XML以其开放、自描述、向前兼容的特性逐渐成为数据交换的事实标准，并将触角伸展到金融行业的不同领域，尽管道路不是很平坦，颇有些泥泞，但XML在金融业的应用依然向前。

渐行渐近的行业标准

　　目前，针对不同的金融应用领域已经出现了几种不同的XML 格式。如Interactive Financial Exchange (IFX)和 Open Financial Exchange (OFX)标准，它们处理的对象是消费者和其他形式的小额银行业务。

　　Financial Information eXchange (FIX)是作为产权交易数据的标准通信协议出现的。SWIFT在加入 ISO 15022 XML 工作组后，已经采用XML 作为主要的表示格式，并把它的历史悠久的数据模型转化成了 XML 形式。Financial Products Markup Language(金融产品标记语言，FpML)用于金融衍生市场事务，通常用于错综复杂的协商。eXtensible Business Reporting Language(可扩展商业报告语言，XBRL)，主要用于商业报告和数据的准备与交换。

　　上述这些标准都不能涵盖所有的银行业务。会计和审计制度的不同使这些标准很难应用在国内的金融行业。国内也缺乏专门的机构和人才对金融数据交换制定适合国情的，并具有一定权威性的标准，但是这些标准具有重要的参考意义不容忽视。

　　XML的诱惑已经使很多银行和金融公司开始制定内部的XML标准用于数据交换和存储。关于标记的制定、属性值的枚举、交易要素的多少、元素的细分等环节的随意编写也严重损害了XML的标准意义。不同银行在相同领域使用不同的XML报文规范，以及同一个银行内部不同系统之间使用不同的XML报文规范在很大程度上削弱了XML语言的开放性。

　　同时，在XML实际用于金融数据交换时要处理好以下五个问题，迈过这五道坎之后，XML将发挥出真正的实力。

　　1. DTD和Schema

　　为了说明XML词汇表的语法规则，可以采用文档类型定义描述(DTD)。DTD使用正式的语法定义XML文档的结构和允许值。通过创建DTD，能够正式而精确地定义词汇表，从而使解析器可以利用DTD验证文档实例的有效性和完整性。遗憾的是DTD的数据类型对XML文件的约束显得并不详尽。例如，账号和申请人的名字均被描述成#PCDATA(字符串)。但是账号是一个整数，还可能是一个32位长的整数。另一个遗憾是，DTD使用非XML文件来描述XML文件。使用XML Schema则可以在一定程度上解决上述问题。XML Schema是一个良好格式的XML文件，提供了多种数据类型定义并允许对这些定义进行扩展、限制和组合以创建自己的复杂类型。但是需要注意的是XML Schema规范正在发展之中。

　　作为银行跨系统的应用，使用DTD或XML Schema可以更好的表现和理解用于交换的XML文档结构，并对XML文档中的结构和内容错误进行指示。但是一般自定义的XML报文均缺少DTD或XML Schema定义，文档规则随意且不断修改。这也许在一个系统内部的交换没有什么问题，但对于跨系统和跨银行之间数据交换则会带来认识的差异和效率的低下。

　　2. 长度与性能

　　自解释的一个主要后果就是XML文档长度难以控制。标记和属性的显示大大加长了报文长度。报文长度的增加产生了两个方面的副作用:报文发送的成功率降低以及解析报文的内存和CPU占用增加。不论是使用SOCKET还是消息中间件进行报文传递一般都有报文长度的限制。随着报文长度的增加，通信的成功率明显降低，尤其是广域网通信。另外无论是使用DOM还是SAX解析XML文档，存在相当的内存和CPU资源占用。

　　金融交易一般交易元素众多，特别是加入客户关系管理和综合账户信息后，发送和返回的信息明显增多。常用的一种方法是对报文压缩和进行传递，实际应用中一般压缩比可以达到30～50%。另外就是使用缩写，例如，将Transaction \_Account \_Number写成tranAccNum，可以降低报文长度，在一定程度上能够避免人们将过多的注意力集中于标记而忽视真正的内容，当然也不能过于简单，否则失去了使用XML语言的意义。将某些信息作为XML属性进行定义也可以减少文档的长度。

　　3. 内存管理

　　所有的XML解析器基本上都分成两类:基于树结构的接口，如文档对象模型(DOM);基于事件的接口，如简单XML API (SAX)。DOM类型的接口使用树型结构作到随机遍历和修改XML文档。但是使用这种类型的接口占用内存较多。当使用XML处理大量的数据交换时会对系统产生压力，甚至出现系统崩溃。

　　基于事件的接口可能是一个好的选择。它不支持对XML文档的随机访问，而是采用一种顺序访问的方式。无论XML文档的大小，所用内存的多少基本上是一定的。同时，由于在运行时不用创建新的对象，处理器时间占用也较少。但是使用SAX类型的接口编程比较复杂，并且没有对文档的后向引用。

　　在实际应用中，如果重视易用性，一般使用DOM接口;如更关注性能优势则选择SAX类型的接口较好。当然也可以通过减少XML元素嵌套层数减少DOM解析树的内存占用。

　　4. 标记粒度

　　元素的粒度在制定XML规范时总能让人产生困惑。某些内容是合在一起成为一个元素还是分开作为多个元素进行标记?例如姓名，是分成姓和名两个元素进行标记，还是放在一起作为一个元素?个人觉得关于粒度的标记定义指导原则就是一个:尽量细分。只有细分粒度越小，才可能支持各种形式的组合。另外许多信息来源于遗留系统，如果采用囫囵吞枣的方式延用其信息格式，表面上看节省了数据细分的难度，但是这对数据的共享和数据的整合会产生重大的障碍。

　　5. 结构化定义

　　很多时候为了提高XML文档解析的效率或缩短文档长度，有意识地避免采用多层结构化的XML文档定义。当然不是说XML文档没有根节点，但是会将节点的层次控制在2 ～3层。这显然违背了XML设计的初衷。没有了结构的XML文档同一般的交换报文还有什么差异呢?实际上大量交换的数据和信息存在层次。

　　在金融领域，可以在业务种类和业务要素两个领域对节点层次进行划分。首先是业务种类领域。一般交易数据包含报文头、客户基本信息、账户信息、交易相关信息等几个大部分组成，其下则包含业务要素领域元素，如账户信息领域下就包括账号、账户类型、存期、币种、册号、笔号、开户日期等多个相关业务要素元素。通过这两个层次的划分，基本可以说明用于交换的XML报文的层次结构。而业务要素领域元素则可以进一步划分层次说明相关属性。