息传输方式的（不要求是电子邮件消息或一个基于队列应用程序中的消息）。

这个部分也许对那些相互发送 XML 文档的人有吸引力。如果你正在填写 HTML 表单，将不会总是发送一个 XML 文档。但是，如有可能在客户端建立 XML 是很有好处的，因为如果你决定在特定的消息下重写一个应用，这样仍然可以使用同样的客户应用程序。理论上，如果以 Web 的形式收到一定单，需要将定单信息传递给填写它的人，这样可以在流程的每一步使用 XML，从开始到定单的填写完成。

下面是一个消息的例子，可以在后面的章节中参考它。它来自 Microsoft 的 BizTalk 模式。在本章结束时将见到 BizTalk 模式。它是一套如何在 XML 中发布模式，以及如何在应用集成和电子商务中使用 XML 消息的指导标准。它的目标是加速 XML 在电子商务中的采用速度。

注意 < BizTalk > 元素有两个不同的元素：<Route> 元素包含报头信息，<Body> 元素包含要发送的消息。

程序清单 12-1

```
<?xml version="1.0"?>
<BizTalk xmlns:="urn:schemas-biztalk.org: BizTalk/biztalk-0.8xml">
  <Route>
    <From Location ID="value" LocationType="value" Process="value"
      Path="value" Handle="value" />
    <To Location ID="value" LocationType="value" Address="value"
      Path="value" Handle="value" />
  </Route>
  <Body xmlns:="urn:your-namespace-goes-here">
    <MessageType>
      -- Your XML document data goes here --
    </MessageType>
  </Body>
</BizTalk>
```

这个例子展示了XML消息的一些要点，从流控制、调试和审计点。强烈建议应在XML消息中包含报头信息作为附加数据。这两个部分通常看作报头和报体，下面我们将分别对其进行介绍。

使用XML或EDI发送数据的关键区别在于EDI使用的是紧耦合的消息，而XML使用的是一种松耦合的结构，它具有更好的弹性并易于扩展。

### 1. 消息报头

报头信息——有时叫做信封——包括执行交换要求的附加信息。并没有一套规定在报头中必须包括什么的信息或定义，但它通常会包括：

- 路由选择。
- 安全性。
- 批处理。
- 错误处理标志。
- 交易定义。
- 其他要求的信息，如同消息关联的路径信息。

使用一套标准方法在XML服务器之间交换和路由交易的能力是全球电子商务的关键。

在许多应用中，这个报头信息并不总是作为XML保存，例如，当在线订购CD时，将从浏览器中获得一个HTML表格，将被发送的订购信息的报头保存在HTTP的报头中。但是如果你使用的系统产生了一个包含这些信息的XML文档，则需要在文档中包含它从哪里来到哪里去的信息。根据应用程序的不同，这个可能需要包含在XML文档中，这样接收的应用才可以处理它。

让我们看看这种类型报头信息的例子。ICE（信息和内容交换）是XML在内容联合领域的一个应用，它是一种正在成长的商务形式。在报头中使用了大量信息表明承载的是多媒体、目录还是其他应用。图12-9显示了消息的报头元素。注意：ICE采用的方法是使用单一的DTD覆盖所有的订阅者往来的交换。

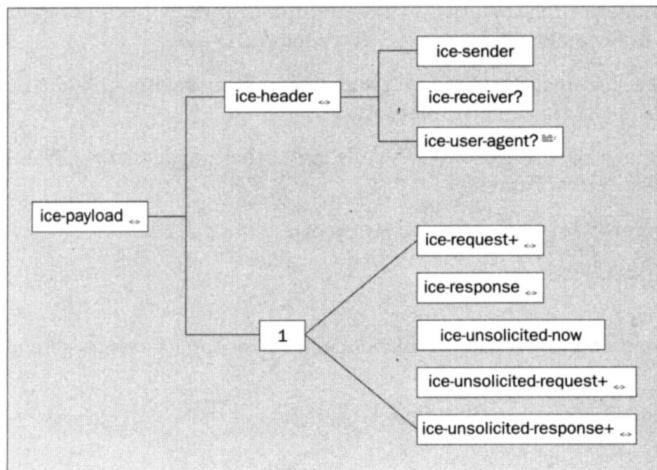


图 12-9

下面是DTD报头片段：

```
<!ELEMENT ice-header (ice-sender , ice-receiver? , ice-user-agent? )>
<!ELEMENT ice-sender EMPTY>
<!ELEMENT ice-receiver EMPTY>
<!ELEMENT ice-user-agent (#PCDATA )>
```

发送方的属性：name, role, sender-id

接收方的属性：name, receiver-id

这表明了一个简单的机制能够非常有用，并提供了一个可以应用在许多电子商务领域的一般解决方案。

#### (1) 多层路由

消息的一个重要的表现是在同贸易伙伴会话时如何引用本地信息或资源。本地信息并不保存在消息中，但是处理应用要求使用它。如同我们在本章前面所看到的，在电子商务中我们处理的是商务会话。会话线程和所有的会话信息要求一个机制保持相关信息。例如，当从贸易伙伴处收到一张发票，我们需要将其同购买定单、贸易伙伴的交易文件等相关联。我们早就注意到，这是XML表达式利于应用开发商的一个方面。有时当贸易伙伴需要将消息进行进一步处理时要求添加信息。目前，处理本地信息没有标准的方案，而其中可以获取巨大的价值，但标志这些资源的通常方法必须注意于实现下列目标：

- 支持链接和嵌套的工作流。
- 允许记录在交易的生命周期中一直存在的关节点（键）。
- 创建的交易应能够在组织中和/或在贸易伙伴（通过Internet、EDI）中存在。
- 允许保持独立的系统关键值，用于记录和访问以层次（XML）或关系形式存储的信息。
- 允许媒介、中间商、VAN和其他第三方准确、简单、快捷地传递消息。
- 具有标准地定义记录访问的语言。
- 传输独立。

图12-10展示了当电子商务通过一个管道时一直存在的信息。

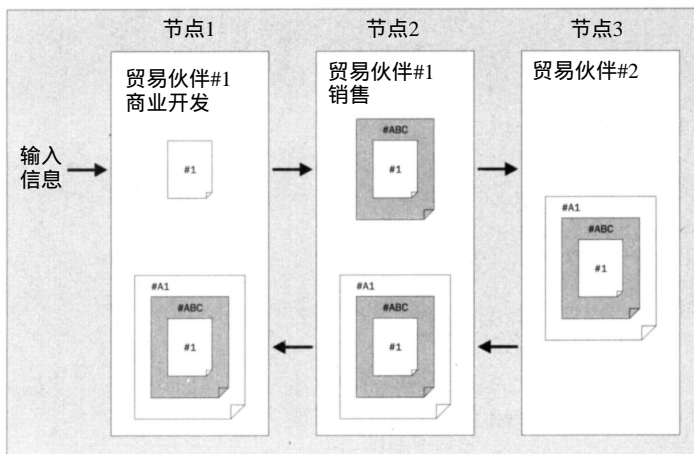


图 12-10

在这个图中，我们可以发现，当信息进入系统并进入第一个贸易伙伴的商务处理部分，在消息的报头中添加了新的元素（#1所示）。这个元素标识这个消息并存放在第一个节点中（当消息在以后某个时候返回或是有一个消息队列。它还可以当作处理过程在该节点的一个入口）。消息——包括附加信息——被发送到节点2，将会在报头加上更多的信息（#ABC所示）。接下来消息到达位于节点3的贸易伙伴，将在报头加上第三套元素。现在每个节点都加上了一些关键信息，这样无论消息在何处存储都可以标记消息，或当消息到达这个节点时执行一个进程。

下面说明在这个领域开始工作时一个可能的起始位置，虽然有许多工作，但还是尽量从各个角度予以说明。

无论是本地信息或资源，附加信息可以是任何元素的逻辑集合。在下面的例子中，在资源的生命周期中我们在每个节点（或贸易伙伴）加上 Echo 元素，新的元素可能会加入到节点的处理进程。这些入口有回调能力——每个元素有一个标识符表明哪个节点和进程加入到入口中。Echo 元素仅仅在逻辑资源等级是可选的（例如对每个消息）并同其他元素无关。该元素并不用来进行数据交换而是使特定的节点拥有进入一直存在的存储格式的入口机制——这意味着这些元素仅同拥有者相关，而同其他节点或贸易伙伴无关（注意每个节点可以使用不同的元素名称）。Echo 元素标识了资源的拥有者、源进程的版本号、本地资源和键值。附加的元素可以作为 Echo 元素的从属资源，使用 EchoItem 同 Echo 元素相关联。如果节点本身需要，EchoItem 的结构是以树型加入到资源中。这里我们可以看到怎样使用 EchoItem 和一个 Echo 元素，这样每个资源可以根据它的 Echo 元素看到不同的处理进程。我们将在下面的例子中说明它：

程序清单 12-2

```
<Resource biz='placeholder'>placeholder</Resource>
<Echo>placeholder</Echo>
<EchoItems>
  <Item>placeholder</Item>
  ...
  <EchoItems>
    <Item>placeholder</Item>
    ...
  </EchoItems>
</EchoItems>
...
```

Echo 元素将拥有者的 ID 作为元素内容。开放的 ID 元素包括的内容参见表 12-2。

表 12-2

属 性	全 名	含 义
q	qualifier	同 X12 类似的保留值，显示指定的授权 ID
t	token	包含一串信息用于节点进程，它是独立于拥有者的。这个令牌包含一个字符串还可以包含索引号等，可以是一个组织任何感兴趣的格式，用于选择一张表（树）和记录（子树）
c	configuration	显示进程的版本号和保存令牌中的存储信息，通常是一个时间戳。确认信息可以在进程和存储升级时用来表示过去的令牌

## (2) 路由的例子

通过使用以上方法，我们看到 Echo元素是如何被表示的。下面的部分包含 5个Echo元素用于一个资源。第一个是可以指定的最小值。描述 ( q="01" ) 显示 109872721是一个DUNS数 ( 一个在欧洲之外被Dunn和Bradstreet分配的数 )，拥有者将参照键值 ( ABC353 ) 作为令牌。

```
<Resource.2 biz='43432432432'>
  <Echo q='01' t='ABC353'>109872721</Echo>
```

在处理这个资源时，将会有以下步骤：

- 系统检查是否有一个或多个 Echo元素其内容同它的描述值匹配。
- 是否不止一个值，查找 q属性值找到同公司/进程匹配的 Echo元素。
- 使用 t属性值从长期存储设备中获得需要处理的记录。

下面的例子演示的是节点拥有者将 c属性作为一个时间戳或令牌时的情况。另外， User项用于定义同节点相关的特殊信息，它用在组织内部，为 109872721。拥有者同时维护一个令牌锚 ( anchor token )，通常用于引用会话双方的原始消息。

程序清单 12-3

```
<Echo q='01' c='19991126:3443' t='345434'>109872721</Echo>
<EchoItems>
  <Item value='JonesGeorge@mycompany.com'>User</Item>
  <Item value='333444'>Anchor</Item>
</EchoItems>
```

第3、第4和第5个Echo元素的例子提供了其他 t属性的可能值。你可以看到，它显示了部分数据资源队列，因为它是用于恢复原始记录的属性。其内容主要由节点拥有者指定：

程序清单 12-4

```
<Echo q='01' t='WHERE
  <CustProfile><UniqueID>ABCD123456789</UniqueID>
  <CustProfile>'>
  139878721
</Echo>

<Echo q='01' t='WHERE <CustProfile><GivenName>George</GivenName>
  <Phone>223333222</Phone><CustProfile>'>
  6134478721
</Echo>

<Echo q='01' t='SELECT KEY(Record) FROM Travel_Header ch, Cust_Body csp
  WHERE csp.Approver='EDI' AND
  CONTAINS (Customer, "George Smith WITHIN Contact") > 0'>
  4137778721
</Echo>
...
```

这里你可以看到能够向消息处理程序添加入口的信息类型。看完报头，下面我们看看消息的主体部分：



## 2. 消息体

消息体中存放的是到达消息头指定的目的地时消息的主要内容，它可以是一个定单、一个确认函、一些重要的信息或需要共享的销售数据等。消息体可以使用任何数量的现存 XML 词汇进行书写，也可以是为了特殊的任务而写入的自主的内容，你可以在第 4 章中获得如何模仿的信息。

随着可以利用的 XML 行业标准的不断增长，可能已经出现了适合你的需求的标准，或有同你的需求相似的标准，这样可以以此为基础在其上加入自己的模式。但是如何找到适合自己需求的那个标准呢？应从你的贸易伙伴开始。也许他们可以向你提供一个模式或建议你到一个 Web 站点进行了解。他们的资源可能是参考了一个 XML 容器（存有许多 DTD 和其他组件模式、代码列表和脚本的站点），本章的后面将进行介绍。

### (1) 使用现存的模式

记住定义一套适合 EDI 交易的词汇表是实现 EDI 系统时一项昂贵的任务。如果在你的问题领域中已经存在一个词汇表，你应该对其进行仔细研究。这样节省的不仅是金钱，更重要的是可以节省书写自己的 XML 词汇表，开发这个词汇表的人已经花费了大量的精力研究什么信息需要在交易的双方进行传输。你可以考虑直接采用这个行业标准或以此为基础。如果你仅仅准备在现存的标准上建立自己的词汇表，应考虑是否复制它的功能或是否应该使用部分同你的命名空间相似的标准部分（可以在第 7 章获得命名空间和模式的概念）。

在我们开始使用模式前，记住我们早先遇到的错误概念：

如果一个团体创建了一个标准定义允许我们所有人去读同样的页面，那么我们的系统是可以互操作的。

我们说，EDI 证明了情况并不是这样。EDI 标准定义了交易的结构，企业并为此达成了协议——通常是一个商业文档。该标准对交换的描述限制很少。为了在这个标准上进行贸易，必须基于贸易伙伴的关系制定附加的特定用于交换的协议，定义使用的元素的子集和使用细节。

### (2) 注册器和容器

有一些存有特定行业信息的注册器和容器供用户共享，它们在覆盖信息的宽度和数量上各有不同。注册器仅仅包括 DTD 和模式，而容器包含更多的信息，如数据库模式、软件代码或规则以及其他管理商务的对象——容器像交易一样对商业对象进行索引，而注册器仅仅存储在交易级别要求的条目。注册器和容器都向组织、特殊的小企业提供信息以帮助它们开发系统或扩展交换信息的能力，通过分享在该行业的经验和代码以实现快速开发。下面是两个目前主要的注册器：

- BizTalk.org——由 Microsoft 发起，由 XML 标准用户团体支持。本章将对其进行详细介绍（<http://www.biztalk.org>）。
- OASIS——由国际财团资助的非赢利组织——高级结构化信息标准组织发起，目的是推广使用基于公共标准的独立于产品的信息格式（XML）（<http://www.oasis-open.org/html/rpublic.htm>）。

由于组织很少只在一个行业领域进行商务活动，大多都要使用在容器中的不同的信息集合。例如，一个生产汽车部件的公司可能选择访问 BizTalk.org 以获得关于 AIAG 的信息，用于同汽车

制造厂做生意；同时获得 VICS 信息用于向汽车部件商店进行销售；获得防卫服务代理以获得国防部的定单。

注册器允许参考的不仅仅是 DTD 和模式，还有逻辑商业片段，以供公司的解决方案进行动态参考。请参阅未来展望中的容器部分。

### (3) 国际化

目前很少有协议做到了全球化。对于全球的电子商务，系统的设计必须有足够的灵活性以适应不同区域的差异。商务活动必须了解在每个国家的商家和消费者所面临的规则、法律、收入、隐私、税收线等。

现在不再讲 XML 可以实现 EDI 了。下面看看对电子商务中如何使用 XML 问题的一些感性认识，我们将在了解它如此重要的原因以前，从一些错误的概念开始本节。

#### 12.2.1 通常的错误概念

对于 XML 在电子商务中的角色有一些错误的概念，例如许多人认为：

“相当简单，通过使用 XML，我们可以创建相当易懂的语言，可以在浏览器中显示一个简单的样式表，可以通过 Internet 进行交互，并可以使用相对廉价的互操作软件。这就是 XML 对于电子商务如此重要的简单答案。”

但是，他们忘了目前我们只有 HTML 和 Web，它们已经改变了电子商务。的确 XML 可帮助 Web 开发商实现后端同 Internet 电子商务网站的协同工作，但许多前端是静态的，并仍然是 HTML（直到广泛使用支持 XML 的浏览器）。另一个错误在于：

“无论你想交换什么信息，一旦你的伙伴理解了你使用的标记，就可以简单容易地实现交互。”

这些人需要扪心自问他们的建议和目前的 EDI 标准有何区别。你必须理解在电子商务中使用 XML 的原因并不只在上面的两句话中。

让我们看看一些在书中和其他教材中曾说过上百次的概念。它们对使用 XML 在贸易伙伴中进行通信的理解是不正确的。

##### 1. 如果我们就 DTD/模式达成协议，那么我们就都可以交换文档

许多 XML 开发人员都有一个错误的概念，即如果我们就一个 DTD/模式达成协议，那么就可以实现交换。它是对下面这个说法的概括：一个团体创建了一个标准的定义允许我们读取同一个页面，那么我们的系统是互操作的。然而，EDI 证明这是行不通的。EDI 标准已经提供了定义好的行业协议认可的交易集合——通常是一个商业文档。但是在现实世界中，即使有了这些标准，也会基于贸易伙伴的关系出现附加的协议从而扩展这个标准使其不再成为标准。

标准需要被扩展有现实的商业原因。为了在竞争对手中获得领先地位，方案提供商需要不断寻找差异以使他们的产品或服务同竞争对手的有所不同。由于这个原因，我们的消息需要能够处理新的包，或转换不标准的方法。

但现在如何扩展一个 DTD 呢？有下面几种可选的方案：

##### 1) 对每个贸易伙伴复制并修改标准（非常坏的选择，将会导致过多的 DTD）

2) 使用再循环 (recursive) 树。如图12-11所示,再循环元素显示为 ice-item-group、ice-item 和 ice-item-ref, 允许ICE扩展并携带贸易伙伴之间定义的包。为了扩展这个包, 一个唯一的参照 id 标识该项, 并允许该树调用自己, 完整的分支能够在选定的点加入并在请求时有效, 这同 DTD 相似。这样贸易伙伴可以使用扩展的消息交换信息。

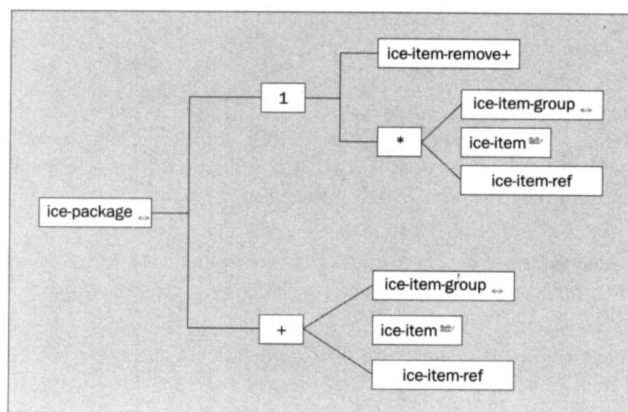


图 12-11

更多的关于ICE的信息可以参阅：<http://www.gca.org/ice/default.htm>

3) 使用DTD接口机制。标准的 DTD 包括一个调用, 使用参数输入到一个定位在接收方服务器或容器中的 DTD 段。如果要求全部使用标准, 那么接口包括一个空的 DTD。如果由于某种原因要求对标准进行扩展, 在指定的接口位置要有元素。

制定 XML 模式 (在第 7 章有述) 的 W3C 承诺为了允许这种对标准模式的扩充, 将定义一个开放的模块。

注意在供应商实现的例子中, 经常会出现打破消息和机制的探索。如果一个公司修复了一个问题, 在服务客户时出现了产品差异, 这在某些情况下会减少竞争。这给标准的参加方一个边界, 一些聪明的供应商会为了个人利益而收集这些标准。这样标准被设计为了一个特定的供应商工作, 而不是所有的商品和行业。这种行为必须被监控。但是供应商的参与还是有好处的, 他们可以带来知识、投资和市场营销组件。

## 2. 我们可以基于 DOM 建立进程

许多 XML 开发商在开始 XML 编程时就学到了 DOM, 并将其看成是在电子商务方案中处理 XML 文档一个简单、理想的方法。但是电子商务的消息可能有几兆字节, 对一个组织来说, 每月收到超过 50GB 的信息并不罕见。当然 DOM 对小的消息很好, 但在处理大的消息时会出现问题。

因为 DOM 使用一个“在内存中”的方法处理 XML 文档, 它将在使用大的文件时消耗过多的系统资源, 而只是用来处理小的消息。当然 DOM 实际使用的内存数量依赖于解析器, 但在多数情况下一个 100KB 的文档将消耗 1MB 的内存。

接收大的文档的较好的方法是使用 SAX 的事件处理进程, SAX 是我们曾经在第 6 章介绍过的

XML的简单API，它不会引起应用程序的内存管理问题。许多 EDI应用中存在的“在内存中”问题迫使公司重写它们的引擎。这不是说不使用 DOM，而是说应仔细考虑你的方案。

如果你的SAX应用创建了它自己的在内存中表示文档，它可能占据你允许解析器占用的全部空间。

### 3. EDI格式人们很难读懂，但XML却易于使用

现今的电子商务系统，以及将来的某些部分将不再向用户显示陌生的标签，而只需要能够被系统开发人员理解。复杂的标签机制减慢了自动化处理能力。事实上，许多信息对系统来说是作为键进行传输并访问本地数据。我们已经注意到，使用 XML的力量在于它允许使用附加的信息，而标签提供了可以访问的进程入口，这样强化了商业规则。

### 4. 我们的基本目标是从内容中删除处理

在使用XML时，我们经常被鼓励在XML文件中只包含内容。例如显示XML的样式规则应保存在一张独立的样式表中。这是一个伟大的目标，但事实是在电子商务交易中的消息必须包括原始的指令，如增加、改变、删除等等同时还有内容。使用标签（如 `<noun.verb>`）标记是为了处理这些更容易。只有同时理解对象和行动，应用程序的实例才可以正确处理。

### 5. 我们可以通过使用有效的解析器使文档有效

确保一个系统完整的一个致命因素是到达的消息能够被正确处理，任何来自贸易伙伴的消息在进入系统前必须经过完全的检查。这才是有效的概念。下面是实现的三个步骤：

1) 消息必须被标记以遵从适合的DTD或模式。

2) 元素的值和字符数据可能需要进行有效性检查和 /或商业规则检查。这同公司的商业规则有关（如在ACME公司就不接收低于1000美元或高于10000美元的定单）。

3) 可能会有附加的处理将XML/EDI格式转换成老的EDI格式（或其他内部信息格式），这个过程可能需要更多的有效性检查。

XML文档的有效性对于获得的数据适合应用程序的要求是十分重要的。文档遵从 DTD或模式用于保证词汇表和语法是正确的，并保证其中是否包含要求的元素。但是，在 EDI系统中，确实会出现这种问题。

在语法/结构有效性和应用的有效性之间进行区分是十分重要的。应用的有效性保证了商务规则必须在应用级执行，而只有在这个等级才能完全理解商务目标。

应用的有效性允许系统的开发人员强化商业规则，处理与不同的贸易伙伴之间的协议。所有进入电子商务系统的任何消息必须经过仔细的检查，防止系统不能处理的值进入系统引起系统崩溃。

由于这个严格的要求，电子商务的完全有效性不能在一个标准的解析器中实现。即使使用有希望的XML模式，它允许我们处理限制，公司还是应该在特殊的解析器中加入规则处理和安全机制以处理这方面的要求。一种转换应用有效性任务的方法是，可以用一个“侦听 /发布”应用程序。它的任务是检查消息的有效性，并在有效时传递给应用程序。

上述的这些错误的概念你也许听过不少次了，它们可能会在不经意时被谈起。讨论完这些错误概念，让我们把注意力转到使用XML的好处上来。

### 12.2.2 在电子商务中使用XML的感受

当我们将电子商务中应用 XML 的经历写入历史书时，其内容可能会被看成是对过去的重复。“那些忘记历史的人注定会重复历史”这条谚语在我们将 EDI 转变成 XML 时是有教育意义的。现在，你应该看到 XML 在电子商务中的角色同 EDI 是很相似的，虽然我们认为 XML 有两个关键优势：

- XML 语法对我们的应用提供了有力的控制。
- XML 的词汇表可以节省在内外交换时的大量表示工作。

还有哪些是由于在电子商务世界使用 XML 而造成的不同呢？为了回答这个问题我们应该看看许多公司不应用 EDI 系统的原因。下面列出的是前 5 个使用 EDI 系统的障碍：

- 系统成本。
- 缺乏标准。
- 缺乏培训。
- 公司态度。
- 安全。

让我们具体看看这些问题。

#### 1. 系统成本

我们曾经说过 EDI 系统的实现是很昂贵的，但在这巨大的成本中，处理部分同 XML 系统是相同的。开发 EDI 系统时许多花费的产生是由于问题的难以解决：首先定义消息标准花费了大量的时间和投资。另外，创建内部数据和每个贸易伙伴之间的对照表也是十分艰巨的。事实上，人力的培训费和开发工具的培训费用在一个组织完成一个 EDI 系统时也是一块较大的花销。但是需要注意的是，仅仅将消息的表达式转换成 XML 格式并不会使开发过程便宜，传统的 EDI 标准可以扩展以包括所有 XML 的机制。那么使用 XML/EDI 节省的费用表现在哪里呢？答案受到下面观念的影响：

“如果我们销售亏本，让我们扩大销售量。”

XML/EDI 向企业打开了合适的市场，开发成本的投资回报（ROI）是一个企业的基础设施好坏的表现。所以认为 EDI 市场太小、人力费用太高的主要软件公司现在都发现了扩大销售和服务量的机会。XML 同产品的合作是 XML 在电子商务中发挥作用的原因。

从软件工具的视点换个角度看看，看看现在销售的后台系统、数据库、消息处理程序等等并没有一个组件定位在互操作性上。

用户/客户期望互操作性。XML 提供了互操作性，并且供应商知道通过使用 XML 杠杆，它将在同行业中获得竞争优势。

#### 2. 缺乏标准

虽然行业创建 DTD 和其他用于 XML 交换的模式需要时间，由于对互操作系统需求的增长以及在商务系统中使用 XML 需求的增长，许多公司都创建了能够用于交换的模式。另外，还有一些行业组织在创建模式的容器，不久我们就会见到。实际上微软的 BizTalk 策略包含着帮助公司



为了XML交换定义合适的模式。

### 3. 缺乏培训

随着XML在组织和公司中被广泛采用，我们的人力资源可以有灵活性地使用，语言和工具的培训投资可以被扣除。不久以后，我们会发现在高等学校中开始教授 XML和HTML。这些因素减少了公司风险并保护了投资，那么公司为什么不转到 XML/EDI上来呢？

### 4. 公司态度

当EDI仅仅在大的公司和政府机构获得很小的一部分市场份额时，小的公司正在更加关注Web的力量，并走向Internet。公司在Web上占有一席之地的压力不断增长，雪球越滚越大，鼓励越来越多的人开始关注潜在的市场。

### 5. 安全

随着内制支持XML的软件不断涌现，实现客户安全的需求不断减少。另外，还有一些特定的开发用于保证完整性、消息认证和/或签名授权服务，例如联合的XML签名机制被W3C和IETF执行（获得更多关于XML签名的机制，请参阅<http://www.w3.org/Signature>）。

我们看到，EDI已经提供了电子交易要求的功能，这在XML中常常被忽视。XML进入各个商业部门的速度令人震惊，它被接受的速度EDI从没有过。我们看见XML在广泛的领域中用来表示数据，从DNA序列到宇航数据。它为XML/EDI带来了重要的意义。随着XML在更多商业领域的应用，我们将快速地变为面向更通用的知识基础。简而言之，XML对电子商务产生如此影响的原因是：

- XML语法对我们的应用程序提供良好的控制。
- XML的词汇表可以节省使用不同的格式表示内部的数据。
- 应用软件内置XML的支持，这样在不同产品之间有了互操作性。
- 在内部使用XML的组织不断增加。
- 不再需要在内部数据表现格式和交换标准之间进行转化。
- 掌握XML显示技术的程序员不断增长。
- 人们对电子商务的兴趣和理解不断提高——带来XML和电子商务的高潮。

上述这些毫无疑问是电子商务采用XML的主要原因，XML并将在将来带来更大的利益，同时我们应该清楚XML和EDI两者的结合会超过任一部分。这正是我们要说的，因为它打开了通向电子商务之门，而并不是70年代出现XML的余热。

说到这里，我们可以从EDI的努力中获得许多教训。不采用它们的经验是愚蠢的。从实现EDI过程中我们获得的东西包括一个具有丰富商务特征的语言、商务规则贸易伙伴管理、知识以及交易经验，这些在使用XML实现电子商务时都可以使用。

## 12.3 展望未来

了解了使用XML和EDI的关键基础之后，下面我们看看其开发策略，它将有利于用户和开发人员获得更简化和更具灵活性的电子商务系统：

- 简单标记语言。
- 发现和调解。

- 容器。
- Bizcodes。
- 代理。
- 模板。

#### 简单标记语言

HTML 语言的成功来自其广泛的可接受性和易用性，但 XML 语言的流行部分是因为对 HTML 语言的限制。所以，面对向 XML 中增加特征的诱惑和加强其灵活性的标准，我们应注意不要使其太复杂。其后的原因很简单：复杂性提高了供应商进入电子商务的门槛，实现对获得接受和流行是很重要的；另一个重要的原因是通常的用户希望获得同 HTML 相似的易用性和一致性。

一些投资电子商务的公司已经开始游说不同的政治团体和标准组织以保证 XML 标准族尽可能地保持简单、紧凑，而不用为将来考虑。一个这样的电子商务供应商在 1999 年 11 月的 W3C XML 模式工作组会议上就这样做了，在一份提纲上力图保持草案的简单。它们提供了傻瓜规则：“如有问题则避之”。这在产品提供商中获得了广泛响应。供应商们知道，当方案更小、更轻、更快、更便宜、更容易以及更可接受，则越易成功，其最终目的是希望保护它们在 XML 技术上的投资。

为什么 XML 的将来并不确定呢？当前 XML 相关的主体定义了以下需求（1）屋内系统，如同文档出版、CD 发布、信息管理等；（2）电子商务的外在系统。要求的这两个方面很奇怪，广泛地讲，屋内系统开发商希望更高的灵活性和在标准上的扩展性，但那些考虑互操作系统的人希望标准简单。这导致推向一个十分简化的标准：对 XML 的更简单的变体叫做简单标记语言（SML）。

由于在复杂性和全球可接受性之间存在着强烈的冲突，毫无疑问我们将生活在 XML 和 SML 的世界中。对于将来每一个加入 XML 家族的建议，都将增加应用程序的处理功能和影响范围。所以 XML 可以被定义成一个值得选择的、简单的、清晰的 XML 的子集，虽然其有可能被增加附加的交易机制。许多人将这看成是一个不幸，但它却可能会成为为了实现计算机同计算机交换的开发人员的唯一方向。

#### 12.3.1 发现和调解

通常，用于现代电子商务的交换机制先于处理的发展，已经可以访问通常的信息如代码列表、条件等等。但是在许多情况下，最终用户或程序员需要从一个站点发现可利用资源的顺序以及一个贸易伙伴自己使用的词汇表。这会产生具有一定广度的新的特征，从能够同新的贸易伙伴的自动化交换，到创建可以为我们自动比较类似产品和服务价格的应用程序。

为了定义“行业标准”消息，我们需要定义标准的模块用于交换这种具有基础设施类型的信息。我们需要发现关于使用的语言和从贸易伙伴处获得的可利用的产品和服务信息，这就是发现和调解的重要之处。在该领域的工作还不成熟，对它的发展有许多指导性意见。

##### （1）正确地发现资源实例

通过发现机制，一个进程（应用程序、贸易伙伴等等）能够发现——从贸易伙伴那里——贸易伙伴支持的词汇表的格式和版本。事实上，作为发现进程的一个部分，当一个客户询问关于一个贸易伙伴所讲语言的信息时，贸易伙伴可以提供上面写着相近词汇表的模板，这样客户仅仅需要提供它们需要查询的详细情况。

例如，如果你希望查询同一个贸易伙伴的合同细节，贸易伙伴应向你发送下面的模板，上面有它们查询时需要的你提供的信息并在查询结束后向你提供结果：

程序清单 12-5

```
WHERE
  <CustProfile>
    <GivenName>$a</GivenName>
    <Contact><Phone>$b</Phone></Contact>
  </CustProfile>
IN "www.orgname.com/custprofile.xml",
$y >
```

这里，贸易伙伴仅仅需要让客户知道 \$a和\$b代表什么，这样客户能够填写细节并执行查询后返回结果。

看完了电子商务领域支持的功能，当出现问题时我们该如何处理呢？发现同样应该通知贸易伙伴可能的偶然策略和资源，但依赖于当时的环境和条件。进入这个领域较早的是CommerceNet的eCo模式。

## (2) eCo模式

eCo模式是一个定位于发现过程的推荐的蓝本，由 CommerceNet财团发布。在一本白皮书中它们写到：

“eCo工作组决定为了促进Internet上异构的电子商务系统B2B的互操作性，贸易伙伴应能达到以下要求：

- 在Internet上发现其他的商务活动。
- 决定是否进行电子商务以及如何能够进入市场。
- 决定提供哪种服务。
- 决定它们的电子商务系统是否及怎样通信。
- 如果可能，决定需要进行哪些修订以保证它们系统之间的互操作性。
- 如果可能，建立不同于Internet的通信渠道。

改善发现过程的眼光和努力是前进的第一步，并提供了一个可以建立的良好模式。为了允许贸易伙伴、Web用户或搜索引擎决定从网站获取什么，在eCo界面中定义了一个叫做bootstrapping的方法。bootstrapping要求在学习更复杂的机制时仅仅使用最少的系统资源。eCo模式调用eco.xml文档用于询问和发现站点支持的界面和资源。该体系结构同样提供探索同eCo兼容的现存的电子商务系统的机制。为了在eCo方案中支持现存的电子商务系统，存在一个机制用于在这些系统的界面上描述和提供信息。

参阅eCo方案请浏览：<http://eco.commerce.net/specs/index.cfm>。

### (3) eCo模式的例子

如果一个实现了 eCo 商务环境的公司希望自己作为一个 eCo 兼容的公司被检索，它应该在它的 Web 网站的根目录下放置下面的文件。下面的例子是一个提供价目服务的公司的文件：

程序清单 12-6

```
<?xml version="1.0"?>
<EcoInterfaces xmlns = 'http://www.commerce.net/eco'>

  <Head>
    <Identifier>http://www.my_company.com/...</Identifier>
    <Creator>http://www.my_company.com/~BillSmith</Creator>
    <Date>19991118</Date>
    <Version>1.0</Version>
    <TimeToLive>86000</TimeToLive>
    <Description>...</Description>
    <Label>...</Label>
  </Head>

  <Interface type="Business">
    <Identifier>http://www.my_company.com.com/business</Identifier>
    <Creator>http://www.my_company.com.com/~WillSmith </Creator>
    <Version>1.0</Version>
    <Date>19991118</Date>
    <TimeToLive>86000</TimeToLive>
    <Description>Catalog Business</Description>
    <Label>My Company</Label>
  </Interface>

  <Interface type="Service">
    <Identifier>http://www.my_company.com.com/eco/OrderService</Identifier>
    <Creator>http://www.my_company.com.com/~JillSmith</Creator>
    <Version>1.0</Version>
    <Date>19991118</Date>
    <TimeToLive>86000</TimeToLive>
    <Description>My order service interface description.</Description>
    <Label>My Company Order Service</Label>
  </Interface>

  <Interface type="Service">
    <Identifier>
      http://www.my_company.com.com/eco/CatalogService
    </Identifier>
    <Creator>http://www.my_company.com.com/~GillSmith</Creator>
    ...

  </Interface>

</EcoInterfaces>
```

### (4) 比较购物

另一个认为发现过程重要的例子是比较购物。如果我们使用 XML 标记了我们的产品和服务，就有可能创建搜索引擎以比较产品和服务的价格，以及向最终用户提供多种购买选择。就好像这些站点能够比较价格并允许你选择购买产品和服务的地点，然后有希望你能够规定更多的关

于保证、送货时间等的选择。但是，这个概念减少了逛商店的需求以及主动查询电子商务站点的需求。

### 12.3.2 容器

对兼容性的进程和词汇表需求的快速增长减少了“烟囱”和冗余的任务。“烟囱”是指内部以解决特殊问题为目的的独立的应用程序，它同时缺乏灵活性。为了行业、协会和独立企业使用XML进行信息交换的工作，他们需要建立标准的机制，这样可以共享词汇表。

容器是不同的DTD和模式存储的地方，对于十分复杂的领域那里有更多的信息可以利用。信息可能包括基于XML的目录机制、确认管理、主题映射、数据结构、UML建模工具以及更多的东西。也可能包括对类型的关系、代名词以及多义词的介绍。甚至可能包括可以编译后重用的软件。当容器开发后，那里的可利用信息的数量将快速增长，不但容器自身希望自己成为一个重要的角色，竞争的标准也会提高其重要性。

企业具有（1）内部概念，（2）同其他同行共享的概念，（3）长住居民共知的概念，（4）全球性概念是没有意义的。定义类型1，2，3被每个企业概念化成不同的范围，这样我们的XML电子商务系统必须调和这些不同。电子商务系统必须在一个标准的动作中处理这些语言的不同，虽然协商、共享的交叉索引、将使用不同词汇表的消息重新打成使用自己词汇表的消息机制可以完成该功能。

另一个必须做的事是优化商务语言版本。那么在哪个级别可以共享更多的信息呢？是在模式或DTD级别吗？如果是，我们是否共享一个购物定单、使其成为一个一般的元素“厨房下水道”以容纳各种实现？或一个定义了无数购买定单的容器，一个用于销售机票另一个用于销售铅笔？那会有多少个呢？成千上万，但恐怕也不能覆盖所有的情况。上述方案都不理想。那么答案又是什么呢？从历史的角度来看，获得完全的标准会有很高的成本，建议的升级要求同步延迟，并需要程序员注意全新的标准版本。如果在我们的处理中建立合适的确认管理就可以减少这些维护问题——可取的是在逻辑单元等级。

XML的容器提供下列工具：

- 在整个生命周期对系统进行模块化（使用UML）开发和测试时作为指导访问行业组件和术语。
- 扩展获得的元素以映射和处理数据直接到组织的商业系统内部。
- 大量覆盖在贸易伙伴之间的交互，如实时交换。
- 利用下一代应用软件直接同来自贸易伙伴的传输数据进行交互。

所以容器可能最终包括标准化的标签的集合、商业组件（程序代码——包括源代码和编译后的代码）、对象、样式表以及行业条款和代码等内容。

为了浏览一个容器并找到工作中用到的合适的行业信息，容器中有必要对新的功能建立关键词，使用索引（信息树）可以进行基于主题（商业领域）的和要求的商业共享功能的查询。索引是功能性的指导，但一个最终用户可能希望通过直接的域路径查询浏览一个索引以获得特殊的商业功能。这种将主题作为语义的基于域的表现树提供了一个从属的界面，以提供传统的关键词或目录风格的查询。



简单地说，这个概念就是“从你的商业领域列表中选择你想做的，相关的逻辑商业单元将同更多的信息一起显示以提供更好的帮助。”例如，一个用户可能在一个金融投资容器中浏览一棵树的层次，以获得商品目录以及一个购买活动，找到执行这个任务的要求、标准的交易、时间和调度、特别的条件（例如限制和边界请求）以及为此提供的事件。索引同样定义独立的数据元素以及同行业内外其他元素的相互关系。

下面是一些在开发容器时其他初步要求：

- DSML（目录服务标记语言）：管理关于人力、资源和进程的数据，同 LDAP 相似。  
<http://www.dsml.org>。
- 统一数据元素模式（UDEF）：是将不同的数据标准（CALS、STEP、X12 EDI等）映射到一个单一的模式，使用一套目标类和属性以定义数据元素。  
<http://www.undef.com>。
- BSR（基本语义寄存器），是国际标准化组织（ISO）努力的成果，遵从 BSI/Beacon项目。  
<http://www.iso.ch/BSR>。
- ISO/IEC 11179-1（国际标准化组织（ISO）的/数据元素组成专业和标准框架）：包括媒体数据。  
<http://www.sdct.itl.nist.gov/~ftp/l8/other/coalition/Ovr11179.htm>。
- UREP（来自UNISYS的统一容器），是一个基于XML扩展的信息系统，定义、综合以及管理媒体数据和商业数据。  
<http://www.unisys.com/marketplace/urep>。
- XMI（XML媒体数据交换格式）：“定义了一个开放的信息交换模式以使使用对象技术的开发商能通过Internet使用一种标准化的途径交换程序数据。”  
<http://www.omg.org>。
- DII COE XML 寄存器：为了提高使用DOD通信、条目浏览和使用DOD系统的数据互操作性。  
<http://diides.ncr.disa.mil/xmlreg/index.cfm>。

#### Bizcodes

我们都熟悉条形码，它包括对不同条目和产品的描述并提供输入接收设备以获得扫描的内容同后台数据库的连接（参见图 12-12）。使用条形码的应用包括存货计数、资产 /包裹跟踪、日志识别、销售定单输入、计量读取以及公园售票等等。条形码不但可以用在收集数据，还可以用在系统中的价格查询以及其他标准的商业操作。条形码同 EDI在信息收集和共享数据领域始终是兄弟技术。



图 12-12

Bizcodes扩展了条形码，吸取了条形码成功、适用、好用的特征。它向电子商务领域的所有媒体数据扩展进行了扩展，并被XML/EDI（<http://www.xml.com>）小组所建议。Bizcodes可能会成为XML容器的关键，允许对信息相关条目、图像掩码、可能值、缺省值、帮助、定义、连接有效类以及限制产生疑问。Bizcodes对于XML词汇表、方言、模式以及不同的重用标准之间的自动映射，同样也提供了一个简单的机制。

获得更多关于Bizcodes和容器中使用简化的XLink和XPointer建议的信息，参见XML/EDI白皮书<http://www.xmledi.com/repository/>。

同熟悉UPC/EAN的产品和条目条形码类似，向生产商、销售商和零售商提供简单的、语义

上中立的工具进行库存控制，Bizcodes提供了一种中立的描述，用于同电子商务媒体数据和同处理相关联的数据元素。

使用Bizcodes的基本优点如下：

- 在XML/EDI容器中存在交叉引用信息。
- 交易通过使用连接处理得到简化。
- 允许重用逻辑单元。
- 提供用于全球电子商务的国际语言转换。
- 简单地在DTD或XML模式结构类型中的单链路链接机制。
- 对信息粘贴进行管理并允许知识附加扩展。
- 属性和关系能被进一步询问。
- 对于行业、国际标准以及企业词汇表使用相同的机制，在德国、日本和印度之间转化价格表示——一个在全球电子商务中十分重要的属性。
- 允许在交易的生命周期中进行短暂定义。
- 允许商业语言的特性收集。
- 允许容器快速学习商业语言，并服从标准。

### 12.3.3 代理

代理技术，是为用户或其他程序实现特定的目标而运行的一个独立程序。它们提供更高的扩展性和数据交换的可维护性。代理可以使我们的 XML/EDI系统具有自适应能力并能够处理大容量的交换而不需人力的干预。

所以我们需要理解如何将软件代理技术融合到 XML系统之中。我们在这里不是讨论“星球大战”风格的能力，而只是指简单的功能如对老式的信息系统接口。这种老式的信息接口包括在一个地址中只能处理两行的系统或每个定单只能处理固定的内容的系统。

当一个定单包含的信息超过固定的内容时，或在系统中出现3行地址时，代理就可以处理这些文档了。它们可以作为强化商业逻辑的手段，也可以作为处理商务逻辑没有遇见过的状态手段。为了作到这些，媒体数据规则的使用和表示应使用XML（我们将看到更多的实现）。

代理的使用去掉了特殊的硬编码约束的需求——使得在商业逻辑中易于使用。这意味着开发新的XML应用的开发人员应该在XML内容中避免使用硬编码约束，以避免在处理模块中出现这样的问题，而应该开始使用可适应的媒体数据驱动的状态机。

媒体数据可以被用在从基于规则的开发环境（RBDE）到比较发现机制的任何数量的状态之中。例如，如图12-13所示，我们可以使用代理找

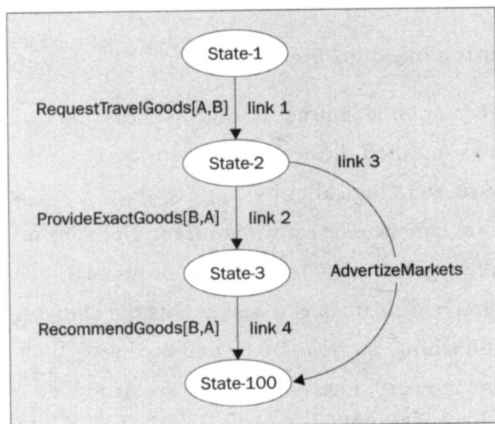


图 12-13

出并比较一个想买商品的细节。在这个例子中，我们寻找的是旅行商品，代理发送对我们要求的旅行商品细节的请求，它指示它要寻找的东西，返回的是满足要求的商品的细节，这些商品相当匹配，其细节应使用适当的标记格式进行描述。代理接下来提供的建议是基于你提供的参数的，如价格、发货时间、保证期等等。

最有可能马上使用的代理工具是跟踪 XML 交易内容以警告支持的员工在信息中潜在的错误状况，特别是相关的版本信息。传统的 EDI 开发人员花了很长的时间才理解应该在贸易伙伴之间进行版本管理。在 XML 驱动领域中，版本必须在概念级以允许语言的升级，我们的代理技术将提供在这个等级处理升级的机制。

#### 商业规则标记语言

不断增加的面向对象程序员将商业逻辑从数据访问和应用逻辑中独立出来。允许同样的商业逻辑被用在不同的应用之中，并允许商业逻辑很容易在其基础上进行升级（从其他应用代码中独立）。IBM 的 CommonRules 1.0 包括开发人员的 API 以强化 Java 或非 Java 应用，允许创建在 Web 上可执行的能够在企业间交换的商业规则——这些企业使用庞杂的规则系统，强化互操作性和冲突处理。通用规则提供了一个通用的 interlingua（国际通信语言）以表示这些交换规则，而使用一个 XML 应用程序叫做商业规则标记语言（BRML）来表示 interlingua。这些规则就可以使用 XML 进行交换，直接作为一个 Java 对象或以其他流的格式。

使用通用规则，一个销售商的 Web 网站/应用能够对它的价格、折扣、退款、撤约策略、定货保持时间以及其他合同上的条款等商业规则同客户的应用/代理进行通信，甚至当供应商的规则通过使用一个不同的规则系统实现，只要该系统被设计成强化现存的基于规则的系统功能。那么当在规则中发现一个冲突时就会出现警告。

规则是定义商业逻辑的有效方法，即当它们使用相对简单的表达式的同时也具有自动执行能力。它甚至允许非程序员在运行时修改商业规则。

更多的信息请参阅 <http://www.research.ibm.com/rules/home.htm>。

### 12.3.4 模板

当我们使用新的词汇表在商业应用中交换数据时，必须考虑谁将做消息转换的工作。许多开发者同意如果接收者要求由客户格式化在词汇表中的消息是相对简单的，而接收程序决定请求什么并将其转换成它的词汇表就不太好了。如果你记住前面发现和调解的例子中的模板，就可以领会模板的能力。如果接收者通知发送者哪里以及怎样发送信息，那么任务就简单多了。

提供你处理交易的语言的细节是处理交易的一个好方法，虽然那些尽力实现能够理解多个词汇表的系统的大公司只能实现它。

模板的思想建立在容器、发现和调解技术、Bizcodes 以及代理之上，这个概念有可能在商务层建立一个通用的讨论协议。模板的思想扩展了我们曾经看到的例子，它们能够被代理动态创建。由于商业交互比较复杂，潜在要求专业知识，所以 EDI/XML 数据操作代理能够确认用户可以在一个较高的层次表达他们的请求，以自然语言提供的模板可能使用同 IBM 通用规则相似的表式。当一个应用开始调用一个交易，电子商务结构组件将自动创建合适的规则模版和 XML 语

法以同用户请求相匹配，要求输入交易信息，并向供应商提交。这些相同的代理将代理整个交易。

## 12.4 理论转化成实现

本章第一部分描述了对未来的展望。我们知道 Internet改变了交换电子信息的方法，我们将不再依靠昂贵的 VAN。虽然当前在电子商务中使用 XML同现存的EDI系统没有本质的不同。我们消除了一些通常围绕着在电子商务中使用 XML的神话，看到如何将XML集成到供应商的产品流程，该领域的现有技术如何使XML这个普通的语言成功地成为电子商务交易的通用语言。我们看到许多基于XML实现电子商务的例子，虽然它们太大以至不能详述，但还是应该说明它的关键优点。

最后需要记住的是，一个在EDI应用上有巨大投资的大公司不会使这个系统无所作为。传统的EDI在过去的几年中会工作得很好，并将一直运行下去。但是，对互操作性兴趣的增加将在这些系统升级时影响它们，或承诺将来在系统中加入这些优点。

在这个部分我们将看到的一些事实，如媒体数据发现和容器已经存在，但毫无疑问他们将具体化。无论代理技术和模板仍是刚刚开始。但是，使用XML使电子商务走入了一个新的方向，它使程序的能力超越了Web，发现你想要的商品、发现供应商使用的语言以及直接从供应商获得定单离我们将不再遥远...

## 12.5 电子商务解决方案

看完了在电子商务中使用XML的理论，我们将看一些行业的具体成果，它们活生生地使用XML实现解决方案。我们将看看已经出现的一些行业 DTD，从中学到某些知识。当然，我们不能讲述每一个实现的行业成果，但是，我们将看一些关键的例子，它将帮助我们看一看其他行业将要模仿的一些重要的特征。特别是我们将看看一些金融、健康以及旅游行业的例子。

另外我们将看到实现一些概念的方法，这些方法在本章中将在本处实现。我们将看到的一个特别的方案是BizTalk。我们将谈谈www.BizTalk.com站点，它是一个框架容器，同时它们也创建XML框架。但是，框架容器只是BizTalk结构的一个部分，它还谈到了产品和服务。不幸的是，开始时对BizTalk结构中的框架容器解释不够清楚，我希望能够解释这些部分是如何相互适应的。

总之，我们将看到：

- 一些为纵向行业创立的标记。
- 对横向行业的处理。
- BizTalk结构。

## 12.6 行业方案

许多已经开发完的电子商务解决方案主要集中在 B2B交易领域。我们将B2B交易分成两种主要类型：

- 纵向市场——公司作为同一处理的各个部分，一个公司需要向另一家公司提供产品和服务，

这样才能完成它们的任务。例如，一个汽车生产公司和一个轮胎生产公司，或一个航空公司和一个行李托运公司或旅行社。

- 横向市场——产生产品和服务不要求相同的产品，如医药生产企业和处方药品的医生。

下面我们看看纵向行业。

### 12.6.1 纵向行业解决方案的关键

当前，供应商、中间商以及客户发现共享和交换信息比较困难。这个困难来自供应商不同的销售渠道。如同许多销售人员和市场专家所知，当前的销售渠道对供应商很难提供不同的服务，到达新的客户群，以及提供创新。渠道定义了供应商和客户之间的相互关系。

同时，客户的能力又不足以联系供应商的服务，选择最适合他们的服务，以及比较不同来源的服务。从他们需求服务的计划和范围，客户必须浏览复杂的相关页面，通常情况下他们不会那样做。

许多专家考虑使用XML和Internet超越传统销售渠道的限制。许多行业已经利用了这个技术，开始采用基于XML的解决方案，这些行业成为电子商务最大的参与者之一。不幸的是，他们仍然采用老的方法实现交易集合，重新做XML表达式的工作而不是扩展它们以使用潜在的新技术，但一旦他们使用了XML，他们探索上一部分所讲的可能性就会更大。

开始时，他们使用自己的方式在电子商务中使用XML，但是有那么多的XML和电子商务标准，我们需要问问自己“使用那匹马来拉我们的车呢？”。许多公司自问下面的问题：

- 我们支持哪种成果？
- 我们注意哪种成果？
- 如果我们支持了错误的结果会有哪些分歧？
- 如果我们等到hoopla消失会有哪些分歧？
- 那个是我们行业的主导方向？
- 我们的商业伙伴（和竞争对手）将走向哪个方向？

XML的一个好处是它允许数据转换。如果使用了“高速路”并且他们的数据足以支持内部和外部的处理，他们同任何指导性的创新结合没有任何问题，只是可能需要将他们的内部结构同行业标准进行映射——转换成可以同他人共享的数据格式。我们在第9章看到了XSLT（虽然XSLT并不总是最好的商业解决方案——我们在案例学习3中进行了分析）。但是，对于当前使用特殊结构的组织的危险在于为了实现同外部实体的协调工作可能不支持内部需求。

在过去的12到18个月中，许多在同一个行业中的公司联合在一起制定语义、消息、内容模式和数据元素需求以用于电子商务。这些组织可以代表他们所在的行业，以及交叉行业和边界行业。但是，如果你看看主要努力的参与者（RosettaNet、BizTalk、ebXML等等），都包括一些同样的大公司。小的和中型的企业（SMEs）不能参与以及作出贡献，这样会处在不利的地位，除非他们使用基于Web的收集工具、Web站点、Internet目录服务器和成员电子邮件。在先前EDI标准存在的地方，财团可以利用这些有益的经验，但是他们也能够扩展一些超越当前EDI贸易伙伴的特性，同时填补一些先前EDI交易所遗留的沟壑。

让我们浏览一下在旅游、人力资源和健康方面的创新，并将其作为纵向行业开发自己的



XML应用的例子。这些将表明不同的信息类型可以通过使用 XML 进行交换，不仅仅是传统的类似使用EDI进行的交易集，还包括文档和其他信息。接下来我们将仔细看看 cXML 的一个特征。这个部分并不想教你怎样为你的行业标记信息，这可以用几本书来描述它（或几卷），但在这些书出版时它们毫无疑问都过时了。但是，它给出了一些创建的例子，以及一些代码，这样你就可以了解那些类型的信息要进行交换以及使用那些表达式。

### 12.6.2 旅游

旅游是世界上最大的行业，它是全球化和互联的。它的特征允许游客创建复杂的旅游路线，围绕定位在成百上千英里范围内的多个目的地。

旅游行业的等级包括航空公司、旅店、汽车租赁公司、全球发布系统以及旅游代理，同时许多商业公司围绕游客提供产品和服务。旅游行业是多样的，正因为此，该行业可以依运输工具分成几个部分（飞机、汽车、火车、轮船），也可以依服务类型（旅店、旅游代理、发布系统）进行划分，甚至可以分为公务或观光旅游。你可能从第 4 章关于数据模块部分记住一些关于旅游公司标记数据的条目，作为划分的一个结果，许多旅游行业部分发展他们自己的唯一标识以及独立的标准、处理以及预约规则。

#### 开放旅游联盟

开放旅游联盟 OTA 定义了旅游行业的未来，定义了一个高成本效益的旅游相关信息通信策略（参见图 12-14）。

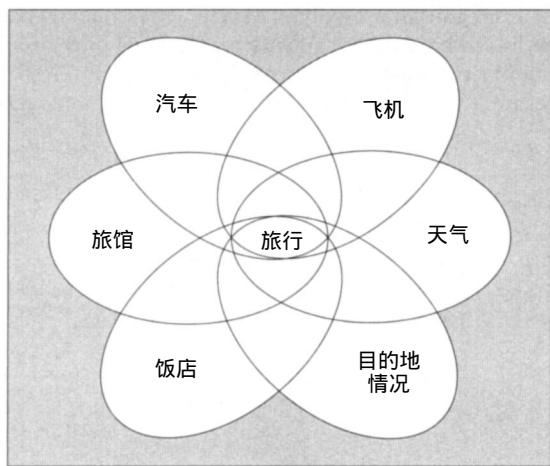


图 12-14

开放旅游联盟尝试去产生一个标准，探索 Internet 带来的低成本、快速通信基础设施提供的广泛访问能力。一旦实现，这个标准将便于在所有的行业参与者之间交换以旅游为核心的信息，无论他们如何联系。

OTA 创建了一个旅行为核心的行业定义。这些消息仅在传统的划分中传递，但却为旅行者提供了一个有用的、广泛的、高成本效益的解决方案。

为了保证联盟的目标可以有效实现，他们遵从数据交换标准联盟（DISA）的合作服务程序。这个程序为所有等级提供服务以支持行业组织建立其关注的行业词汇表和消息。在合作关系上，两个组织已经达成了许多关于 XML 模块定位、详细的交换机制等方面的共识。在 2000 年早些时候计划提出演示版本，同年将加入更多的消息。

更多关于 OTA 的信息请参阅：<http://www.opentravel.com/>。

上面展示了纵向行业如何能够聚在一起相互合作，以制定允许最大行业之一的不同的参与者能够实现相互通信。这不仅有利于帮助旅游供应商进行销售，同时也有利于帮助小的供应商将完整的包放在一起（以同大的旅游供应商竞争），并使客户能够十分容易地在线订购这个旅程。

## 12.7 人力资源

有许多正在开发基于 XML 的人力资源管理系统，以对全球范围内的成千上万的员工的简历、时间和情况进行管理。创建的工作流程需要得到管理人员的指导。工作流程意味着不同等级的人员应具有不同等级的数据访问权利。同旅游行业不同的是，人力资源没有一个一致性的通用方法来定义 XML 如何使用或管理，换句话说，存在着多个标准（见图 12-15）。

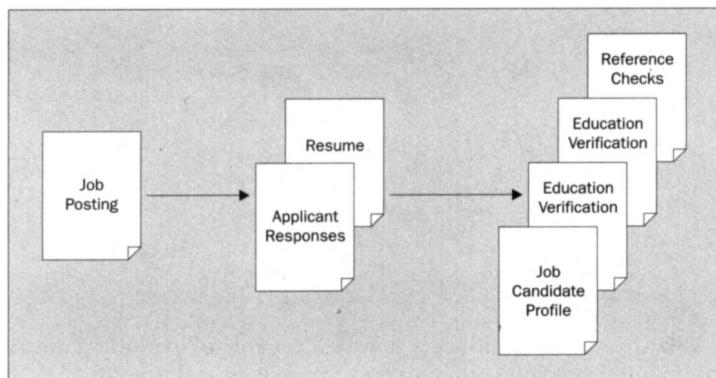


图 12-15

在这个部分，我们将看看一些消息的语法，它能够被转换成标记语言中用于人力资源的另一个：HR-XML，这样你将会看到这些消息是怎样的。

### 12.7.1 HR-XML

HR-XML 组织是一个非赢利的组织以开发和促进人力资源相关的 XML 词汇表标准，目的是为了实现电子商务和人力资源数据交换自动化。它的成员包括：Icazrian 公司、SAP、IBM、Skill Village 公司、job.com 公司、Personic 软件公司、Lawson、Structured Methods 公司以及 J.D. Edwards（虽然写本书时 PeopleSoft 公司和 Oracle 公司正在评估它）。HR-XML 组织已经开发了 3 个预备模式，每一个都可以作为一个 DTD 或一个 XML 框架被 BizTalk 组织确认：

- 职位发布（Job posting）

- 简历 ( Resumes )
- 候选人概要 ( Candidate profiles )

让我们看看这三个步骤是如何工作的，以及一些如何使用的例子。

注意提供这些例子是为了说明一些在 HR-XML模式中的长度和细节标准。你应该更为细致地阅读这些模式的细节以获得如何使用的全部信息。

可以在下面的网站获得关于HR-XML组织各个模式标准的文档：<http://www.hr-xml.org/>。

#### (1) 职位发布

开始时，一个职位通过使用职位发布模式进行发布，详细介绍一个特定的职位。这些信息能够很容易地在雇员、新进人员、职业 Web网站和职业介绍所之间进行交换。包括一个职位发布文档的信息应能够表明雇主是谁、联系方法和职位描述以及如何申请这个职位。

文档的例子可以在下面的网站下载，你会获得本书没有显示的其他代码：<http://www.wrox.com>。文件名叫做posting.xml。

为了显示如何使用职位模式，我们将看一个发布的例子。我们首先定义 BizTalk的报头部分，接下来是0.81 BizTalk模式以用于JobPosting模式（在写本书时这些模式还没有升级到已经发布的1.0版）。

#### 程序清单 12-7

```
<?xml version="1.0" ?>
<BizTalk xmlns="urn:schemas-biztalk-org/biztalk-0.81.xml">
  <Body>
    <JobPosting xmlns="urn:schemas-biztalk-org/HR-XML-org/JobPosting">
```

我们首先描述了雇主组织的信息，将在 HiringOrg元素中进行显示，通过 type属性的agent值（模式将这个值定义为一个代理、主要的或非指定值的计数）。我们开始提供一些通常的关于雇主组织细节，而Citix代理公司作为代理的例子。

#### 程序清单 12-8

```
<HiringOrg type="agent">
  <OrgName>Citix Agents Co.</OrgName>
  <Website>http://www.citix.com</Website>
  <BusType>Technical Staff Provider
    <SIC>99371234</SIC>
  </BusType>
  <EmployerDesc>Provide contract staff for international XML
    positions.
  </EmployerDesc>
```

接下来完成的是一些联系细节的信息，包括联系人姓名（你的真名！）以及联系地址（请不要向这些永远收不到的地址发送任何东西），电话号码以及电子邮件地址。

#### 程序清单 12-9

```
<Contact>
  <Name><First>Steven</First><Last>Livingstone</Last></Name>
  <JobTitle>Technical Director</JobTitle>
```

```

<Address>
  <AddressLine>1022 Brinlow Avenue</AddressLine>
  <AddressLine>Yoker</AddressLine>
  <City>Glasgow</City>
  <PostalCode>G13A 999</PostalCode>
  <Country>Scotland</Country>
</Address>
<PhoneNumbers>
  <Voice>
    <AreaCode>001</AreaCode>
    <TelNumber>100000</TelNumber>
  </Voice>
</PhoneNumbers>
<Email>ceo@citix.com</Email>
</Contact>
</HiringOrg>

```

现在，由于我们已经说明雇主组织是一个代理，他们必须表明真实的雇主。模式将其定义为PrincipalEmployer，并要求一些关于雇主组织的基本信息。我们的公司叫做 Global XML公司，是一个软件开发商，其代码如下所示：

#### 程序清单 12-10

```

<PrincipalEmployer>
  <OrgName>Global XML Inc.</OrgName>
  <Website>http://www.global-xml-inc.com</Website>
  <BusType>Software Development
    <SIC>83648236</SIC>
  </BusType>
  <EmployerDesc>Global Software company specialising in
    E-Commerce Internet Content Development.
  </EmployerDesc>
</PrincipalEmployer>

```

职位发布文档的最后部分给出了关于实际职位的细节。下面的例子是一个 XML开发人员，在Glasgow工作，以及在JobPurpose元素中描述一些提供这个职位的原因（不必要）。我们同样可以强调说明应聘人员的素质要求 QualifRequired，如使用过的软件包和一般经验。最后，HowToApply元素将包含向雇主提交的一张申请表的任何信息。

#### 程序清单 12-11

```

<JobInformation>
  <JobTitle>XML Developer</JobTitle>
  <Location>Glasgow, Scotland</Location>
  <Description>
    <JobPurpose>
      <ul>
        <li>Develop XML Schema's representing the business
          processes of our E-Commerce partners;
        </li>
        <li>To aid software developers with their
          implementation of XML techniques;
        </li>
        <li>Advise of the implementation and progression of

```

```

        XML techniques;
    </li>
</ul>
</JobPurpose>
<QualifRequired>
<SoftwareRequired>XML Wizard '99</SoftwareRequired>
<ExperienceRequired YearsOfExperience="1">
    You must have some commercial experience of XML.
</ExperienceRequired>
</QualifRequired>
<Preferences>
    <p>Prefer candidate with prior experience in the
        banking or financial industry.
    </p>
</Preferences>
<Compensation>
    <p>You'll make a mint!</p>
</Compensation>
</Description>
<HowToApply>
    <p>Send in a full CV and some examples of your work.</p>
</HowToApply>
</JobInformation>
</JobPosting>
</Body>
</BizTalk>

```

现在，你已经知道一个职位如何在发布时标记了，让我们看看一个响应上述职位的简历的例子。

## (2) 简历

简历模式允许一个职位代理创建一个“丰富”的简历，它可以用在数据中心并十分易于查询。标签上的标记允许在标签等级保证个人信息的安全。简历文档也可以发到雇主或代理以按照他们的条件进行选取。

作为例子的文档可以在下载的文件 resume.xml 中找到。我们的例子以通常要求的 BizTalk 头部开始。

### 程序清单 12-12

```

<?xml version="1.0"?>
<BizTalk xmlns="urn:schemas-biztalk-org/biztalk-0.81.xml">
<Body>
<Resume xmlns="urn:schemas-biztalk-org:HR-XML-org/Resume">

```

ResumeProlog 部分允许个人指定其受雇条件，如时间和报酬信息（如当前的报酬和要求的报酬）。

### 程序清单 12-13

```

<ResumeProlog>
    <RevisionDate>
        <Date><Day>5</Day><Month>January</Month><Year>1999</Year></Date>
    </RevisionDate>
    <AvailabilityDate>

```



```
<StartDate><Day>15</Day><Month>January</Month>
    <Year>2000</Year></StartDate>
<EndDate><Day>5</Day><Month>February</Month>
    <Year>2000</Year></EndDate>
</AvailabilityDate>
<CompensationDetail>
  <Rate>
    <Current>£45 per hour</Current>
    <Required>£50 per hour</Required>
  </Rate>
  <Benefits>
    <Required>Travel Expenses and Lunch coupon desired.</Required>
  </Benefits>
</CompensationDetail>
</ResumeProlog>
```

最后的部分是 ResumeBody，可以在 PersonalData 和 ResumeSection 标签中包含更多的信息。PersonalData 标签包含关于个人简历的联系信息，如个人姓名、地址、电话号码以及电子邮件地址。

#### 程序清单 12-14

```
<ResumeBody>
  <PersonalData>
    <Name><First>Jane</First><Last>Doe</Last></Name>
    <JobTitle>Primary Web Developer</JobTitle>
    <Address>
      <AddressLine>973 FunnyName Road</AddressLine>
      <AddressLine>Anderton Place</AddressLine>
      <City>SaddleWich</City>
      <PostalCode>H77 8I9</PostalCode>
    </Address>
    <Email>ntw_uk@hotmail.com</Email>
    <Voice>
      <AreaCode>09772</AreaCode>
      <TelNumber>980-0283</TelNumber>
    </Voice>
    <EmployeName>InterCo Internet Consultants Ltd.</EmployeName>
  </PersonalData>
</ResumeBody>
</Resume>
</Body>
</BizTalk>
```

可能有多个 ResumeSection 部分用于每个类型属性 ( SecType )，类型属性在模式中定义如下所示：

- Objective
- Experience
- Personal
- References
- Education
- Certifications

- Licenses
- QualifSummary ( 资历简介 )
- Skills
- ProfAssociations ( 专业协会成员 )
- Unspecified

这些部分用于提供一些关于资历的详细信息，技能和经验部分常常被查询，但会作为个人隐私数据。每个部分包括一个 SectionTitle ( 部分名称 ) 以及相关的细节如资历简介，如下所示：

程序清单 12-15

```
...
<ResumeSection SecType="QualifSummary">
  <SectionTitle>University Qualifications</SectionTitle>
  <SecBody>
    <P>BSc in Physics and a MSc in Information Technology Systems
      at Strathclyde University.</P>
  </SecBody>
</ResumeSection>

<ResumeSection SecType="Experience">
  <SectionTitle>Working with
    <EmployerName>InterCo Internet Consultants Ltd.</EmployerName> as the
    <JobTitle>Development Lead</JobTitle> which also involved being a
    <JobTitle>Technical Documenter</JobTitle> and
    <JobTitle>System Tester</JobTitle>.
    <StartDate>
      <Date><Month>June</Month><Year>1998</Year></Date>
    </StartDate>
    to present. Also gained my <CertificationQualif>Level 2 International
    Certification in XML</CertificationQualif>.
  </SectionTitle>
</ResumeSection>
</ResumeBody>
</Resume>
</Body>
</BizTalk>
```

简历部分同下面将要讲到的候选人概要部分很相似，但后者更为简洁并有更多的一致结构。我们将看看 HR-XML 最后一个标准模式——候选人概要。

### (3) 候选人概要

候选人概要允许在雇主、新进人员、职业 Web 网站和职业介绍所之间交换候选人信息，并可以获得关于候选人工作目标、资历以及工作和教育历史的数据。相关的数据可能是：

- 从非结构化的简历词汇分析获得的。
- 工作候选人在在线申请表中填写的。
- 通过电话交谈收集的。
- 从 XML 标记的简历中抽取的。

例子文档来自下载的 profile.xml 文件。下面的例子展示了一个同 BizTalk 兼容的候选人概要文档在开始时必须的元素：

## 程序清单 12-16

```
<?xml version="1.0"?>
<BizTalk xmlns="urn:schemas-biztalk-org/biztalk-0:81.xml">
<Body>
<CandidateProfile xmlns="urn:schemas-biztalk-org:HR-XML-org/CandidateProfile">
```

PostDetail元素包含可以使用这个文档的范围信息，如下所示。

## 程序清单 12-17

```
<PostDetail>
  <StartDate>
    <Date><Month>January</Month><Day>05</Day><Year>2000</Year></Date>
  </StartDate>
  <EndDate>
    <Date><Month>February</Month><Day>25</Day><Year>2001</Year></Date>
  </EndDate>
</PostDetail>
```

JobCandidateContact元素包括候选人的联系信息，如个人姓名、地址、电话号码和电子邮件地址。

## 程序清单 12-18

```
<JobCandidateContact>
<Name>
  <First>Steven</First>
  <Last>Livingstone</Last>
</Name>
<Address>
  <AddressLine>973 FunnyName Road</AddressLine>
  <AddressLine>Anderton Place</AddressLine>
  <City>SaddleWich</City>
  <PostalCode>H77 8I9</PostalCode>
</Address>
<PhoneNumbers>
  <Voice>
    <AreaCode>09772</AreaCode>
    <TelNumber>980-0283</TelNumber>
  </Voice>
</PhoneNumbers>
<Email>ntw_uk@hotmail.com</Email>
</JobCandidateContact>
```

Objective元素可以用来显示一个候选人在某个潜在职位上的特别的条件或优先条件。在下面的例子中，主要的条件是在英国、每周工作 37 小时。

## 程序清单 12-19

```
<Objective>
  <Location>US</Location>
  <Industry>Software Consulting</Industry>
  <Schedule>
    <FullTime>
      <HoursPerWeek>37</HoursPerWeek>
```

```

        </FullTime>
    </Schedule>
    <AvailabilityDate>
        <Date><Day>15</Day><Month>1</Month><Year>2000</Year></Date>
        <Comment>4 weeks notice is needed to my current employer.</Comment>
    </AvailabilityDate>
</Objective>

```

接下来我们突出资历，可以包括技术资格、经验、软件经验、编程语言、许可证、证书、设备知识、硬件和操作系统知识。

每个元素都可以具有 YearsOfExperience 属性、等级（0~5等）以及你对该项感兴趣的等级。

#### 程序清单 12-20

```

<Qualifications>
  <CertificationQualif>
    Level 2 International Certification in XML
  </CertificationQualif>
  <ExperienceQualif YearsOfExperience="2" level="4" interest="4">
    XML Commercial Development
  </ExperienceQualif>
  <PrgmLangQualif level="4" YearsOfExperience="6" interest="3">
    Visual Basic
  </PrgmLangQualif>
  <PrgmLangQualif YearsOfExperience="6" level="3" interest="2">
    C++
  </PrgmLangQualif>
  <SkillsQualif YearsOfExperience="2" level="3" interest="4">
    Development Lead
  </SkillsQualif>
  <SkillsQualif YearsOfExperience="4" level="4" interest="2">
    Technical Documenter
  </SkillsQualif>
  <SkillsQualif YearsOfExperience="7" level="3" interest="1">
    System Tester
  </SkillsQualif>
</Qualifications>

```

在我们例子的最后，要说明的是候选人的历史（History），包括教育（Education）情况和职位（Position）情况。教育情况应该给出进入大学的细节以及获得的学位和学习时间。

#### 程序清单 12-21

```

<History>
  <Education>
    <School>Strathclyde University</School>
    <Location>Glasgow, Scotland.</Location>
    <Degree>BSc Physics</Degree>
    <StartDate>
      <Date><Year>1992</Year></Date>
    </StartDate>
    <EndDate>
      <Date><Year>1996</Year></Date>
    </EndDate>
  </Education>

```

我们将在讲完候选人的工作经验后结束候选人概要部分，工作经验部分详细介绍雇主姓名、职位名称以及工作情况的简介。

程序清单 12-22

```
<Position>
  <EmployerName>InterCo Internet Consultants Ltd.</EmployerName>
  <Location>Milan, Italy.</Location>
  <JobTitle>Development Lead</JobTitle>
  <Industry>Software Development</Industry>
  <StartDate>
    <Date><Month>June</Month><Year>1998</Year></Date>
  </StartDate>
  <EndDate>
    <Date><Month>January</Month><Year>2000</Year></Date>
  </EndDate>
  <Summary>
    <p>Developed XML Software applications for the company Intranet.</p>
  </Summary>
</Position>
</History>
</CandidateProfile>
</Body>
</BizTalk>
```

上面的例子可以使你对 HR-XML 标准有一个好的理解，XML 在允许许多雇主、雇员和代理之间交换雇佣和候选人信息方面起着重要作用。你应该浏览全部模式以了解该标准所有的能力。但是，例子展示了不同的消息如何分割以及 XML 如何使对工作的描述更加容易。记住，如果你要求更多的信息那么内部对其进行扩展是可能的，虽然扩展的信息不能够被没有准备接受你的扩展的其他方所识别。

上述已经被设计成 DTD 模式和 Microsoft 的 BizTalk 兼容的模式，所以基于上述模式创建的文档可以在雇员和雇主之间无缝交换。

更多的信息请参见：<http://www.hr-xml.org/>。

但这并不是在人力资源领域的唯一一个起步的标准，另一个标准已经包括在由应用理论通信（ATC）和美国职位银行（AJB）合作开发的标记语言工作组（WFML）中。AJB 是美国联邦政府机构，目的是统一和保存从州雇佣代理机构获得的简历和职位列表。该组织的目的是收集使用 XML 标准化的人力资源/电子招聘系统。更多的信息请浏览 <http://www.xml-hr.org>。另外，还有一个人力资源标记语言（HRML），由 Datamain 公司创建。HRML 主要注重的是职位发布，已经作为一个职位列表和简历的国际表示方式而被广泛使用。最近加入到网站上的版本包括信封、私有标签、工作类别标签。更多的关于 HRML 的信息请浏览：<http://www.hrml.com/>。

## 12.7.2 健康

根据电子健康市场报告，现在基于 Web 健康方面的应用提供商大都使用 XML 进行编码，如 Microsoft 公司和计算机科学公司都把 XML 作为一种减少成本、缩短时间的方法。关于 Windows 的健康 V 系统，Microsoft 健康用户组年会在 San Diego 举行，该公司声称他们正在开发一种特别



用于健康的基于Windows的分布式Internet结构（DNA）叫做Window DNA for Healthcare。同时，计算机科学公司（CSC）发布了一个XML标记产品，开发中得到了美国国防部的联合计算机辅助查询和逻辑支持机构的程序管理员的协助。XML为具有更多多媒体要求的健康行业提供了更多新的思路。

#### 1. HL7 SGML/XML 专门兴趣组

HL（健康等级）7 SGML/XML 专门兴趣组的名字使人一见就知道是使用XML进行开发。SIG以前主要定位在医药行业，如药方、政策和制造、临床注释以及电子病历（EPR），同时也定位于传统的基于XML表达式的HL7消息。

HL7——健康等级7是ANSI授权的标准化开发组织（SDO）之一，主要工作在健康领域。许多SDO产品标准（有时叫做协议或规范）适用于特殊的健康领域，如制药、医用设备、图像或保险（要求处理）交易。HL7的主要领域是临床和管理数据。账单简化报告、保险要求、流行病报告以及一出生就死的病人报告通过抽取、复制、连接或结合等方式从基本的临床文档中获得。

合作研究组（New Rochell, NY）报告互联网上的制药行业将在2001年产生1.4到2.8亿美元的年收入。到某一天，医生将通过安全系统进入网站上的处方机构。这样：

- 确认医生的资格。
- 检查病人适用的格式申请。
- 检查病人的建议药品反应的个人历史。
- 引导病人进入适用的疾病管理程序。
- 以最合适的价格购买药品。
- 将药品发送给病人。
- 从病人的信用卡中扣除费用。
- 公告健康计划。
- 调节商店库存。

下面是一个使用XML编写处方的例子：

程序清单 12-23

```
<PRESCRIPTION>Viagra
  <FORM>50 mg. capsule</FORM>
  <DISPENSE>6</DISPENSE>
  <DOSAGE>1 cap(s)po</DOSAGE>
  <INSTRUCTIONS>As desired</INSTRUCTIONS>
  <REFILL>1</REFILL>
  <SUBSTITUTE>may substitute</SUBSTITUTE>
  <NUMBER NOTYPE="DEA">AB1234567</NUMBER>
</PRESCRIPTION>
```

SIG采用的结构方法设计良好并实现了原子化的概念，如帐单代码或控制的词汇表能够在结构的任何等级插入作为标记文本。例如，HL7病人记录计划提供代码和原子化标记结构，将在每个等级被定义在结构化的DTD中。

- 等级1——可编码的报头。等级1提供人们可读内容的完全互操作性，但并不指定译码成互操作的机器处理代码。
- 等级2——可编码上下文。文档主体结构化地分成几个部分以支持最小的处理，如填写表格、安全采用临床内容。它包含同样的等级1可编码的报头。
- 等级3——可编码内容。文档的主体必须是完全支持XML的结构和规范，以符合HL7参考信息模式（RIM），同时必须符合上述两个等级。

更多的信息请参阅<http://www.mcis.duke.edu/standards/HL7/sigs/sgml/index.htm>。

看完了上面的三个例子，你应该知道XML如何使用在纵向行业。我们展示了开放旅游联盟如何使用XML以集成旅游行业不同的部分以提供服务。根据OTA模式标记其数据的公司将能够发送他们的信息到任何理解它的人。我们观看了使用在HR-XML中的语法和消息的类型，它给出了如何标记商业信息的绝好例子。我们还看到了HL7如何起步，这样我们可以同更广大的用户共享健康信息，不仅使我们在订购处方药时更容易，更可以允许访问紧急药品记录使我们生活更安全。并不是说上述的工作离开XML就不可以完成，而是说使用XML是一种潮流，软件提供商承诺将可能变成现实。

## 2. RosettaNet

RosettaNet的目的是开发一套行业范围的电子商务互操作标准以帮助行业成员集成他们的商业会话。RosettaNet的模式是基于语言的概念要求同日常口语相关联，并要求在B2B交易中被完全理解。如果我们想象通过电话进行的商业订购方式，为了一个会话的产生，需要能够产生并听见“声音”。如果我们扩展这个比喻，我们需要一个字母表以产生单词。如果使用单词并应用语法，就可以创建一个对话。对话就可以形成一个商业进程，它可以在一些仪器中传输如电话。

这个模式听起来比较简单，但是它基于标记和单词。如果你看完了图12-16，你就会理解RosettaNet如何将语言模式应用在电子商务领域。

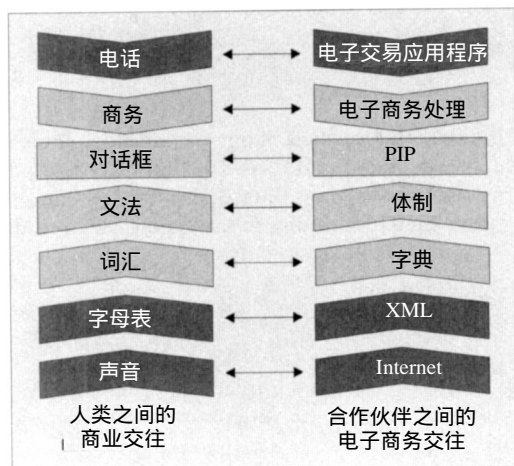


图 12-16

在Internet中它可以同产生声音的过程相比较。XML作为该模式中的一个字母表，而电子商务应用程序作为用于交换电子商务进程的工具。

为了激发电子商务获得的潜能，需要字典、框架、伙伴界面处理和电子商务进程。RosettaNet尝试通过注重于建立一个主字典以填补这个缺陷，主字典可以用来定义产品、伙伴和商务交易的属性。主字典以及其实现框架（怎样执行交换）用来支持作为伙伴界面处理（PIP）的对话。

RosettaNet已经定义了下列任务，并正在逐步实现：

- 商业处理模块——定义并认证在供应链每个等级独立的当前商业处理元素，并进行分析以发现任何不适当的匹配和低效率。
- 通过商业处理分析仿真模型，一个通常的处理显示，展示对伙伴界面处理（PIP）目标列表表格重新校对的机会，以及评估实现PIP结果的商业影响（主要是时间和金钱方面的功能）。
- 从中创建一个PIP蓝图以处理规范，包括伙伴角色（买方、卖方、装配商、分类发布商等等）互操作界面以达到一定的商业目标。它包括特定的 PIP服务、交易和消息——包括目录属性以及以UML形式的分类和查询图表。
- 为PIP蓝图创立的PIP协议。该协议规范基于网络的应用程序如何遵守 RosettaNet实现框架以获得同执行在一个 PIP蓝图中特定进程的互操作性。它包括一个基于 DTD实现框架的XML文档（或一些文档），一个有效的工具和每个界面的实现目标。
- 在并行处理中，必须开发两个数据目录以提供被 PIP要求的通用属性集。首先是一个技术属性目录（所有过程分类的技术规范），其次是一个商业属性目录包括价目属性、伙伴属性（用于描述供应链伙伴公司属性）和商业交易属性。

在开发数据目录时，语言相关的元素几乎是 X12和EDIFACT代码列表的三倍，段和元素为目录提供了基本的来源。实现处理更多交互性的关键是使语言更多地满足组织的需求，而不是尝试使用通常的共享数据定义。

更多的关于RosettaNet的信息请浏览：[http://www.rosettaNet.org/general/index\\_general.html](http://www.rosettaNet.org/general/index_general.html)

RosettaNet正在产生实际的结果：10月实现了42个PIP，以及在15个RosettaNet伙伴中产生6个成功的领先者，并向其提供信贷。

## 12.8 cXML——纵向行业的一个详细例子

cXML版本1.0是一个在Internet使用XML实现B2B电子商务交易的协议。有两个可以利用的机制用于应用程序的交换，允许在 HTTP协议上的基于请求-响应交易机制，或不限特定方法的单向传输。在写该书时，有计划将 cXML移植到BizTalk兼容的模式中。

### 12.8.1 为什么有cXML

客户对商家（C2B）和商家对商家（C2C）交易越来越全球化，而分布式集成成为电子商务应用成功的关键。直到最近，实现商务应用集成还是一个困难和昂贵的事，但是随着如 BizTalk和cXML的出现，将来该项事业将更为简单和便宜。

本部分将具体介绍什么是 cXML以及它如何能够用在你的商业应用系统中以提供一个十分有效和开放的系统。

可以在下面的站点下载cXML规范：<http://www.cXML.org>

## 12.8.2 cXML协议规范

cXML协议对一个XML文档实例的结构提供三个主要的可选部分。前两个将该文档分成两个逻辑部分，如同我们在本章前面部分解释的那样——具有<Header>元素的报头部分和具有<Message>或<Request>元素的主体部分——第3部分是一个单独的<Reponse>元素。报头部分主要控制传输信息，主体部分可以包含任何XML信息，但更常见的是cXML商业文档定义——我们将做简要介绍。

在这个部分看一个实际的例子是很有用的，先了解cXML文档看起来是一个怎样的模式，接下来将解释出现在文档中的概念。一个cXML文档的实例会如下所示，它展示了一个消息（在<Request>元素中），通过payloadID唯一定义，从steven@somewhere.com发送，由donna@anotherplace.com和kod@anotherplace.com接收。

程序清单 12-24

```
<cXML version="1.0" payloadID="19991001.8263.872344@somewhere.com"
timestamp="1999-10-01T01:11:03-05:35">
  <Header>
    <From>
      <Credential domain="SomeUserIDForDomain">
        <Identity>steven@somewhere.com</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="RemoteDomainUserID">
        <Identity>donna@anotherplace.com</Identity>
      </Credential>
      <Credential domain="AnotherRemoteDomainUserID">
        <Identity>kod@anotherplace.com</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="SomeUserIDForDomain">
        <Identity>steven@somewhere.com</Identity>
        <DigitalSignature type="PK7 self-contained"
          encoding="Base64">
        </DigitalSignature>
      </Credential>
      <UserAgent>Ariba ORMS 6.0</UserAgent>
    </Sender>
  </Header>
  <Request deploymentMode="test">
    cXML Business Document Definition ...
  </Request>
</cXML>
```

让我们看看这个文档的重要部分以及它是如何满足cXML规范的。

### 1. cXML信封

cXML信封是一个cXML实例必须的部分，它封装报头和主体信息以创建一个唯一的根（root）元素——在cXML中是<cXML>。cXML根元素的DTD表示为：

程序清单 12-25

```
<!ELEMENT cXML ((Header, Message) |
                 (Header, Request) |
                 (Response))>

<!ATTLIST cXML
  version      %uint;          "1.0"
  payloadID    %string;        #REQUIRED
  timestamp    %datetime.tz;   #REQUIRED
>
```

在cXML根元素中有三个属性可以使用（参见表 12-3）。

表 12-3

属 性	描 述	值
Version	cXML协议的版本状态。在本书的写作时为 1.0	"1.0"
PayloadID	唯一的定义符，基本目的是能够在需要时定义和跟踪消息。建议使用	Datetime.process id.random number@hostname
timestamp	该属性以ISO8601格式表示消息发送时的日期和时间。这意味着我们的时间必须符合该格式	YYYY-MM-DDThh:mm:ssTZD

timestamp按下列方式读取：

YYYY       =四位数字表示的年份  
MM           =两位数字表示的月份（01=1月，等等）  
DD           =两位数字表示的月中的日期（01~31）  
hh           =两位数字表示的小时（00~23）（不使用AM/PM）  
mm           =两位数字表示的分钟（00~59）  
ss           =两位数字表示的秒（00~59）  
TZD          =时区描述

时区描述的值如下所示：

Z：表示为UTC（协调的通用时间），Z必须大写。  
+hh:mm：表示本地时间比UTC快hh小时mm分钟。  
-hh:mm：表示本地时间比UTC慢hh小时mm分钟。

更多的关于ISO8601的信息请浏览：

<http://www.w3.org/TR/PR-html40/types.html#h-6.11>

<http://www.saqqara.demon.co.uk/datefmt.htm>

在上面的例子中，已经定义了根标签：

```
<cXML version="1.0" payloadID="19991001.8263.872344@somewhere.com"
timestamp="1999-10-01T01:11:03-05:30">
```

上面文档的实例是 1999年10月1日，进程ID是8263，随机数是872344，在主机 somewhere.com上进行的处理。

## 2. 报头部分



如上实例所示，报头部分必须包括三个重要的成分。这三个部分是：

- From部分
- To部分
- Sender部分

其DTD描述如下：

程序清单 12-26

```
<!ELEMENT Header (From, To, Sender)>

<!ELEMENT From (Credential+)>
<!ELEMENT To (Credential+)>
<!ELEMENT Sender (Credential, UserAgent)>
<!ELEMENT UserAgent (#PCDATA)>
```

注意整个报头被封装在元素信封（cXML元素）中，它包含一些子元素以描述文档的传输。我们现在将看看这些子元素在cXML文档中的角色。

### 3. From

From元素如同电子邮件中的From一样，它简单定义了请求的来源。可能包含一个或多个Credential元素以允许请求者定义自己。

关于From元素的例子描述了用户，其电子邮件地址是 steven@somewhere.com，他的域ID是 SomeUserIDForDomain，如下所示：

程序清单 12-27

```
<From>
<Credential domain="SomeUserIDForDomain">
<Identity>steven@somewhere.com</Identity>
</Credential>
</From>
```

我们将在后面对Credential元素进行简短介绍。

### 4. To

同使用From元素描述消息的来源类似，使用To元素描述请求接受者的信息。在我们的例子中，定义了两个域ID以及其电子邮件地址以定义目的用户。

程序清单 12-28

```
<To>
<Credential domain="RemoteDomainUserID">
  <Identity>donna@anotherplace.com</Identity>
</Credential>
<Credential domain="AnotherRemoteDomainUserID">
  <Identity>rambo@anotherplace.com</Identity>
</Credential>
</To>
```

### 5. Sender

Sender元素在使用cXML实例进行授权传输中起到十分重要的作用，它是在报文通过系统时

报头中唯一改变的元素。通常，它同 From元素类似，虽然在 Sender元素中通常包括带有授权信息的 Credential元素以定义发送者。Sender元素通常包含授权信息，它也是在 cXML消息路由通过系统时唯一会改变的元素。

在我们的例子中，有一个具有 SomeUserIDForDomain域ID的用户，而 steven@somewhere.com 作为实例中发送者的定义。

#### 程序清单 12-29

```
<Sender>
  <Credential domain="SomeUserIDForDomain">
    <Identity>steven@somewhere.com</Identity>
    <DigitalSignature type="PK7 self-contained" encoding="Base64">
    </DigitalSignature>
  </Credential>
  <UserAgent>Ariba ORMS 6.0</UserAgent>
</Sender>
```

注意在我们的例子中具有 Credential元素，我们使用 Identity标记来包含发送者的电子邮件地址，同时使用 DigitalSignature元素作为类似的口令以使系统具有认证机制。除了 DigitalSignature方法，ShareSecret元素也可以在发送方具有用户名/口令时使用，而 HTTP连接那一端的请求响应者应该识别该内容。

注意，我们在 Sender元素中还使用了一个 UserAgent元素——我们将在下面解释这个元素。

#### 6. Credential

Credential元素用于允许用户使用同 SMTP协议中的 From域相同的方式定义他自己，其具有一个叫做 domain的属性允许我们将这个实例同特定的域发送者关联。如果需要，可能使用不止一个的 Credential元素以定义发送者，这样允许请求者使用多种方式定义自己（同 SMTP和 X.400 协议中电子邮件地址类似）。

下面是在 From元素中使用的 Credential元素的 DTD描述：

#### 程序清单 12-30

```
<!ELEMENT Identity ANY>
<!ELEMENT Credential (Identity, (%cxml.authentication;?))>
<!--ATTLIST Credential
  domain %string; #REQUIRED
-->
```

上面的 DTD部分对 cXML报头中的 From元素来说通常是足够了。

但是，对于 Sender元素，它通常包括一些授权信息和一些用户信息。这时对于 Credential元素的 DTD描述如下：

#### 程序清单 12-31

```
<!ELEMENT DigitalSignature ANY>
<!--ATTLIST DigitalSignature
  type %string; "PK7 self-contained"
  encoding %string; "Base64"
-->
```

```
<!ELEMENT SharedSecret ANY>
<!ELEMENT Identity ANY>
<!ENTITY % cxml.authentication "SharedSecret | DigitalSignature">
<!ELEMENT Credential (Identity, (%cxml.authentication;)?)>
<!--LIST Credential
    domain %string; #REQUIRED
-->
```

DigitalSignature有两个属性，参见表 12-4。

表 12-4

属 性	描 述	值
type	数字签名的类型，建议使用 PK7 self-contained	建议： PK7 self-contained 也可为其他值
encoding	如何在 HTTP 中对签名进行编码。缺省是在 Web 中广泛使用的“base64”编码，你也可以选择自己感兴趣的编码格式	缺省： base64 也可为其他值

当两个合作伙伴同意使用一种通用的证书格式时可以使用 DigitalSignature 元素，同时通过类型属性定义认证类型。

另一个认证方法是使用 SharedSecret 元素，它充当了口令的角色，其同 Identify 的结合就好像用户名/口令一样。通常，在一个传输进行前需要对其进行安全交换。

#### 7. UserAgent

UserAgent 元素通过使用简单的字符串来表示 UserAgent，它接受 cXML 文档，同时我们在 HTTP 中也有 UserAgent。

这样我们完成了一个 cXML 文档中对报头和信封的介绍。下面将看看消息传输的不同方法。

### 12.8.3 消息传输

在 cXML 中，消息传输有两种实现方法：

- 请求/响应消息传输。
- 不对称或单向传输。

让我们看看这些方法是如何实现的。

#### 1. 请求/响应消息传输机制

一个 cXML 请求/响应的流程如图 12-17 所示。客户发送一个 cXML 请求到一台服务器并保证在每个信封中只有一个请求，这样简化了服务器的实现。发送的 XML 数据可能为任何类型，但 cXML 的主要目的是使用在 cXML 实现中指定的请求类型。

#### 2. 请求

一个 cXML 请求实例应该使用下面的模式：

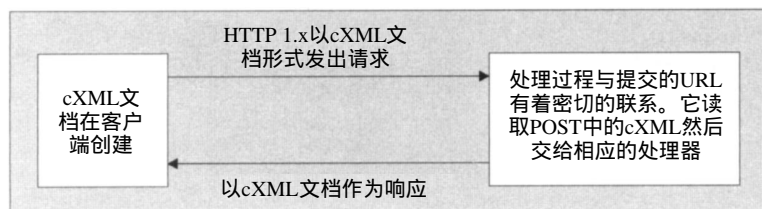


图 12-17

程序清单 12-32

```
<cXML>
  <Header>
    Our header details here
  </Header>
  <Request>
    Our Request details here
  </Request>
</cXML>
```

模式的DTD如下所示：

程序清单 12-33

```
<!ENTITY % cxml.requests "OrderRequest |
                          PunchOutSetupRequest |
                          GetPendingRequest |
                          SubscriptionListRequest |
                          SubscriptionContentRequest |
                          SupplierListRequest |
                          SupplierDataRequest"
>

<!ELEMENT Request (%cxml.requests;)>
<!ATTLIST Request
  deploymentMode (production | test) "production"
>
```

Request元素拥有一个属性叫做 deploymentMode，该属性可能的取值为 production或test，取值同你是否正在测试这个系统还是已经发布使用该系统。Request值的定义是cXML规范的一个部分，用于允许开放消息传输。

下面是在测试过程中的一个供应商列表的请求，买者同其有着被供应关系：

程序清单 12-34

```
<Request deploymentMode="test">
  <SupplierListRequest/>
</Request>
```

### 3. 响应

响应元素发送给客户告知应用程序运行结果，甚至在没有任何结果时也发送。一个被成功处理的cXML文档可能具有以下模式：

## 程序清单 12-35

```
<cXML>
  <Header>
    Our header details here
  </Header>
  <Response>
    <Status code="200" text="OK"/>
    Our Request details here
  </Response>
</cXML>
```

下面我们看一看一个响应实例的 DTD 描述：

## 程序清单 12-36

```
<!ELEMENT Status (#PCDATA)>
<!ATTLIST Status
  code %uint;      #REQUIRED
  text %string;    #REQUIRED
>
<!ENTITY % cxml.responses "OrderResponse |
                           PunchOutSetupResponse |
                           GetPendingResponse |
                           SubscriptionListResponse |
                           SubscriptionContentResponse |
                           SupplierListResponse |
                           SupplierDataResponse"
>
<!ELEMENT Response (Status, (%cxml.responses;?))>
```

Response 元素必须包含一个 Status 子元素，它表明该实例是否被成功处理，并在不成功时显示原因。该元素有 code 和 text 属性，code 属性的值同 HTTP 相似模块的值类似，而 text 属性包含对 code 的文字描述（用于日志等）。使用 500 表示错误，错误、XML 翻译错误或应用程序错误应在返回的 XML 文字中加以表示以帮助纠错。

Response 也可能遵循 cXML DTD 规范，我们将进行简单描述。对先前的 SupplierListRequest 例子的响应例子如下所示。

## 程序清单 12-37

```
<Response>
  <Status code="200" text="OK"/>
  <SupplierListResponse>
    <Supplier corporateURL="http://www.citix.com"
      storeFrontURL="http://www.deltabiz.com">
      <Name xml:lang="en-US">Citix Co</Name>
      <Comments xml:lang="en-US">Our biggest customer</Comments>
      <SupplierID domain="DUNS">475435</SupplierID>
    </Supplier>
  </SupplierListResponse>
</Response>
```

注意你可以在 cXML 规范文件 supplier.mod 中查看 SupplierListRequest DTD 的内容，该文件可



以从<http://www.cxml.org>下载。

另一种cXML使用的消息传递机制是不对称或单向传输。我们将在下面看到。

#### 4. 不对称传输机制

在同步传输机制（如HTTP）无效时使用不对称传输，这样我们不能保证我们的请求一定会得到一个响应。图12-18显示了一个不对称传输响应逻辑上的流程。

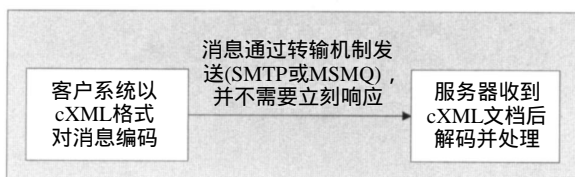


图 12-18

一个不对称cXML消息的结构同请求/响应格式的消息结构并不完全不同。

程序清单 12-38

```
<cXML>
  <Header>
    Our header details
  </Header>
  <Message>
    Our message details
  </Message>
</cXML>
```

事实上，cXML单向实例的结构同请求/响应结构的不同在于Message元素。Message元素的DTD描述如下。

程序清单 12-39

```
<!ENTITY % cxml.messages "PunchOutOrderMessage |
                           PunchOutOrderAckMessage |
                           SubscriptionChangeMessage |
                           SupplierChangeMessage"
>

<!ELEMENT Message (Status?, (%cxml.messages;))>
<!ATTLIST Message
  deploymentMode (production | test) "production"
  inReplyTo      %string; #IMPLIED
>
```

Message元素具有一个可选的状态（Status）子元素，如上所描述。它还有两个可以利用的属性——deploymentMode属性在前面已经介绍过，而inReplyTo属性可选。

为了模仿不对称的请求/响应功能，inReplyTo属性表明响应消息将要响应给何人，可以是原始消息的payloadID。

下面是一个不对称请求/响应的例子，用于交换一个买者的内容描述。

```
<Message inReplyTo="19991001.8263.872344@somewhere.com">
  <SubscriptionChangeMessage type="update">
    <Subscription>
      <InternalID>6235</InternalID>
      <Name xml:lang="en-US">Annual Cost</Name>
      <Changetime>2000-01-13T11:33:01-28:10</Changetime>
      <SupplierID domain="DUNS">82735236</SupplierID>
      <Format version="2.1">CIF</Format>
    </Subscription>
  </SubscriptionChangeMessage>
</Message>
```

你可以在 cXML 规范文件 Subscription.mod 中查看 SubscriptionChangeMessage cXML 模块，该文件可以从 <http://www.cxml.org> 下载。

### 5. 不对称传输

当前虽然没有原因表明为什么不使用如 SMTP 或 MSMQ 等方法，在 cXML 规范中最常用的不对称传输消息是通过 HTTP（如可能）或 URL 格式的编码。

通过 HTTP 的单向传输，客户简单地 POST cXML 消息到服务器并在没有接到任何响应信息时结束这个连接。但是，在使用 URL 编码方式时，cXML 文档不能直接发送到服务器而是作为一个隐藏的 HTML 表单域，允许服务器 Web 页面显示一些面向用户的信息。更多关于这个传输方法的细节可参见 cXML 文档。

### 6. 基本和通用的元素

cXML 实体调整的一些标准如下：

- isoLangCode 表示 ISO Language Code 639。
- unitOfMeasure 实体遵循 UN/CEFACT 建议。
- HTTP/1.1 标准用于进行 URL 定义。

### 7. cXML 定义

cXML 规范定义了许多标准文档，按下列通常的标题进行定义。

- Order 定义。
- PunchOur 定义。
- Catalog 定义。
- 订阅管理定义。
- 消息恢复定义。

cXML 1.0 规范文档十分详细定义了所有的 cXML 文档，但是我们主要解释上述每个部分的对象。

### 8. Order 定义

它包含 OrderRequest 和 OrderResponse 文档，这里 OrderRequest 同购买定单相似，而 OrderResponse 同供应商对购买定单的响应类似。

### 9. PunchOur 定义

cXML 中的 PunchOur 文档允许消费者从远程目录中查看以及向其当前的购物车中添加商品项

目而不失去其状态。PunchOutSetUp文档允许用户向远程系统定义他自己并能重定向当前的会话到一个远程浏览会话上。

这种方式有效地允许我们支持合作伙伴发布目录而不用对每个系统都下购买定单。

#### 10. Catalog定义

cXML的Catalog定义包含三个元素：Supplier、Index和Contract，以描述在一个购物系统中持续的或保持的数据。买者使用 Supplier元素获得更多关于供应商的信息，如联系地址和定单细节。

Index元素描述供应商的商品和服务信息。最后，Contract元素描述客户和供应商之间的合同形式，如每个特定项目的价格。

#### 11. 订阅管理定义

订阅管理定义允许一个中介或第三方管理供应商的信息以及供应商和消费者之间目录内容。

#### 12. 消息恢复定义

消息恢复定义是一个十分有趣的定义，该部分描述了在一个系统当服务器不能直接接受消息时如何能够查询消息，不能直接接受消息的原因可能由于防火墙或代理服务器。

#### 13. cXML定义文档

作为上面定义的补充，还有一系列的文档包含着 cXML标准。它们在cXML站点可以下载并都使用.mod作为文件名后缀。它们规范了如何开发和发布与 cXML 1.0兼容的应用程序。

这些文档按照它们的目的简列如下。更多的信息请参考 cXML文档。

- cxml.dtd作为所有cXML文档构件基础的DTD描述。
- Base.mod定义基本的元素用于构建更高层的 cXML。
- Common.mod在cXML定义中通常使用的类型。
- Contact.mod供应商和项目列表之间的联系。
- Index.mod允许我们修改正在被系统处理的商品和 /或服务列表。
- Item.mod包含一个项目的消息信息。
- Pending.mod在响应中返回的数据元素。它们遵循完全的 cXML消息格式并通过请求 /响应机制进行传输。
- Subscription.mod显示在一个客户的订购内容中的一些修改。既然是一个消息，它可以在任何时候到达——首先需要发送不明示的请求。
- Supplier.mod供应商品和服务的供应商。包括供应商 ID列表以定义供应商。
- Transaction.mod定义在cXML中的交易元素。
- Transport.mod描述在cXML中的封装传输。

#### 14. DoD的CatXML

美国国防部的联合电子商务程序办公室 ( JECPO )，为了支持DoD的部分供应商，开发了一套CatXML用于分布式目录查询和供应商请求响应的无缝电子交换。它是大的组织使用 cXML的例子，其关键是 CatXML设计。CatXML定义了供应商如何使其在线并通过合作目录销售商品和服务。合作目录是基于高扩展性结构的当前的目录，允许供应商将他们的产品销售到一个广大

的领域，如一些门户网站。JECPO开发CatXML的目标是定义一个交换规范以使供应商很容易地集成到现代电子商务机制中去。

具有在线电子目录的公司适合加入 DoD的EMall系统，公司将收到一张空白的购买协议允许公司向DoD的EMall客户明示其产品和服务。客户就可以通过使用 VISA或美国政府购买卡从EMall中下定单了。

我们通过一个开发用于纵向行业的语言的例子对 cXML进行了总结。下面我们将看看横向行业。

## 12.9 第1步——横向行业

目前，精力主要集中在应用 EDI、XML、合作容器以及代理技术创建 XML/EDI结构以简化处理过程。这个部分将展现给你的是电子商务应用程序如何使用这些结构。

### 12.9.1 ASC X12

在1999年X12会议的最后三个月，成立了一个专门的 XML建议委员会，目的是发布认为可能简化的基于 X12标准的XML规范，并能在 X12标准的各个层次运行。该工作需要和技术报告类型1：X12-XML:Representation of X12 Semantics in XML Syntax (X12-XML：在XML表达式中表示 X12语义)进行扩展。该报告来自 X12C下属的关于 XML的委员会叫做 X12CTG3。

更多的信息请浏览：<http://www.disa.org/x12/x12c/X12CTG3/x12ctg3.htm>

X12CTG3子委员会的目标是保留 X12标准丰富的语义——而不是简单使用 XML复制X12标准。这样X12的实现者会进行进一步的实验，如数据转化。在定义一个方法时，处理过程表明将X12会话简单转化成XML/EDI并不是最有效的解决方案，而任何工作还要满足X12的下一个版本。结果简单的转换算法在寻找中失败了。X12CTG3获得的经验给X12提交了一系列的课题，希望X12能够采用以使X12标准更有前途。

例如，在一个X12消息中太多的语义隐含在代码列表中。例如在一个 X12消息的MEA（措施字段）给出了一个重量——但是什么的重量呢？如果它的前面有一个 HL（继承循环）字段HL03的元素的内容会告诉应用程序它是运输或载货汽车或盒子的重量。

在将什么移到XML/EDI方面做了大量的工作，方法很有意义。在定义特定的媒体数据容器或附加的路由和控制机制时产生了一点争议。一般认为这些领域是不成熟的需要改进。

下面给出的例子给出了一个XML表示的850购物定单交易集技术报告，为了清楚和简明的目的而进行了一些删减。下面是X12的设计：

程序清单 12-41

```
BEG*00*SA*1513339**19990621~  
N1*ST**11*001786664W101~  
PO1**10*CA*24.48**UI*XYZ05879000032~  
PO4*72~
```

用XML表示为：

## 程序清单 12-42

```

<PurchaseOrder X12:PROMOTEDQUAL="Original" >
<Header>
    <BEG-BegSegPurchaseOrder>
        ...
    </BEG-BegSegPurchaseOrder >

    <Loop-H310-N1-Name X12:PROMOTEDQUAL="ShipTo">
        ...
    </Loop-H310-N1-Name >

    <Detail>
        < Loop-PO1-BaselineItemData >
            ...
        </Loop-PO1-BaselineItemData >

        < Loop-PO1-BaselineItemData >
            ...
        </Loop-PO1-BaselineItemData >
    </Detail>
</Header>

< Summary>
    < CTT-TransTotals>
        ...
    < AMT-MonetaryAmt>
        ...
</Summary>

```

那么另一个主要的EDI语言UN/EDIFACT会是怎样的呢？

## 12.9.2 XML-EDIFACT

XML-EDIFACT的努力是将来自 UN/EDIFACT的不同方法转换成 XML/EDI。努力的结果是一个自由的对等模块：XML::Edifact带有GNU一般许可证，能够将任何具有格式正规的UN/EDIFACT消息转化成人可读的XML或是相反过程。方法是将原始 UN/EDIFACT目录中的词作为标记并在该命名空间定义文档。

下面是一个消息片段：

## 程序清单 12-43

```

<?xml version="1.0"?>
<!DOCTYPE teleord:message
    SYSTEM "http://www.xml-edifact.org/LIB/xml-edifact-03/teleord.dtd">

<!-- XML message produced by edi2xml.pl (c) Kraehe@Bakunin.North.De -->

<teleord:message
    xmlns:teleord='http://www.xml-edifact.org/LIB/xml-edifact-03/teleord.rdf'
    xmlns:trsd='http://www.xml-edifact.org/LIB/xml-edifact-03/trsd.rdf'
    xmlns:trcd='http://www.xml-edifact.org/LIB/xml-edifact-03/trcd.rdf'
    xmlns:tred='http://www.xml-edifact.org/LIB/xml-edifact-03/tred.rdf'
    xmlns:uncl='http://www.xml-edifact.org/LIB/xml-edifact-03/uncl.rdf'

```

```

xmlns:anxs='http://www.xml-edifact.org/LIB/xml-edifact-03/anxe.rdf'
xmlns:anxc='http://www.xml-edifact.org/LIB/xml-edifact-03/anxc.rdf'
xmlns:anxe='http://www.xml-edifact.org/LIB/xml-edifact-03/anxe.rdf'
xmlns:unsl='http://www.xml-edifact.org/LIB/xml-edifact-03/unsl.rdf'
xmlns:unknown='http://www.xml-edifact.org/LIB/xml-edifact-03/unknown.rdf' >

<!-- SEGMENT LIN+16 -->
<trsd:line.item>
  <tred:line.item.number>16</tred:line.item.number>
</trsd:line.item>

<!-- SEGMENT PIA+5+1861081383:IB -->
<trsd:additional.product.id>
  <tred:product.id.function.qualifier uncl:code="4347:5">Product identification
</tred:product.id.function.qualifier>
  <trcd:item.number.identification>
    <tred:item.number>1861081383</tred:item.number>
    <tred:item.number.type.coded uncl:code="7143:IB">ISBN (International
      Standard Book Number)</tred:item.number.type.coded>
  </trcd:item.number.identification>
</trsd:additional.product.id>

<!-- SEGMENT IMD+F+BPU+:::CASS -->
<teleord:item.description>
  <tred:item.description.type.coded uncl:code="7077:F">Free-form
</tred:item.description.type.coded>
  <teleord:item.characteristic.coded teleord:code="7081:BPU">Book Publisher
</teleord:item.characteristic.coded>
  <trcd:item.description>
    <tred:item.description>CASS</tred:item.description>
  </trcd:item.description>
</teleord:item.description>

<!-- SEGMENT QTY+21:3 -->
<trsd:quantity>
  <trcd:quantity.details>
    <tred:quantity.qualifier uncl:code="6063:21">Ordered quantity
  </tred:quantity.qualifier>
    <tred:quantity>3</tred:quantity>
  </trcd:quantity.details>
</trsd:quantity>

<!-- SEGMENT PRI+AAA:12.5:SR:DPR::LBR -->
<trsd:price.details>
  <trcd:price.information>
    <tred:price.qualifier uncl:code="5125:AAA">Calculation net
  </tred:price.qualifier>
    <tred:price>12.5</tred:price>
    <tred:price.type.coded uncl:code="5375:SR">Suggested retail
  </tred:price.type.coded>
    <tred:price.type.coded uncl:code="5387:DPR">DiscountPrice
  </tred:price.type.coded>
    <tred:measure.unit.qualifier>LBR</tred:measure.unit.qualifier>
  </trcd:price.information>
</trsd:price.details>

</teleord:message>

```



更多的信息请浏览：<http://www.xml-edifact.org>

### 12.9.3 电子商务XML工作组

1999年12月初，联合国贸易便利和电子商务（UN/CEFACT）组织以及高级结构化信息标准组织（OASIS）合作18个月后开始启动一个世界范围的项目以标准化XML商务规范。UN/CEFACT和OASIS建立了一个电子商务XML工作组开发技术框架以允许XML作为一个一致性的工具交换所有电子商务数据。电子商务XML工作组（ebXML）的工作成果将被放在XML.org和UNECE/CEFACT网站的开放域中。

除了“市场、知觉和教育”小组，下面是一些技术项目小组：

ebXML要求——短期和长期项目目标。

商务处理方法——框架和互操作能力；模块交互（UML到XML模式或相反过程）和ebXML的设计模式。IBM的XMI工具包（可以从<http://www.alphaworks.ibm.com/tech/xmitoolkit/>站点获得）支持Java、Rational Rose、UML模块之间的共享和转化，同时包括一个进行XML1.0文件读写API。该工具包是该领域的一个出色的领先产品。

传输/路由和打包——对路由信息、安全、保证的信息转发、批处理、一个集合中的相对消息、服务品质等进行封装。交付的是电子商务的技术规则和指导方针。

核心组件——在一些统一的方面使用现存的语义，重用/组合现存的元素/组件，扩展现存的消息以包含新的应用，定义核心和横向的数据语义。交付的是ebXML和EDIFACT工作组D组的消息设计规则。

技术结构——在数据元素、语义转换和国际化之间进行语义等价。交付的包括ebXML和EDIFACT工作组D组印象设计规则和将EDIFACT消息转化成XML消息的指导方针。

注册器和容器——版本、开发工具的工具、容器的容器、容器的工具和工具的容器等。交付的是注册器结构、实际的容器和交互性指导方针。

技术协调和支持——维护ebXML网站、认证、测试环境、开发商起步工具以及发布软件兼容性标准。

确认会议的参加者来自其他XML创始机构、行业组织、标准组织、供应商联合会以及独立的企业。如此混杂的参与者是必需的，如此我们才能获得在交互性的所有等级达成一致的机会。

在前面描述了横向行业的实现方案后，我们将详细看看另一个工作：Microsoft的BizTalk。

### 12.10 第一个横向步骤

BizTalk框架起源于Microsoft，同跨越广泛行业的许多主要组织协同工作，目的是指导创建和维护XML数据模式以允许电子商务和应用的集成。这些模式位于<http://www.BizTalk.org>站点的BizTalk库中，对于那些允许他们的系统通过使用相同的模式同其他系统进行集成的任何人来说，他们可以随时使用这些模式。这些模式是完全只读的，这样一旦你开始使用一个特定的模式以集成你的系统，该模式不会被改变——虽然版本升级时模式会进行升级。

目标是为每个行业创建一个可利用的模式库，在最小的离线通信基础上获得最大限度的重用性和互操作性。如果贸易伙伴遵循库中的模式那么他们应该能够交换消息。理论上，一个未

来的贸易伙伴为了同一个组织进行数据交换只要应用该组织发布到库中的模式就可以了。而且两个伙伴之间不需要进行前期的沟通。实际上，一些前期的离线通信还是需要的以建立贸易协定。这个库提供了现实的利益。它要求贸易伙伴使用模式要求的消息进行通信，并帮助未来的贸易伙伴在不用长期协调的基础上就可以交换文档。

同贸易的努力结果（如 EDI）不同的是，BizTalk 模式并没有努力向贸易伙伴强加商业语义。但它对标记进行了限制，他们对促进数据交换也就进行了这些工作。BizTalk 消息的核心内容留给了贸易伙伴。这样允许 BizTalk 模式的设计者在商业要求的各个等级自由的定义用于交换的数据。

BizTalk 库已经包含面向低层交换的模式，如个人购物定单，以及高层的面向商务的内容如建筑结构。在库中的每个模式都同一个例子消息和人可读的文档相关联。这样未来的贸易伙伴就可以根据新的模式准备他们的软件而不用咨询模式提供者。另外作为一个模式容器，BizTalk.org 提供了一个机制使你可以在一个模式中注册兴趣的领域。这个功能有两个目的：你可以在模式改变时得到通知，以及给容器的浏览人员一个有用的建议，以决定是否接受这个模式。

现在我们纵览了 BizTalk，先来看看其规范。

### 12.10.1 BizTalk 标记规范

在写作本书时，BizTalk 框架标记规范的版本号是 1.0。粘贴到库中的模式使用 Microsoft 的 XML 数据简化模式（XML-DR），它是一个用于模式的使用 XML 语法的 XML 精简版本。这个实现遵循了开始时的 W3C 的模式草案，该草案最早出现在 1990 年。从那以后，W3C 的模式草案得到发展并同 XML-DR 出现了分歧。但值得注意的是，BizTalk 小组在格式化 XML 模式规范使其更为明白，并成为 W3C 的一个建议。BizTalk 将使用这个标准。虽然用于 XML-DR 实现的模式被支持的时间不确定，但会提供工具以简化过渡过程。

规范主要包括以下方面：

- 文档结构。
- 文档认定。
- 遵循 BizTalk 模式。
- 文档路由信息。

### 12.10.2 规范的目标和当前的限制

BizTalk 规范的目标是允许企业、商户或开发商使用标签交换信息而不需使用标准化编程工具或分布式计算技术。它通过要求一个适用所有交换信息的最小标签集来实现这个目标。标签的设计便于定义和路由 BizTalk 消息。

嵌入在要求标签中的是元素。元素就像用户定义的 XML 文档的容器。组织可以设计文档类型以满足相关商务交换的需要。由于消息的设计是开放的，组织必须拥有一个向合作伙伴展现模式的手段——这就是在 BizTalk 站点上的消息模式公共库。它允许贸易伙伴在这个库中查询合作伙伴粘贴的模式。如果开发商希望在现存的模式上进行扩展，他们可以免费地获得这些模式，

```
<?xml version="1.0"?>
<bizTalk_1 xmlns:="urn:schemas-biztalk-org/biztalk_1.xml">
  <header>
    <delivery>
      <message>
        <messageID>12346</messageID>
        <sent>1999-12-31T23:59:59-05:00</sent>
        <subject>Generic Sample</subject>
      </message>
      <to>
        <address>http://www.generic_co.com/recv.asp</address>
        <state>
          <referenceID>1</referenceID>
          <handle>1</handle>
          <process>receiveProcess</process>
        </state>
      </to>
      <from>
        <address>
          http://www.bland_co.com/send.asp
        </address>
        <state>
          <referenceID>23</referenceID>
          <handle>2</handle>
          <process>orderProcess</process>
        </state>
      </from>
    </delivery>
  </header>
</bizTalk_1>
```

```
</delivery>
<manifest>
  <document>
    <name>bland_co_order</name>
    <description>Bland Sample Order Document</description>
  </document>
  <attachment>
    <index>1</index>
    <filename>whatIwant.jpg</filename>
    <description>picture of what I want</description>
    <type>jpg</type>
  </attachment>
</manifest>
</header>
<body >
  <bland_co_order
    xmlns:= "x-schema:http://www.bland_co.com/schemas/order.xml">
    -- Your XML document data goes here'--
  </bland_co_order>
</body>
</bizTalk_1>
```

这个例子乍看起来相当宏大。但是我们将看到，它包含着报头信息和你的消息。上面描述十分复杂的报头实际上对消息的组织者的影响甚少。 BizTalk的报头提供了许多可选的元素以向信息的接收者发送消息。这些元素中的值留下来由应用程序实现。上面的例子描述了对所有元素的使用。考虑到应用的需要，你的文档可能会很简单。

BizTalk标签规范的两个有趣的特征需要注意。首先，虽然 XML-DR模式支持的数据类型超过XML 1.0中的定义， BizTalk1.0模式并没有使用强的类型。 Microsoft的XML解析器MSXML支持同IE5一同发布的数据类型，但是该模式是完全平台独立的，保证了这些类型对非 Microsoft平台的可访问性。当然，模式实现本身是同平台相关的，但这是不可避免的。一旦 W3C发布一个XML模式建议，强的数据类型将被所有的 XML平台所接受。另一个有趣的特征是每一个 BizTalk的元素都没有属性同其关联，除了对命名空间的声明。在 BizTalk 1.0模式中除了命名空间以外的所有信息都是通过XML元素进行沟通的。

#### 1. BizTalk文档的根 ( root )

所有的BizTalk文档必须附加一个下面所示的根元素：

```
<bizTalk_1 xmlns:="urn:schemas-biztalk.org:bizTalk/biztalk_1.xml">
</bizTalk_1>
```

这个元素定义了BizTalk的命名空间和并指定文档遵循 BizTalk 1.0，使用的元素来自 BizTalk的命名空间。显示的命名空间描述符是使用周知的 URN来表示BizTalk 1.0，这样一个客户在需要时就可以根据该元素利用这个文档。在根元素中，消息头和主体元素应该显示。

#### 2. 文档报头

BizTalk兼容的消息包括文档根和文档主体。在进入文档主体前，让我们看看可选的报头元素。报头的目的是提供一个变化的信息以为路由、交付和处理该文档提供便利。如果使用报头元素，它只能出现一次。

报头元素可能包含表 12-5中所列的元素

表 12-5

元素	显示次数	目 的
delivery	0或1次	描述消息和其路由
manifest	0或1次	描述消息包含的内容和同消息关联的附件

delivery 元素通常对消息进行描述，它从哪里来，到哪里去。 manifest消息描述消息的内容以及相关的文件。在报头元素中包含的消息绝大部分用于发送和接收应用程序。许多在报头中包含的元素是框架要求保持的，它们向应用程序提供信息，但模式本身并不需要其服务。

manifest元素详细描述文档的内容。它提供用于命名文档的元素以及描述同消息关联的一个或多个非XML文件的元素。

#### (1) delivery元素

该元素定义了消息，谁发送了它，谁将接收它。下面是该元素的壳以及直接子元素：

程序清单 12-45

```
<delivery>
  <message>...</message>
  <to>...</to>
  <from>...</from>
</delivery>
```

通常，表12-6中所列元素允许在 delivery元素中使用。

表 12-6

元 素	显示次数	目 的
message	1次	提供描述符、时间戳以及人可读的消息标签
to	1次	提供源URL以及接收者的内部状态信息
from	1次	提供源URL以及发送者的内部状态信息

message元素对跟踪BizTalk消息队列是非常有用的。而该元素中包含的信息对框架本身是没有用处的。但是，它对构造包含在多个 BizTalk消息中的商务处理以及在 BizTalk处理中实现审计特征是有裨益的。表12-7中是message元素中包含的内容。

表 12-7

元 素	显示次数	目 的
messageID	1次	应用程序中的GUID设定的顺序消息编号
sent	1次	消息产生时创建的时间戳
subject	0或1次	描述消息类型的人可读标签

注意由于在 BizTalk框架中没有使用强的数据类型，sent元素的格式应该遵循XML-DR规定的 datetime.tz数据类型（ISO8601的子集），例如，YYYY-MM-DDTHH:MM:SS[+|-] HH:MM。下面是我们关于message元素的例子：

## 程序清单 12-46

```
<message>
  <messageID>12346</messageID>
  <sent>1999-12-31T23:59:59-05:00</sent>
  <subject>Generic Sample</subject>
</message>
```

在这个例子中，消息定义符是一个由消息产生者指定的一个顺序号。这对一系列 BizTalk 消息进行排序或开发审计程序提供了便利。这个消息在 1999 年 12 月 31 日午夜前 1 秒被发送，发送的时区为格林尾治标准时间后 5 小时（美国东海岸时间）。Subject 元素对 BizTalk 是无所谓的，用于在调试程序时描述消息的类型。

to 和 from 元素共享同样的内容模块。它包含发送（或接收）站点的 URL 以及一个 state 元素。state 元素提供三个条目，这样一个应用程序就能够使用它将一些内部状态信息同消息唯一关联。在获得细节前，先看看 to 和 from 元素的内容模块（参见表 12-8）。

表 12-8

元 素	显示次数	目 的
address	1 次	发送（或接收）站点的 URL
state	0 或 1 次	用于将发送或接收站点内部状态信息同消息关联的信息

address 元素是一个 URI。它必须能够表明消息的逻辑地址，但它不是一个 URL。如果 URI 是 trading-partner-supplies:purchasing，一个 BizTalk 服务器将会将其映射成为一个 URL 地址来描述这个贸易伙伴的购买处理进程。通常，发送站点将提供 URL。

当一个商务交换包含一系列相关 BizTalk 消息时就需要状态信息。在这种情况下，一个进程需要能够将一个特定的消息同一些内部数据相关联，如一个活动服务器页面（ASP）会话变量可以包含一个属性列表。规定一个通常的机制来完成这项工作是很困难的，但 BizTalk 在它的内容模块中提供了三个层次的规范（参见表 12-9）。

表 12-9

元 素	显示次数	目 的
referenceID	1 次	商务交换的唯一描述符
handle	0 或 1 次	提炼 referenceID 以指定交换的一个特定部分，例如在一个工作流程中的一步
process	0 或 1 次	提炼 handle 以提供一个处理上下文（context），例如一个安全上下文

现在我们对 delivery 元素有了全面的了解，让我们回顾下面的例子：

## 程序清单 12-47

```
<delivery>
  <message>
    <messageID>12346</messageID>
    <sent>1999-12-31T23:59:59-05:00</sent>
    <subject>Generic Sample</subject>
```



```

</message>
<to>
  <address>http://www.generic_co.com/recv.asp</address>
  <state>
    <referenceID>1</referenceID>
    <handle>1</handle>
    <process>receiveProcess</process>
  </state>
</to>
<from>
  <address>http://www.bland_co.com/send.asp</address>
  <state>
    <referenceID>23</referenceID>
    <handle>2</handle>
    <process>orderProcess</process>
  </state>
</from>
</delivery>

```

我们已经看完了 message 元素，下面让我们关注以下 to 和 from 元素。来自 delivery 元素的表明发送者消息是在 Bland 公司创建的 ASP 页面中进行产生，该页面位于 [http://www.bland\\_co.com/send.asp](http://www.bland_co.com/send.asp)。在处理中同消息关联的状态信息通过 orderProcess 表明，在交互的第 2 步，商务交换的唯一描述符为 23。这对接收者来说没有什么意义，但是可以被接收处理程序拷贝到任何响应消息中去。消息的接收是在 Generic 公司实现，处理页面是 [http://www.generic\\_co.com/recv.asp](http://www.generic_co.com/recv.asp)。这个消息是消息扩展的一部分，所以处理程序为我们保留了一些状态信息。referenceID、handle 和 process 元素允许接收者获得合适的本地数据以处理这个消息。

你可以看到，BizTalk 消息中 delivery 元素的复杂性依赖于你的商务交换的复杂性。简单的交换可以省略 state 元素。

## (2) manifest 元素

这个元素向一个接收应用程序提示传输的内容。该元素可选，当多个文档同该次交互相关时使用该元素。有两种类型的文档可以被传输：BizTalk 消息和非 XML 附件。所以该元素具有两个子元素（参见表 12-10）。

表 12-10

元 素	显示次数	目 的
document	0 或多次	描述包含在交互中的单一的 XML 消息
attachment	0 或多次	描述包含在交互中的单一非 BizTalk 附件

你可能在一次交互中在 <body> 里具有不止一个的 BizTalk 消息（就是说交互的不止一个实例）——如果商务工作流语义包含不止一个 XML 消息则必须放在 BizTalk 消息的 <body> 里。这样的一个 BizTalk 消息也被称作购物车文档（boxcar document）。我们的例子不是一个购物车文档。但是如果我们的例子是为了订购一个商务会议的供应品，那么将所有的相关购买定单放在一个 BizTalk 消息中就太合适了。在这种情况下，我们就可以使用一个购物车文档通过多个 document 元素包含多个购买定单。表 12-11 中是这个元素的内容模块。

表 12-11

元 素	显示次数	目 的
name	1次	根中document元素的名称
description	0或1次	对document人可读的描述

相反，一个附件是一个并不包含在 BizTalk文档主体中的关联的文件。它对二进制信息是很有用的，如图像或具有一定格式的数据文件。多方 MIME传输通常使用附件发送文档。表 12-12 是attachment元素的内容模块：

表 12-12

元 素	显示次数	目 的
index	1次	附件定义符
filename	1次	附属文件的文件名
description	0或多次	对附件的人可读描述
type	0或多次	定义文档类型的关键字

下面让我们看看例子中的 manifest元素：

程序清单 12-48

```
<manifest>
  <document>
    <name>bland_co_order</name>
    <description>Bland Sample Order Document</description>
  </document>
  <attachment>
    <index>1</index>
    <filename>whatIwant.jpg</filename>
    <description>picture of what I want</description>
    <type>jpg</type>
  </attachment>
</manifest>
```

在这个例子中，Bland公司在 BizTalk消息的主体中发送了一个定单文档。该文档有一个根元素叫做bland\_co\_order。它有一个附件，一个 JPEG文件叫做 whatIwant.jpg。

### 3. 文档主体

文档的主体是必须的。它是模式设计者参与设计的部分。在没有现成模式时，你需要为你的商务处理流程开发模式；如果现成的模式不合适，需要改造它们。所写的 XML文档应当遵循这些模式并作为body元素的子元素出现。报头元素需要同一些相关的数据描述一起到达目的地，而body元素则是发送消息的原因。body元素具有一个或多个子元素。每个子元素都是根据 BizTalk模式书写的文档。一个购物车文档有超过一个这样的子文档。开发用于交互的模式决定了这些元素的内容。

下面是我们例子的主体：

程序清单 12-49

```
<body >
  <bland_co_order
```

```

xmlns:= "x-schema:http://www.bland_co.com/schemas/order.xml">
-- Your XML document data goes here --
</bland_co_order>
</body>

```

我们的例子中包含了单一的文档，通过使用 bland\_co\_order元素进行标记。我们已经声明了一个命名空间以允许 BizTalk使主体可用。通常，你希望到 BizTalk的库中查找这个模式，但 Bland公司选择在他自己的主机上安装这个模式。

#### 12.10.4 一个用于Wrox书店的BizTalk文档

现在我们将定义一个XML文档的例子，该例子遵循 BizTalk框架版本0.81规则。我们将首先定义一个遵循XML数据精简模式标准（XML-DR）的文档模式，该模式可以被存储在 BizTalk库中。我们将使用该模式传送关于书的数据。

这个模式能够被用来为客户在电子商务网站或一个远程服务器请求关于特定书目的信息，同时，也允许管理员或一个远程服务器去修改或提交关于书的信息。该模式显示如下：

程序清单 12-50

```

<?xml version="1.0"?>
<Schema name="BookInfo"
  xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">

  <!--A book has a cost defined by both a currency and price-->
  <ElementType name="Currency" content="textOnly" model="open" dt:type="string"/>
  <ElementType name="Price" content="textOnly" model="open" dt:type="number"/>
  <ElementType name="BookCost" content="eltOnly" model="closed">
    <element type="Currency"/>
    <element type="Price"/>
  </ElementType>

  <!--Information about the book-->
  <ElementType name="Title" content="textOnly" model="open" dt:type="string"/>
  <ElementType name="Author" content="textOnly" model="open" dt:type="string"/>
  <ElementType name="ISBN" content="textOnly" model="open" dt:type="number"/>
  <ElementType name="Subject" content="textOnly" model="open" dt:type="string"/>
  <ElementType name="PubDate" content="textOnly" model="open" dt:type="date"/>
  <ElementType name="RefURL" content="textOnly" model="open" dt:type="uri"/>
  <ElementType name="TitleData" content="eltOnly" model="open">
    <element type="Title"/>
    <element type="Author" minOccurs="1" maxOccurs="*" />
    <element type="ISBN"/>
    <element type="Subject"/>
    <element type="PubDate"/>
    <element type="RefURL"/>
  </ElementType>

  <!--General Information about the book-->
  <ElementType name="BookNumber" content="textOnly" model="open" dt:type="string"/>
  <ElementType name="Summary" content="textOnly" model="open" dt:type="string"/>
  <ElementType name="BookInfo" content="eltOnly" model="open">
    <element type="BookNumber"/>

```

```
<element type="Description"/>
<element type="TitleData"/>
<element type="BookCost"/>
</ElementType>

</Schema>
```

让我们仔细看看这个模式。所有的 BizTalk 模式文档必须使用当前的 XML-DR 模式草案，所以我们将命名空间设为 Microsoft 的 XML-DR 实现，这样频繁使用的模式数据类型将贯穿整个 BizTalk 文档。

#### 程序清单 12-51

```
<?xml version="1.0"?>
<Schema name="BookInfo"
  xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
```

由于 BizTalk 规范没有数据定义，所以在消息主体中没有防止数据定义的手段。

我们发布的文档需要允许提供书的成本，这样客户和零售商可以决定它的价格。我们首先定义了一个字符串表示所使用的货币——这意味着可能有一个货币表示的菜单。然后使用了一个数字类型的元素表示书的价格。

我们定义了一个元素类型叫做 <BookCost>，它包含 <Currency> 和 <Price> 两个元素，这表明当 BookCost 元素的模型设为 closed 时，它们是该元素下所有可以获得的元素。

#### 程序清单 12-52

```
<!--A book has a cost defined by both a currency and price-->
<ElementType name="Currency" content="textOnly" model="open" dt:type="string"/>
<ElementType name="Price" content="textOnly" model="open" dt:type="number"/>
<ElementType name="BookCost" content="eltOnly" model="closed">
  <element type="Currency"/>
  <element type="Price"/>
</ElementType>
```

模式中另一个重要的部分是定义书的详细信息。我们定义了一个名为 <Title> 的元素类型来表示书名，以及 <Author>、<ISBN> 和 <Subject> 元素，它们的用法不言自明。

#### 程序清单 12-53

```
<!--Information about the book-->
<ElementType name="Title" content="textOnly" model="open" dt:type="string"/>
<ElementType name="Author" content="textOnly" model="open" dt:type="string"/>
<ElementType name="ISBN" content="textOnly" model="open" dt:type="number"/>
<ElementType name="Subject" content="textOnly" model="open" dt:type="string"/>
```

发布日期的元素由日期模式数据类型声明，其采用的是 ISO 8601 格式。这意味着指定的日期将是

YYYY-MM-DD 格式。

```
<ElementType name="PubDate" content="textOnly" model="open" dt:type="date"/>
```

通常在定义一本书时附加参考的 URL，我们使用<RefURL>元素来实现，将其定义为一个数据类型 uri——统一资源定义符。

```
<ElementType name="RefURL" content="textOnly" model="open" dt:type="uri"/>
```

我们接下来定义了一个父元素 <TitleData>来包含这些元素。注意一本书可能有多个作者，所以允许使用多个< Author >元素。

#### 程序清单 12-54

```
<ElementType name="TitleData" content="eltOnly" model="open">
  <element type="Title"/>
  <element type="Author" minOccurs="1" maxOccurs="*" />
  <element type="ISBN"/>
  <element type="Subject"/>
  <element type="PubDate"/>
  <element type="RefURL"/>
</ElementType>
```

为了完成我们的文档模式，定义了一个根元素叫做 <BookInfo>，它被定义包含前面所说的所有元素，以及两个新的元素叫做 <BookNumber>和<Summary>，前者维护书的内部 ID，后者包含一些书的内容概括。

#### 程序清单 12-55

```
<!--General Information about the book-->
<ElementType name="BookNumber" content="textOnly" model="open" dt:type="string"/>
<ElementType name="Summary" content="textOnly" model="open" dt:type="string"/>
<ElementType name="BookInfo" content="eltOnly" model="open">
  <element type="BookNumber"/>
  <element type="Summary"/>
  <element type="TitleData"/>
  <element type="BookCost"/>
</ElementType>
</Schema>
```

这样，一个遵循该模式的 XML 文档实例就可以出来了：

#### 程序清单 12-56

```
<?xml version="1.0"?>
<BookInfo xmlns="x-schema:books.xml">
  <BookNumber>423423</BookNumber>
  <Summary>
    This book details Site Server Commerce 3.0 and can be used by beginners and
    experienced developers alike.
  </Summary>
  <TitleData>
    <Title>Professional Site Server Commerce 3.0</Title>
    <Author>Marco Tabini</Author>
    <Author>Steven Livingstone</Author>
    <ISBN>983479387</ISBN>
    <Subject>Commerce</Subject>
    <PubDate>1999-11-20</PubDate>
    <RefURL>http://www.wrox.com</RefURL>
  </TitleData>
```

```

<BookCost>
  <Currency>US$</Currency>
  <Price>59.76</Price>
</BookCost>
</BookInfo>

```

为了能够使用定义在 BizTalk 模式中的规则传输我们的文档，我们必须在实例上加入一些附加的标签，这些标签我们在前面已经讨论过。BizTalk 的实例显示如下。注意该实例是如何将我们的文档包含在 BizTalk 标签中以及如何定义路由细节。

#### 程序清单 12-57

```

<biztalk_1 xmlns="urn:schemas-biztalk-org:biztalk/biztalk-1.0.xml">
<header>
  <header>
    <delivery>
      <message>
        <messageID>1</messageID>
        <sent>1999-12-31T18:24:00-05:00</sent>
        <subject>Book Information</subject>
      </message>
      <to>
        <address>http://www.deltabiz.com/biztalk/book.asp</address>
      </to>
      <from>
        <address>http://www.citix.com/biztalk/info.asp</address>
      </from>
    </delivery>
    <manifest>
      <document>
        <name>BookInfo</name>
        <description>Book Summary Information</description>
      </document>
    </manifest>
  </header>
</header>
<body>
  <BookInfo xmlns="urn:schemas-biztalk.org:deltabiz.com/books.xml">
    <BookNumber>423423</BookNumber>
    <Summary>
      This book details Site Server Commerce 3.0 and can be used by
      beginners and experienced developers alike.
    </Summary>
    <TitleData>
      <Title>Professional Site Server Commerce 3.0</Title>
      <Author>Marco Tabini</Author>
      <Author>Steven Livingstone</Author>
      <ISBN>983479387</ISBN>
      <Subject>Commerce</Subject>
      <PubDate>1999-11-20</PubDate>
      <RefURL>http://www.wrox.com/</RefURL>
    </TitleData>
    <BookCost>
      <Currency>US$</Currency>
      <Price>59.76</Price>
    </BookCost>
  </BookInfo>
</body>
</biztalk_1>

```



```
</BookInfo>
</body>
</biztalk_1>
```

注意，我们忽略了包含在 to 和 from 元素中的状态元素。这是一个相当简单的例子，只是对交易请求的信息响应，这样就没有交互的状态信息。

#### 使用 BizTalk 实现的解决方案

当前，同 BizTalk 模式（在服务器端）一起工作的 BizTalk 服务器正在开发，这样就发布了一个 Jumpstart 工具包以允许同 BizTalk 文档协同工作，以感受这种共享模式背后的强大力量。当前支持 COM 和 Perl 实现方式。

该工具包当前是早期版本，这样你会在使用时遇到不少问题。但使用它会获得有用的经验。你需要在工作站上安装特定的软件，并使用提供的安装程序进行安装。由于尺寸和复杂度（需要好几个章节才能将其讲明白），我们不需要运行整个工具包，但对它组件的概要介绍将会给你一些好的认识。

Jumpstart 工具包可以自由下载：<http://www.biztalk.org/Resources/tools.asp>。这个压缩文件包括对该工具安装的介绍以及要求预装的所有软件列表。

### 12.10.5 BizTalk Jumpstart 工具包简介

模式是一个简单的在两个或更多程序间的界面规范。模式的实现要求一个服务器作为程序间的媒介。这个服务器需要从一个程序接收消息，根据框架的标签检查它的内容，并将消息交付给接收程序。如果模式没有使用这样一个中间服务器，我们将返回到当前的事务状态，这时所有的集成必须被嵌入到参与程序的代码中。Microsoft 事实上正在开发这样的一个服务器，叫做 Microsoft BizTalk 服务器。该服务器在本书的写作时（1999 年冬天）是基于 Windows 平台的。既然 BizTalk 模式在发布 BizTalk 服务器前运行得很好，开发商需要一些软件以练习使用该模式。这个软件就是 Microsoft 的 BizTalk Jumpstart 工具包。在这个工具包中，开发商写的应用程序可以将应用程序连接到 Jumpstart 服务器软件。

虽然 BizTalk 服务器将要求运行在 Windows Microsoft 2000 之上，Jumpstart 工具包可以运行在 Windows NT4 上。

该工具包包含下列组件：

- 选择器（selector）——一个 COM 组件，接收消息并将其路由到相应的应用程序适配器。
- 计时工具（Timer Utility）——让开发商测试顺序消息的不对称提交的程序。
- 一致性工具（Persistence Utility）——一个维护状态信息的工具。
- 插入产生器——一个 VB 的插件，可以阅读框架并产生一个商务对象，向程序员隐藏 XML DOM 操作细节。
- 消息组件——一个 COM 组件，向程序员隐藏 XML DOM 构造 BizTalk 标签的细节。
- 应用适配器项目类型——一个项目软件帮助程序员所写组件同 BizTalk Jumpstart 组件可以集成在一个应用。
- 命名空间服务器——一个类似消息类型目录的应用程序，将消息名同处理消息的应用适配

器相匹配。

- 属性管理器——一个 Jumpstart 工具包的确认工具。

应用程序使用消息组件和插入产生器产生的类来创建 BizTalk 消息。应用接下来使用一个适配器将消息传送到选择器。选择器对消息进行检查并决定其类型。消息类型将作为一个键值存放到由命名空间服务器维护的表中，以定义消息的接收方。决定了接收方的选择器调用相应的适配器将消息发送到接收程序。该程序使用组件阅读消息。但是一些组件——注意是选择器、命名空间服务器和属性管理器——代表同样的 BizTalk 服务器运行组件，对于 BizTalk 服务器用户是没有用处的。时间工具是一个测试工具，同 Jumpstart 工具包联系紧密。一致性工具可能会有用，但是一个行业应用将会使用其他方法。这个流程对一个稳定性、产品化的 BizTalk 服务器是一个轻量级的版本。Jumpstart 组件已经足够产生 BizTalk 集成项目的原形，但不能用在生产目的。但是 Jumpstart 工具包允许组织在 BizTalk 服务器正在开发时准备要完成项目。

### 12.10.6 BizTalk 服务器

Microsoft BizTalk 服务器是微软基于 Windows 平台的 BizTalk 模式实现。它是一个帮助企业应用集成的产品，它通过使用通常的互操作协议如 HTTP 来交换结构化的商务文档。同模式类似，BizTalk 服务器依赖于在应用间结构化的商务文档的流动。BizTalk 服务器包含以下任务：

- 定义结构化的消息格式。
- 定义不同格式之间的映射。
- 组织伙伴和工作流的协议。
- 管理和确认服务器进程。

BizTalk 服务器使用规范和协议并使用适当的协议在伙伴间路由文档。BizTalk 有一个 API，但服务器同样能够运行没有限制的应用逻辑。同 BizTalk 服务器协同工作对程序员最大的好处是，许多对两个应用的集成工作通过确认可以由服务器进行。这是对当前状态的一个巨大进步。

#### 1. BizTalk 编辑器

使用 BizTalk 进行应用集成的关键是交换结构化的消息。无论这些消息使用 XML 或其他可以解释的文本模式，程序员必须能够规范这些消息的结构。这就是 BizTalk 编辑器的功能。该编辑器是一个图形化的工具，使用三层结构来制定消息规范文件，或简单的规范。数据库术语影响着 BizTalk 编辑器，这是考虑到用于组成消息的大量信息来自数据库。

程序员对记录 and 域进行操作。一个记录代表一些对象或实体，而域则是对象的属性。在一个数据库风格的表结构中该方法工作得很好，这里行是独立的对象，其域段的值描述了对对象的属性。但同关系表不同的是，一个规范可以在记录中包含其他记录。这就允许我们描述父子关系，而关系数据库则通过使用数据库连接来表示。在 BizTalk 编辑器中表示一个规范的树同样同文件目录结构相似，只是记录替代了目录而域段替代了文件（参见图 12-19）。

在写一个基于 XML 的规范时，程序员可以使用基本的 XML 数据类型以及在 XML-DR 模式中获得的类型来描述记录 and 域段。所有在一个模式中可选的类型对规范产生器都是适用的。BizTalk 编辑器允许程序员充分利用 XML 的优点而不用时自己成为一个 XML 专家。由于编辑器允许使用不同于 XML 的格式，界面被仔细设计以使界面中不会充斥着 XML 术语。

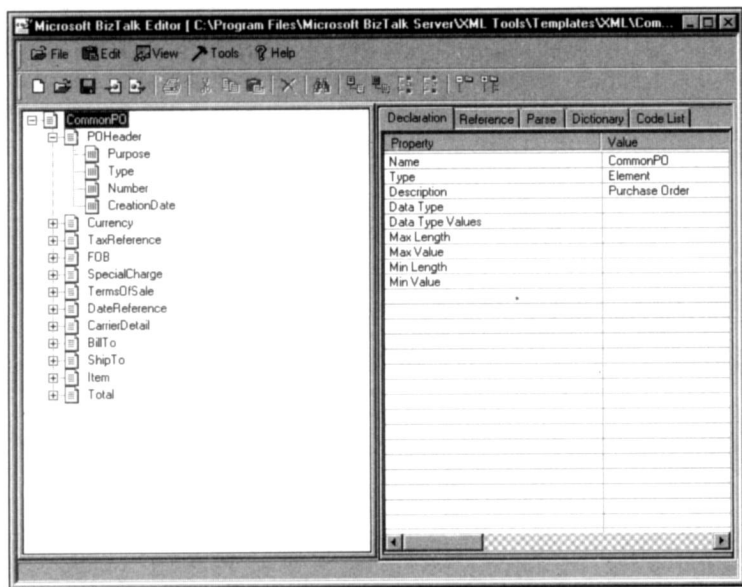


图 12-19

程序员在开始定义规范时不需要从零开始。系统初始时就有将 EDI商务消息格式和 HL7医药消息格式转换成 XML 的功能。BizTalk 服务器包含一个基本的 XML 商务常用消息规范模板集合，如购物定单和发票。ADO，主流的 Microsoft 数据库访问技术，从版本 2.0 以后允许数据库记录集可被保存为 XML。程序员可以使用这个特征在现存的数据库表格模式中模块化他们的规范。

许多逻辑上的消息格式在格式上同 XML 不同。EDI 格式、X12 和 EDIFACT 都使用有限的消息格式。构建主模式系统的消息经常使用有限的特征进行书写或有固定的位置格式。BizTalk 编辑器支持这些格式的使用。开始时，BizTalk 中包含了 X12 和 EDIFACT 的规范。程序员可以装入这些规范并按照需要对它们进行修改。更为重要的是，BizTalk 编辑器允许程序员指定所有可能的表现和位置以满足特定的格式。这个信息被 BizTalk 服务器保留和使用以对获得的消息进行翻译。

## 2. BizTalk 映射器

将你自己的消息格式映射为合作伙伴的消息格式在应用集成中是一个关键的任务。BizTalk 映射器工具支持程序员处理这个任务。它是一个图形化的编辑器，装入两个规范并允许程序员指定怎样将记录和域映射成另一个规范中的记录和域，它们可以是简单的一对一关系、一对多关系或多对一关系。或是对所包含的功能或基于脚本的会话进行映射。

界面中包含两个对应两个规范的面板和两者之间一个表示映射的映射网格（参见图 12-20）。左边的规范面板中包含的是源规范。它代表需要映射到新的格式的收到的信息。新的格式在目的规范中指定，在右边的规范面板中指定。映射网格表示关系以及执行这个映射需要的任何中间的处理要求。简单的关系通过拖拉在一个规范中的一个记录或域到另一个规范中的一个记录或域即可实现。在这个情况下，BizTalk 服务器执行从一个获得的消息到相关的本地消息类型的转换，只要通过拷贝获得消息的记录或域的内容到映射规范中的记录或域中即可。但有时没有这种简单的转换。有些内容需连接以形成新的域的内容，或需要对源域中的内容进行处理才能

转换成目的域中的内容。

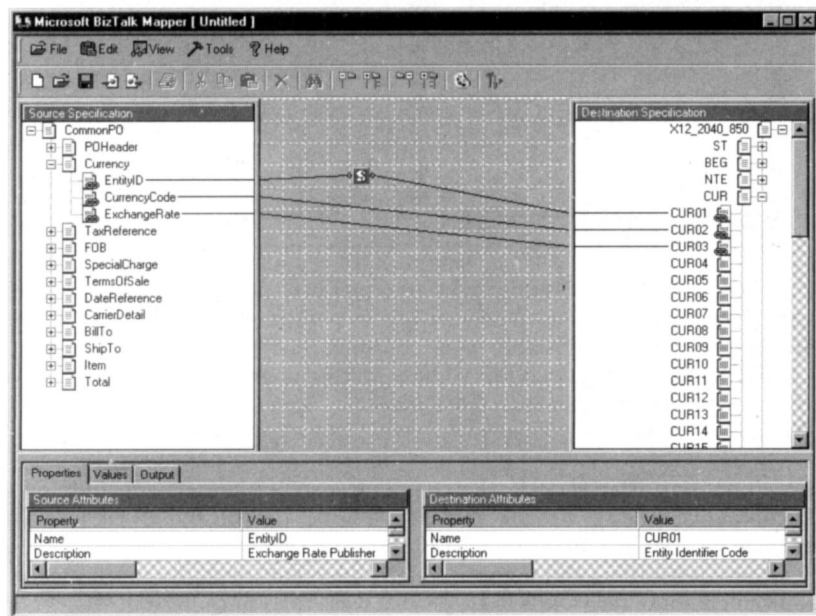


图 12-20

中间的处理在进行两种规范之间转化时需要进行。通过 BizTalk 应用程序的预定义功能或操作员的布置来实现。用户可以通过使用脚本提供更为高级的处理，叫做“functoid”。functoid 在 XSL 风格的表单中作为脚本功能实现，以在映射中将源域段转化成目的域段。

当映射完成后，程序员使用映射器进行编译。这时会获得一个 XSL 风格的表单。运行服务器进程利用这个表单对一个接收消息的实例进行转换以产生符合目的规范的消息实例。

### 3. 配置 workflow

除了规范设计和映射外，BizTalk 风格的集成商必须配置可使 BizTalk 工作的合作协议。这件工作在制定消息格式时进行，明确服务器端从特定源地址获得一个消息时应该进行哪些操作以发往目的地址。使用的工具叫做 BizDesk，使用户可以在定义的工作流程上对商务处理进行管理。

BizTalk 服务器使用设计者指定的协议在运行时路由消息。服务器在消息和其目的地之间作为一个媒介。应用程序可能通过使用一个 COM API 或指定一个周知的协议来同 BizTalk 进行集成。在后一种情况下，BizTalk 的组件将监视协议以从逻辑应用中获得消息来源信息。这样允许程序员工作的应用程序不需要进行修改。

遗憾的是，由于 Microsoft 最初并没有明确 BizTalk 到底是什么。你可以看到，它集成了许多我们在前面章节中见到的思想。它包括 BizTalk XML 模式和 DTD 容器的特征，这就允许用户使用共享的词汇表在贸易伙伴间进行交易，甚至接收方可以将获得的消息转换成使用另一个词汇表的消息格式。总之，它要证明自己是一个有用的工具，能将 EDI 风格的交易推广到许多小的或中型企业。



## 12.11 小结

在本章中我们描述了许多方面。从介绍现存的 EDI应用如何工作，到了解应用程序怎样使用 XML。我们看到在电子商务中使用 XML的问题并不是简单地共享一个模式或在贸易伙伴间按照模式发送消息。例如，需要对 DTD或贸易伙伴之间的规则模式进行客户化，我们并不总是能够如愿使用简单的支持技术（如 DOM或XSLT），需要考虑当需要同一个应用程序集成时将会发生什么。

尽管出现了许多我们没有预见到的问题，XML工具应用和应用软件（为了获得互操作性）还是有助于提高人们对这个领域的兴趣，越来越多的人开始要求在他们的电子商务系统中使用 XML。

接下来我们看了看一些纵向行业是如何通过使用 XML获得利益的，以及他们的消息是什么样子的，越来越多的行业加入到创建 XML模式的工作中以描述他们需要的处理流程，而其他的如UML、图表、数据库模式以及软件都将帮助用户实现这个解决方案。

最后，我们看到了一些公司是如何推动市场发展的。Microsoft的BizTalk模式不仅帮助了纵向行业，同时也帮助了横向行业以及公司内部的交易，BizTalk向我们展示了XML在这些解决方案中的力量。希望所有的例子看起来都比学习现存的 EDI方案简单。记住学习 XML语言的关键是对基础的东西深刻领会。如果你理解怎样书写和阅读 XML文档，在学习任何新的适用不同商业的XML语言中你就成功了一大半。

我们现在有了一个将成千上万的企业带入到电子商务领域的好的途径，障碍仍然需要明确并加以克服。在 1998年，一个很受尊敬的行业分析指出，到 2002年新兴行业的主要交换使用 XML工具的机会是 80%。

技术正在将预期变成现实。为了达到这个预期，我们需要将注意力放在全球化的电子商务版本和保证在扩展新的 XML工作草案时出现的复杂情况时，不要忘记提供简单、一致性、易学、易维护的电子商务系统。同其前辈不同的是，这些电子商务系统同语言（如 Java、VB等）、数据库（所有）和操作系统（Windows 2000）紧密结合。在使用 XML技术进行工作时，我们需要在贸易交换外关注这些领域。

我们必须参加本地的电子商务用户组织以及所属行业的初始工作。如果你已经参加了一个协会并希望保持同现在或将来的标准的兼容性，参加像数据交换标准协会合作服务这样的组织。如果你的公司提供 XML工具，参加基于供应商的组织如 OASIS、RosettaNet或其他。许多供应商或中小型企业的问题是它们不能努力参加一个大的国家的、甚至国际化组织。技术能够帮助我们，有助于合作的工具能够帮助我们使用很少的时间和有限的资源共享更多的思想和努力。

如果你的公司名词（信息）和动词（处理）没有很好地定义，现在是开始培训雇员的时候了。现在就开始定义一个企业的结构以适应你的要求，掌握一些工具。现在是使用这项关键的电子商务技术的最好时机。

有帮助的链接

XML/EDI Group: <http://www.xmledi.org/>

Bizcodes: <http://www.bizcodes.org/>

Data Interchange Standards Association: <http://www.disa.org/>

The World Wide Web Consortium(W3C): <http://www.w3c.org/>

Interactive Financial Exchange: <http://www.ifxforum.org/>

Open Travel Alliance: <http://www.disa.org/opentravel.com/index.htm>

RosettaNet: <http://www. rosettanet.org>

Electronic Business XML: <http://www.ebxml.org/index.html>