ICS 35.240.01
A11
备案号：

JR

中华人民共和国金融行业标准

JR/T 0107.5—2014/ISO 17369-5:2013

统计数据和元数据交换（SDMX）第5部分：注册表规范 逻辑功能和逻辑接口

Statistical data and metadata exchange (SDMX)—
Part 5: Registry specification: logical functionality and logical interfaces

（ISO 17369-5:2013, SDMX registry specification: logical functionality and logical interfaces, IDT)

2014－08－28 发布　　2014－08－28 实施

中国人民银行　发布

目　次

前　言

JR/T 0107《统计数据和元数据交换（SDMX）》分为七个部分：

——第1部分：框架；

——第2部分：信息模型　UML概念设计；

——第3部分：SDMX-ML　模式和文档；

——第4部分：SDMX-EDI　语法和文档；

——第5部分：注册表规范　逻辑功能和逻辑接口；

——第6部分：SDMX技术说明事项；

——第7部分：Web服务用法指南。

本部分为JR/T 0107的第5部分。

本部分依据GB/T 1.1-2009规则起草。

本部分等同采用ISO 17369-5：2013《统计数据和元数据交换（SDMX） 第5部分：注册表规范 逻辑功能和逻辑接口》。

本部分由中国人民银行提出。

本部分由全国金融标准化技术委员会（SAC/TC 180）归口。

本部分主要起草单位：中国人民银行调查统计司、中国金融电子化公司。

本部分主要起草人：盛松成、徐诺金、姚力、巴运红、任全忠、潘润红、李曙光、韩建国、贾树辉、李兴锋、霍建兵、邓琳莹。

引　言

统计数据和元数据交换（SDMX）标准由SDMX国际组织发起并提出。SDMX国际组织是由国际清算银行（BIS）、经济合作与发展组织（OECD）、欧盟统计局（Eurostat）、欧洲中央银行（ECB）、国际货币基金组织（IMF）、联合国（UN）和世界银行（WB）七个国际组织联合建立，其制定发布的《统计数据和元数据交换》标准规定了统计人员在采集、处理和交换统计数据时所使用的统计概念和方法，规范了对外披露统计信息时统计数据的机构范围、地理区域、存流量性质、时间属性、频度以及文件格式等内容。

SDMX 标准提供了统计数据及元数据交换和共享的标准化格式，可以达到更好地扩展和高效率使用的目的。目前SDMX标准主要应用领域为部分国家中央银行和统计部门。本标准作用在于规范我国金融统计标准体系的内部处理和对外发布，促进金融统计的互联互通、信息共享和业务协同，提高信息共享的效率，满足金融综合统计的需要。

关于 SDMX（STATISTICAL DATA AND METADATA EXCHANGE）商业前景的文件用于推广“数据共享”模型，促进低成本、高质量的统计数据与元数据交换的实现。数据共享通过允许组织只需将数据发布一次，并使其合作伙伴根据需要提取数据和相关的元数据而减少了组织发布消息的负担。

本部分所述的注册表服务是旨在帮助SDMX团体进行SDMX的库容扩大管理，并达到支持数据共享以实现报告和分发的目的。

在本部分中，注册表接口的函数和表现形式以下述三种方式进行描述：分别为文本形式、表格、摘自SDMX信息模型的UML图（SDMX-IM）；还包括其他一些不属于SDMX-IM的UML图表，这些图表可以帮助我们清晰描述和实现注册表接口的函数（这些图表以“逻辑类图”进行标识）。

统计数据和元数据交换（SDMX）
第5部分：注册表规范　逻辑功能和逻辑接口

1　范围

本部分规定了SDMX注册表的逻辑功能和逻辑接口相关内容。

本部分适用于金融统计中数据和元数据的交换和共享。

2　SDMX 注册表/存储库的目标

2.1　目标

从广义上讲，SDMX注册表/存储库的目标是允许组织能够以已知的格式发布统计数据和参考元数据，使得感兴趣的第三方可以在最短的时间内准确无误地发现并理解这些数据。以下是实现该目标的两种机制：

——维护并发布结构化元数据。这些元数据描述了数据和参考元数据来源（如数据库、元数据库、数据集、元数据集）的结构和有效内容；

——使应用程序、组织机构和个人分享并发现数据和参考元数据。SDMX 数据共享方便了数据和参考元数据的传播。

2.2　结构化元数据

对于维护机构，设置结构元数据和交换背景（以下简称“数据供应”）包含以下步骤（见图1）：

——同意并生成数据结构规范（本章节称为数据结构定义或 DSD，还被称为“关键字族”），该规范定义了数据集的维度、测量值、属性及其有效值集合；

——根据需要，定义 DSD 的子集或视图，该子集或视图允许某些称为“数据流定义”的内容来限制；

——同意并生成参考元数据结构规范（元数据结构定义），该规范定义了元数据集的属性和显示方式及其有效值和内容；

——根据需要定义 MSD 的子集或视图，该子集或视图允许某些称为“元数据流定义”的内容限制；

——定义哪些主题域（规范为类别方案）是与数据流和元数据流定义相关的，以便于用户浏览；

——定义一个或多个数据供应商名单（包括元数据供应商）；

——定义发布哪些数据供应商认同的给定数据流和/或元数据流定义——即所谓供应协议。

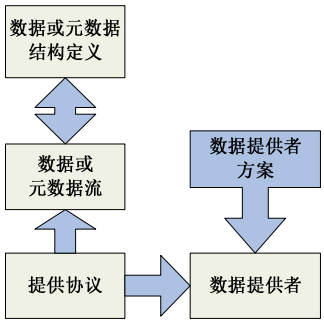

图 1 SDMX-IM 基本结构内容原理图

2.3 注册

数据提供者发布数据和参考元数据包括以下步骤（见图2）：

——使得元数据和参考数据在 SDMX-ML 一致性数据文件中可用(作为 SDMX-ML 查询 SDMX-ML 数据的响应)；数据和参考元数据文件或数据库必须可通过网络访问，而且必须与公认的数据流或元数据流定义（“数据结构定义”或“元数据结构定义”子集）保持一致；

——在一个或多个注册表中注册已发布的元数据和数据文件或数据库的存在性。

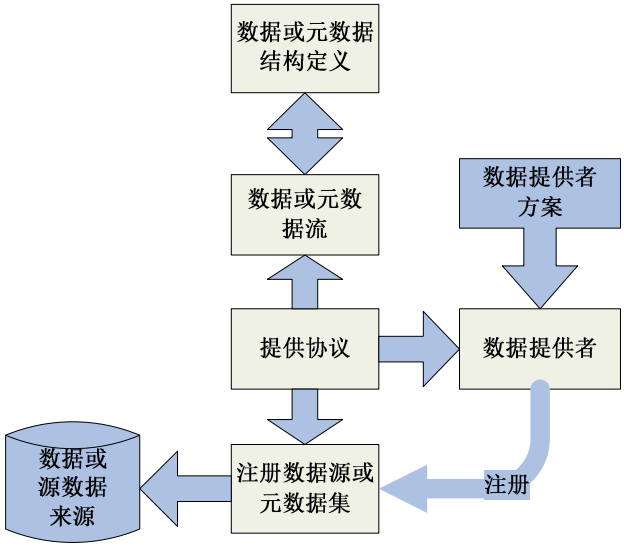

图 2 SDMX-IM 中已注册的数据、元数据源原理图

2.4 通知

通知最新发布或重发布的数据、参考元数据，或结构元数据中的变化给有关当事人：

注册表能够可选择地支持基于订阅的通知服务,即发送电子邮件或通知HTTP地址告之所有符合准则要求的数据，而这些准则包含在订阅申请中。

2.5 发现

发现已公布的数据和参考元数据,要与注册表进行交互，该交互需要完成以下逻辑步骤：用户先调用某服务，而服务再与注册表及已启用的SDMX数据或参考元数据资源之间进行交互。具体步骤如下（见图3）：

——有选择地浏览一个主题类别目录，找出用于构造被搜索的数据和元数据类型的数据流定义（及“关键字族”）和元数据流；

——在选定的“数据结构定义”或“元数据结构定义”上构建一个查询，该查询必须规定哪些数据是必需的,通过服务向 SDMX 注册表提交查询,将返回满足查询的数据和参考元数据文件以及数据库的清单（或其 URL，即通用资源地址）；

——处理查询结果集合，并从返回的 URL（通用资源地址）中检索数据和/或参考元数据。

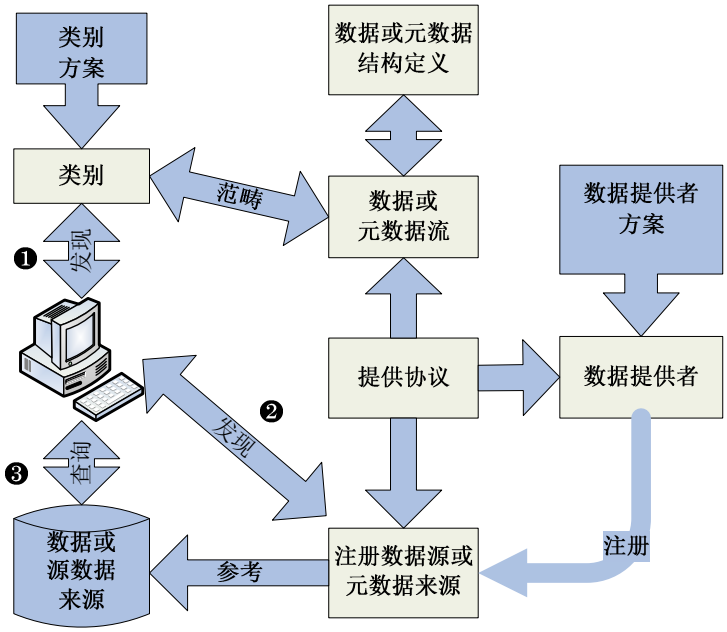

图 3 SDMX-IM 中数据、元数据发现、查询原理图

3 SDMX 的注册表/存储库结构

3.1 结构示意图

SDMX注册表/存储库体系结构来源于上述规定的目标。它是一个基于结构元数据库的分层体系结构，该结构元数据库支持供应元数据库，而且后者支持注册表服务。SDMX-ML结构全面支持这些功能.

应用程序建立在这些服务之上，这些服务包括启动应用程序所需要整个统计生命周期，其中包括报告、存储、检索、和发布，另外，还包括维护结构化元数据。

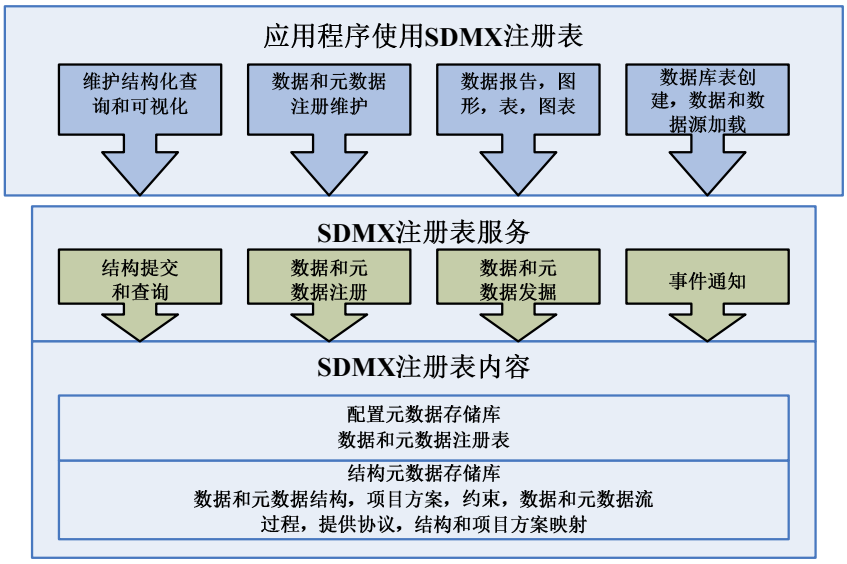

图4 注册表内容和服务概图

3.2 结构元数据库

基本层是支持SDMX结构元数据产物生命周期的结构元数据服务层，这些产物包括维护机构、数据结构定义、元数据结构定义、提供许可协议、过程等。该层由结构发布（Structure Submission）和查询服务（Query Service）提供支持。

注意SDMX-ML Submit Structure Request报文几乎支持所有的SDMX结构化产物，SDMX-ML Submit Structure Request唯一不支持的是：

——数据和元数据源的注册；

——订阅和通知。

以上两种结构化内容由基于注册的独立报文提供支持。

3.3 供应元数据库

供应元数据库的功能是支持结构化元数据的定义，结构化元数据功能是描述各种数据存储的类型，并且为符合SDMX的数据库或文件建模，并且将这些数据链接到其数据源，链接指向数据提供商，或特定的数据或元数据流。这在SDMX模型中整个过程被称为提供协议。

这一层的功能由数据和元数据注册服务来实现。

4 注册表接口和服务

4.1 注册表接口

注册表接口包括：

——通知注册事件；

——提交订阅请求；

——提交订阅反馈；

——提交注册请求；

——提交注册反馈；

——查询注册请求；

——查询注册反馈；

——查询订阅请求；

——查询订阅反馈；

——提交结构请求；

——提交结构反馈。

可以通过两种途径调用注册表接口：

a) 通过SDMX-ML文档根节点的名称。

b) 作为注册接口报文的子元素，这里注册接口指SDMX-ML文档的根节点。

除了这些接口，注册表必须提供一种查询结构化元数据的机制，细节在4.2.2节讨论。

注册表中的这些接口中，除了通知注册事件，都是成对设计的。调用SDMX-RR接口的请求文档是第一个文档，而接口返回的报文为反馈文档。

应注意是，所有的接口都是同步的。对于通知注册事件，无论事件何时发生，无论订阅者是谁，通知注册事件形成的文档都由SDMX-RR实时发给所有订阅者。因此，它不符合“请求-反馈”模式，是不同步的。

4.2 注册表服务

4.2.1 简介

本节所描述的服务并不都实现为独立的web服务。

4.2.2 结构提交和查询服务

该项服务必须实现以下SDMX-ML接口

——提交结构请求；

——提交结构反馈。

这些接口允许在一种受控的模式下，新建、修改和删除结构定义，它也允许对结构化元数据内容进行部分或整体地查询、检索。为增加体系架构可扩展性，能被SDMX-RR 处理的结构化元数据的最优粒度为MaintainableArtefact（见SDMX信息模型的下一节）。

4.2.3 结构化查询服务

注册表必须提供查询结构化元数据的机制。该查询可以是SDMX-ML查询报文或者是SDMX REST接口（定义参见SDMX标准第7部分），或者该查询机制两种都提供。注册表通过包含有根节点SDMX结构报文对这两种查询的结果进行反馈。

——结构

可被查询的 SDMX 结构化产物为：

——数据流和元数据流；

——数据结构定义和元数据结构定义；

——代码表；

——概念方案；

——报告分类；

——提供协议；

——结构集；

——进程；

——分级代码表；

——约束；

——类别方案；

——类别和类别对象（例如分类数据流、元数据流、数据结构定义、元数据结构定义、已注册数据源和元数据源的供应协议）；

——组织方案(机构方案、数据提供者方案、数据使用者方案、组织单元方案)。

作为 SDMX-ML 查询报文组成部分的 SDMX 查询报文包括：

——结构查询；

——数据流查询；

——元数据流查询；

——数据结构查询；

——元数据结构查询；

——目录方案查询；

——概念方案查询；

——代码表查询；

——分层代码表查询；

——组织方案查询；

——报告分类查询；

——结构集查询；

——进程查询；
——组织查询；
——供应协议查询；
——约束查询。

4.2.4 数据和参考元数据注册服务

该服务必须实现以下SDMX-ML 接口:
——提交注册请求；
——提交注册反馈；
——查询注册请求；
——查询注册反馈。

注册服务允许符合XML格式的SDMX文件以及包含已发布的数据和相关元数据的网络可读取数据库在SDMX注册表中进行注册。注册过程能验证数据集和元数据集内容、能根据内容概念值(比如数据属性维度值,元数据属性)或所有关键字抽取简单表示形式、并将该表示形式作为记录存储在注册表中。这被称为SDMX-IM约束。

受注册表存取控制机制限制，数据和元数据注册服务能验证以下内容：
——允许数据提供者注册数据集或者元数据集；
——数据集和元数据集内容满足验证约束。这取决于结构化存储库中定义的验证约束，并引用相关数据流、元数据流、数据提供者、数据结构定义、元数据结构定义、提供协议；
——要查询的数据源必须是存在的--通过注册服务查询来确定；
——存在简单的数据源（通过 URL 地址找到可访问的文件）；
——已注册数据的数据结构定义或元数据结构定义必须正确；
——组件（维度、属性、度量、组件标识符等）必须与数据结构或元数据结构中的定义保持一致；
——这些组件对应概念的有效表示必须符合数据结构的定义或元数据结构的定义。

注册的动作属性取值如下：见表 1。

表 1

操作属性值	操作
追加	在注册表中增加注册信息
替换	识别提交注册请求中所包含的注册信息 id，并替换已存在的相同 id 的注册信息
删除	识别提交的注册请求中所包含的注册信息 id，并删除已存在的相同 id 的注册信息

注册有3个布尔属性，这些属性决定SDMX所兼容的数据集或元数据集索引应用如何通过注册表来索引数据集或元数据集，索引程序的有关操作如下：见表2。

表 2

布尔属性	布尔值为“真”时的操作
indexTimeSeries	兼容的索引应用必须为所有时间序列值（为数据集注册）或元数据目标值（为元数据集注册）建立索引
indexDataSet	兼容的索引应用必须为数据集（数据集注册）各种维度真正值域或每个元数据属性的实际值域建立索引。

	注意，数据比关键索引需要的存储空间要少，但是这种方法不能确保在数据集中没每个维度值都有具体指向。
indexReportingPeriod	兼容的索引应用必须数据集或元数据集数据的时期范围建立索引。

4.2.5 数据和参考元数据发现

数据和参考元数据服务实现以下注册表接口：

——查询注册请求；

——查询注册反馈。

4.2.6 订阅和通知

订阅和通知服务实现以下注册表接口：

——提交订阅请求；

——提交订阅反馈；

——通知注册事件。

数据共享模版依赖于能够从数据供应商分发系统中提取信息的数据和元数据用户。为有效实现该目的，数据用户需知晓何时提取数据，即，当注册表中的某些内容发生变化时（如数据集已更新且重新注册）。此外，SDMX系统也希望知道是否增加了新的数据结构定义、代码表或元数据结构定义。订阅和通知服务包含两部分：订阅管理及通知。

订阅管理涉及已鉴别用户提交订阅请求，请求包括如下内容：

——查询或约束表达式形式的过滤器，它能够指出哪些内容是用户感兴趣的（如：特定数据流中的新数据，或者域范围的新数据，或者数据结构定义中的变化）；

——URI 或 XML 通知报文发送结点的列表，支持的结点类型有：电子邮件（mailto:）和 HTTP 协议（http://地址）；

——已提交订阅列表的请求；

——订阅的删除。

通知要求结构元数据库和供应元数据库来监控用户感兴趣的任一事件（订阅请求查询的目标），并且发布SDMX-ML通知文件到相关订阅中规定的结点。

4.2.7 注册操作

下表定义了各种注册表接口报文的注册表注册操作：见表3。

表 3

接口	操作
全部	a) 如果是“替换”操作，除非最终属性设为“真”，否则，Registry MUST 中现有维护对象的所有内容都将被提交的对象所替代，如果最终属性设为“真”，则仅需要对如下结构作出改变： —— 名称—适用于可维护对象及其所包含的元素，如代码表中的代码； —— 描述—适用于可维护对象及其所包含的元素，如代码表中的代码；

	—— 注释—适用于可维护对象及其所包含的元素，如代码表中的代码； —— ValidTo； —— ValidFrom； —— StructureURL； —— ServiceURL； —— Uri； —— isExternalReference. b) 交叉引用结构必须包含在所提交的文档或提交请求的注册表中。 c）如果是“删除”操作，则注册表要确认对象是否能删除，删除必须确认： 1) 没有设为“真”的最终属性； 2) 没有被注册表中的其它任何对象所引用。 d)必须遵循SDMX 架构文档的版本规则。 e）必须遵循SDMX 架构中所规定的元素和属性的具体规则。
提交结构请求	如果在可维护内容层提交结构，则以上所有的操作都处于可维护内容层。
提交配置请求	无额外操作
提交注册请求	如果数据源是一个文件（简单数据源），则可能要根据注册表中的布尔属性集对文件进行检索和索引。 对于包含查询数据源，注册表必须验证数据源存在并且可以被查询。

5 SDMX 对象标识

5.1 标识、版本和维护

所有主要的SDMX信息模型类都继承自：

——IdentifiableArtefact：使得该类对象能够唯一地被标识（见下述关于标识的章条）以携带多语言名称和描述，同时具有用户定义的 URI，并携带多语言注释；

——NamableArteface:具有 IdentifiableArtefact 所有特征，同时能用多种语言命名和描述；

——VersionableArtefact：具有所有上述特征以及版本号和有效期限；

——MaintainableArtefact：具有所有上述特征，同时该对象是否是最终版本及不可更改和删除的标识以及与该对象维护机构的合作方式。

5.1.1 标识，版本和维护模型

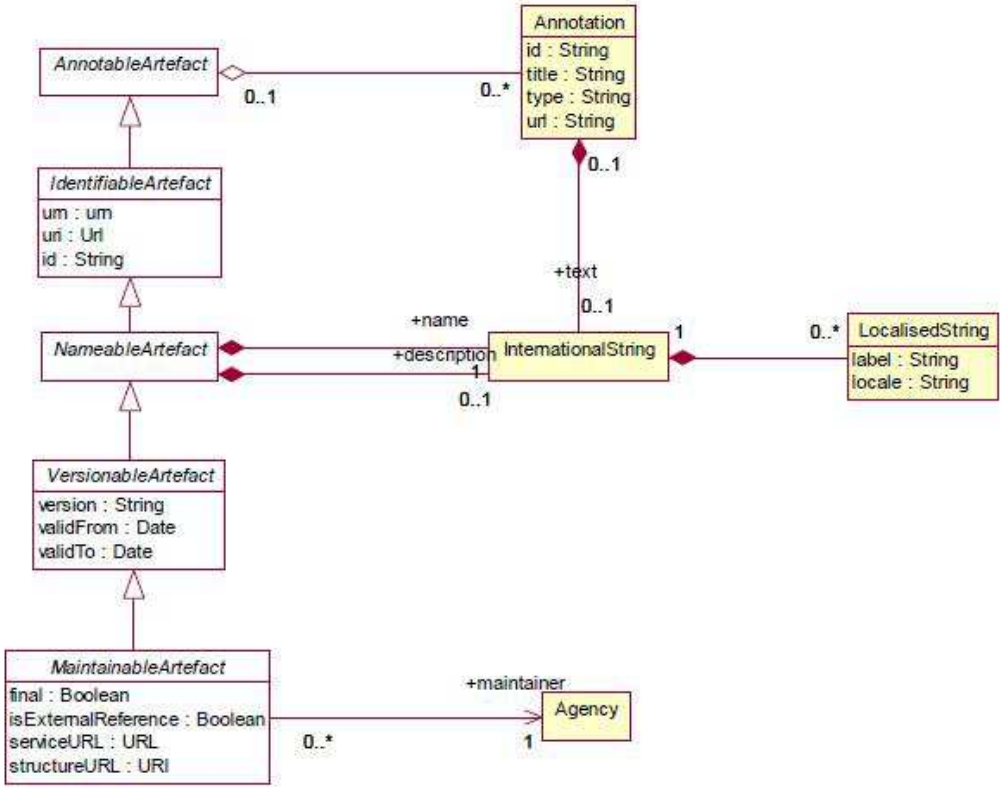

图5 SDMX-IM的基本类图

下表给出了注册表中存储的下述任一对象的标识及相关数据属性：

——可注释对象（Annotable）；

——可标识对象（Identifiable）；

——可命名对象（Nameable）；

——版本可控对象（Versionable）；

——可维护对象（Maintainable）。

表 4 对不同对象的一般属性进行了描述和说明。

表 4

对象类型	数据属性	状态	数据类型	注释
可注释的	注释标题	C	字符串	
	注释类型	C	字符串	
	AnnotationURN	C	字符串	
	国际字符串格式的注释文本	C		可以使用特定语言变量
可识别的	关于Annotable plus的所有内容			
	Id	M	字符串	
	Uri	C	字符串	
	Urn	C	字符串	尽管 urn 是可计算出的，不必进行提交和物理存储，但是注册表必须为每个对象返回 urn，并仅仅为通过 urn

				引用的对象提供查询服务
可命名的	关于 Identifiable plus 的所有内容			
	国际字符串格式的名称	M	字符串	可以使用特定语言变量
	国际字符串格式的描述	C	字符串	可以使用特定语言变量
版本可控的	关于 Identifiable plus 的所有内容			
	版本	C	字符串	版本号。如果未显示，则默认版本号为 1.0
	ValidFrom	C	日期/时间	
	ValidTo	C	日期/时间	
可维护的	关于 Versionable Plus 的所有内容			
	Final		布尔值	“真”值表示是最终确定的对象，除非发布新版本，否则是不能更改的。注意如果提供的一个“最终”对象没有引用其它对象，则它就有可能被删除
	isExternalReference	C	布尔值	“真”值表示实际资源是在注册表之外获得的，实际引用是由注册表 URI 或 structureURI 所提供的，两者都应返回一个合法 SDMX-ML 文件
	serviceURL	C	字符串	查询资源的网址
	structureURL	C	字符串	获取资源的网址
	(Maintenance) agencyId	M	字符串	对象必须与某维护代理相关联

5.2 SDMX 对象的唯一标识

5.2.1 机构

SDMX的维护机构(Maintenance Agency)在机构方案（Agency Scheme）中进行维护，它本身是组织图的一个子类—具体内容参见以下类图。

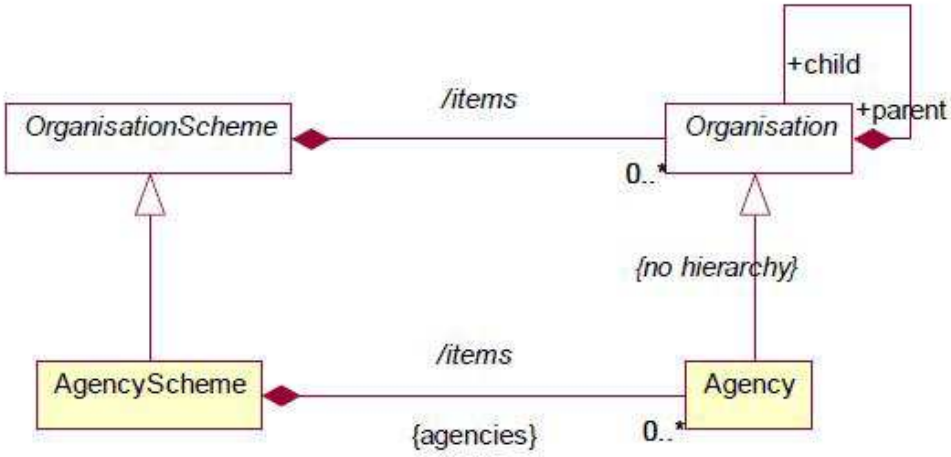

图 6 Agency 表模型

SDMX机构非常重要，SDMX所使用的机构ID系统是一个n层结构。最上层结构由SDMX维护，最上层结构能声明子机构，并且任何子机构也能声明自己的子机构。机构方案（Agency Scheme）有固定的id和版本，不能在SDMX对象识别机制中明确声明。

为实现以上阐述的内容，SDMX采用以下规则：

a) 机构必须隶属一个机构方案（Agency Scheme）（组织方案（Organisation Scheme）的子类）。

b) 机构必须在一个（不同的）机构方案（Agency Scheme）中声明。

c) 最上层机构是SDMX，并在最上层机构方案（Agency Scheme）中维护。

d) 在最上层的机构能自己维护一个单独的机构方案（Agency Scheme）。第二层机构能自己维护一个单独的代理机构，如此类推。

e) 机构方案(Agency Scheme)没有版本标记，但有默认版本号1.0，而且不能将版本标记为“final”。

f) 也可能有一个机构方案（Agency Scheme）被任何一个机构维护，则有固定的ID为AGENCIES。

g) 维护机构并没有继承组织机构的层级，因此，每个机构方案（Agency Scheme）都是一个扁平的维护机构列表。

h) 机构标识符的格式是agencyID.agencyID等。识别机制的最上层机构是在SDMX代理机构中注册过的。换句话说，SDMX不属于机构分层ID结构的一部分。然而，SDMX本身是一个在最上层代理机构中的维护机构。

支持分层agencyID结构。如下例所示：

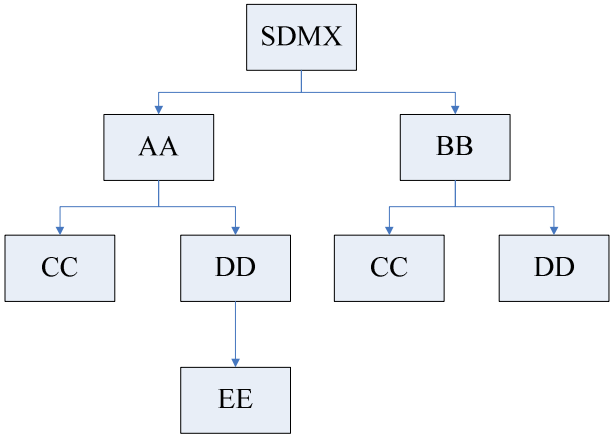

图 7 分层结构代理实例

以下组织维护一个代理机构：

——SDMX— 包含机构 AA、BB；

——AA— 包含机构 CC、DD；

——BB— 包含机构 CC、DD；

——DD— 包含机构 EE。

除SDMX外，每个机构都由完整的分层形式来表示。

如：agencyID的EE的id ，表示为AA.DD.EE。

例如以下摘录：

```
<structure:Codelists>
<structure:Codelist id="CL_BOP" agencyID="SDMX" version="1.0"
 urn="urn:sdmx:org.sdmx.infomodel. codelist.Codelist =SDMX:CL_BOP[1.0]">
    <common:Name>name</common:Name>
</structure:Codelist>
<structure:Codelist id="CL_BOP" agencyID="AA" version="1.0"
 urn="urn:sdmx:org.sdmx.infomodel. codelist.Codelist =AA:CL_BOP[1.0]" >
    <common:Name>name</common:Name>
</structure:Codelist>
<structure:Codelist id="CL_BOP" agencyID="AA.CC" version="1.0"
 urn="urn:sdmx:org.sdmx.infomodel.codelist.Codelist=AA.CC:CL_BOP[1.0]" >
    <common:Name>name</common:Name>
</structure:Codelist>
<structure:Codelist id="CL_BOP" agencyID="BB.CC" version="1.0"
 urn="urn:sdmx:org.sdmx.infomodel. codelist.Codelist =BB.CC:CL_BOP[1.0]">
    <common:Name>name</common:Name>
</structure:Codelist>
</structure:Codelists>
```

图 8 机构识别实例

所有这些维护机构都有一个名为CL_BOP的ID号列表。每个机构都通过分层机构结构被唯一识别。

5.2.2 统一资源名称（URN）

为了在分布式网络环境中使得SDMX注册表/存储库间具有互操作性，具有一个能唯一识别（因此访问）所有一流（可识别的）SDMX-IM对象的模式是非常重要的。大部分唯一标识符是组合体（包括维护机构、或父对象标识符），并且需要被定义为唯一的单一字符串。这种在全球范围内实现的唯一标识符被称为通用资源名（URN），此URN产生于SDMX-RR API中的实际识别组件。也就是，任何可识别的内容的URN都由各自的组件标识符（代理、ID、版本等）所构成。

CASE Rules for URN

URN，大小写敏感，任何对象的ID必须是大写。因此，CRED_ext_Debt是非法的，应为CRED_EXT_DEBT。

URN的一般结构如下：

SDMXprefix.SDMX-IM-package-name.class-name=agency

id:maintainedobject-id(maintainedobject-version).*container

object-id.object-id

*this can repeat and may not be present (see explanation below)

注：在SDMX信息模型中，所有版本可控的部分都是可维护的，因此，唯一允许的版本信息是为可维护对象而设计的。

维护代理标识符与可维护内容的标识符之间用冒号“:”隔开，SDMX URN语法中所有其它标识符都用“.”隔开。

在下面解释中URN目标被称为实际对象（*actual object*）

SDMXPrefix: urn:sdmx:org.
SDMX-IM package name: sdmx.infomodel.package=
The packages are:

base
codelist
conceptscheme
datastructure
categoryscheme
registry
metadatastructure
process
mapping

maintainable-object-id 是可维护对象的标识符。当可识别对象是可维护对象或它本身包含在可维护对象之中时，就经常会用到可维护对象的标识符。

(maintainable-object-version)是可维护对象的版本，用括号括起来，并经常会用到它。

container-object-id是中间对象标识符，这个中间对象标识符包含了URN识别的实际对象。中间容器对象并不是必需的，许多实际对象都没有中介容器对象。例如，一个代码在被维护对象（代码表）中，它也没有中间容器对象。但是，元数据属性有中间容器对象（报表结构）并且这个中间容器对象可能是父元数据属性。由于这个原因，容器对象id可能重复，通过每一次重复在层次结构的更低一级识别对象。注意，如果在模型中只包含一个对象，那么这个对象就不在URN结构内。这同样也适用于属性描述、维度描述、维度描述，这些描述都只有一个对象，且都有固定的对象ID。因此，尽管每一种对象都有URN，但是当实际对象是数据属性或维度/度量、维度/时间维度，或度量时，属性描述id，维度描述id以及度量描述id就没有URN。

注意，尽管代码可能有父代码，概念可能有父概念，但是因为它们都在扁平结构中进行维护，所以都没有container-object-id。

例如序列是agency:DSDid(version).DimensionId而不是

agency:DSDid(version).DimensionDescriptorId.DimensionId。

object-id是实际对象的标识符，除非这个实际对象是可维护的对象。Object-id经常在最后出现，其后不再有其它字符。

URN结构一般实例

实际对象是可维护的

SDMXPrefix.SDMX-IM package name.classname=agency id:maintained-object-id(version)

实际对象包含在没有中介容器对象的维护对象中

SDMXPrefix.SDMX-IM package name.classname=agency
id:maintained-object-id(version).object-id

实际对象包含在有中介容器对象维护对象中

SDMXPrefix.SDMX-IM package name.classname=agency
id:maintained-object-id(version).contained-object-id.object-id

实际对象包含在没有中间容器对象的维护对象中，但是对象类型本身是分层的。

在这种情况下，对象id本身可能不是唯一的，但是在上下层次结构中是唯一的。URN的一般语法中，所有中间结构对象（当然，维护对象除外）都作为包含对象出现。这儿有一个例子是类别方案（Category Scheme）中的类别，这个类别是分层的，并且所有的中间类别出现在一个包含对象中。下面的例子显示了目录表“类别方案（Category Scheme）/类别/类别”的一般结构。

SDMXPrefix.SDMX-IM package name.classname=agency

id:maintained-object-id(version).contained-object-id.object-id

实际对象包含在有中介容器对象的维护对象中，而且对象类型本身是分层的。

在这种情况下，当父对象被看作是包含对象时，它的一般语法和上面例子的语法一样。这儿的例子是元数据属性，包含的对象是报表结构（第一个包含对象的id）和元数据属性（后续包含对象的id）。下面的例子显示了MSD/报表结构/元数据属性/元数据属性的一般结构。

SDMXPrefix.SDMX-IM package name.classname=agency

id:maintained-object-id(version).contained-object-id.

contained-object-id contained-object-id.object-id

URN结构的具体实例

由最上层代理TFFS维护的数据结构定义CRED_EXT_DEBT 版本1.0,的URN为:

urn:sdmx:org.sdmx.infomodel.datastructure.DataStucture=TFFS:CRED_EXT_ DEBT(1.0)

由代码列表CL_3166A2 版本 1.0中的ISO维护的阿根廷代码的URN为：

urn:sdmx:org.sdmx.infomodel.codelist.Code=ISO:CL_3166A2(1.0).AR

由目录表SUBJECT_MATTER_DOMAINS 版本1.0中的SDMX维护的目录（id为1），其子目录（id为2）的URN为：

urn:sdmx:org.sdmx.infomodel.categoryscheme.Category=SDMX:SUBJE
CT_MATTER_DOMAINS(1.0).1.2

元数据属性等级为CONTACT_DETAILS/CONTACT_NAME，且其是由报表结构为：

CONTACT_REPORT的MSD CONTACT_METADATA 版本 1.0的SDMX进行维护，这个元数据属性的URN为：

urn:sdmx:org.sdmx.infomodel.metadatastructure.MetadataAttribut
e=SDMX:CONTACT_METADATA(1.0).CONTACT_REPORT.CONTACT_DETAILS.CO
NTACT_NAME

如果TFFS将ABC作为TFFS的一个子代理进行定义，那么由ABC维护并被定义为EXTERNAL_DEBT 版本1.0的数据流的URN为：

urn:sdmx:org.sdmx.infomodel.datastructure.Dataflow=TFFS.ABC:EX
TERNAL_DEBT(1.0)

SDMX-RR必须为全球唯一标识符表提供支持。SDMX-RR必须能从各自所提交的标识符属性来创建URN，并能将URN转换为这些标识符属性。这些标识符属性为：

——可识别、可命名的内容：id(在某种情况下，id 必须是分层的)；

——可维护的内容：id，版本，agencyid。

SDMX-RR必须能resolve SDMX 内容的唯一标识符，并且，如果内容在注册表中，则生成内容的SDMX-ML解释。

5.2.3 关于 SDMX-IM 包及类表

下表列出了SDMX-IM中所有的包、包中所含有的具体的类以及类的对象，这些类对象必须具有URN。表5描述了SDMX-IM包及其包含的类。

表 5

包	URN类名（不同的模型类名）
Base	Agency
	OrganisationUnitScheme

	AgencyScheme
	DataProviderScheme
	DataConsumerScheme
	OrganisationUnit
	DataProvider
	DataConsumer
Datastructure	DataStructure (DataStructureDefinition)
	AttributeDescriptor
	DataAttribute
	GroupDimensionDescriptor
	DimensionDescriptor
	Dimension
	MeasureDimension
	TimeDimension
	MeasureDescriptor
	PrimaryMeasure
	Dataflow (DataflowDefinition)
Metadatastructure	MetadataTarget
	DimensionDescriptorValueTarget
	IdentifiableObjectTarget
	ReportPeriodTarget
	DataSetTarget
	ReportStructure
	MetadataAttribute
	MetadataStructure(MetadataStructureDefinition)
	Metadataflow (MetadataflowDefinition)
Process	Process
	ProcessStep
	Transition
Registry	ProvisionAgreement
	AttachmentConstraint
	ContentConstraint
	Subscription
Mapping	StructureMap
	StructureSet
	ComponentMap
	ConceptSchemeMap
	OrganisationSchemeMap
	CodelistMap
	CategorySchemeMap
	ReportingTaxonomyMap

	ConceptMap
	OrganisationMap
	CodeMap
	HybridCodelistMap
	CategoryMap
	HybridCodeMap
	ReportingCategoryMap
Codelist	Codelist
	HierarchicalCodelist
	Hierarchy
	Hierarchy
	Code
	HierarchicalCode
	Level
Categoryscheme	CategoryScheme
	Category
	Categorisation
	ReportingTaxonomy
	ReportingCategory
Conceptscheme	ConceptScheme
	Concept

5.2.4 SDMX 对象的 URN 标识组件

下表描述了所有具有标识的SDMX对象类型的标识组件。注意，实际的属性都为标识，但在其类名或复合类名前增加前缀以显示指向。例如：conceptSchemeAgencyId其实是Agency类的一个标识属性，它与ConceptScheme有关。表6描述了SDMX可识别内容的识别组件。

符号“*”表明该对象是可维护的。

注意，为简便起见，所给出的URN的例子都是省略了前缀的。事实上所有的URN都应该有前缀。

urn:sdmx.org.sdmx.infomodel.{package}.{classname}=

表 6

SDMX 类	关键属性	URN 举例
Agency		IMF Sub agency in the IMF AGENCIES IMF.SubAgency1
ConceptScheme	conceptSchemeAgencyId:conceptSchemeId (version)	SDMX:CROSS_DOMAIN_CONCEPTS(1.0)
Concept	conceptSchemeAgencyId: conceptSchemeId(version).conceptId	SDMX:CROSS_DOMAIN_CONCEPTS(1.0).FREQ
Codelist	codeListAgencyId:codeListId(version)	SDMX:CL_FREQ(1.0)
Code	codeListAgencyId:codelistId(version). codeId	SDMX:CL_FREQ(1.0).Q

Hierarchical Codelist	hierachicalCodelistAgencyId: hierarchicalCodelistId(version)	UNESCO:CL_EXP_SOURCE(1.0)
Hierarchy	hierachicalcodeListAgencyId: hierarchicalcodelistId(version).Hierarchy	UNESCO:CL_EXP_SOURCE(1.0). H-C-GOV
Level	hierachicalcodeListAgencyId: hierarchicalcodelistId(version).Hierarchy.Level	ESTAT:HCL_REGION(1.0).H_1.COUNTRY
HierarchicalCode	hierachicalCodeListAgencyId: hierarchicalcodelistId(version).hierarchy.hierarc hicalCode	UNESCO:CL_EXP_SOURCE(1.0). H-C-GOV.GOV_CODE1
DataStructure	dataStructureDefintitionAgencyId: dataStructureDefintitionId(version)	TFFS:EXT_DEBT(1.0)
Dimension Descriptor Measure Descriptor Attribute Descriptor	dataStructureDefinitionAgencyId: dataStructureDefinitionId(version). componentListId where the componentListId is the name of the class (there is only one occurrence of each in the Data Structure Definition)	TFFS:EXT_DEBT(1.0).DimensionDescriptor TFFS:EXT_DEBT(1.0).MeasureDescriptor TFFS:EXT_DEBT(1.0).AttributeDescriptor
GroupDimension Descriptor	dataStructureDefinitionAgencyId: dataStructureDefinitionId(version). groupDimensionDescriptorId	TFFS:EXT_DEBT(1.0).SIBLING
Dimension	dataStructureDefinitionAgencyId: dataStructureDefinition (version). dimensionId	TFFS:EXT_DEBT(1.0).FREQ
TimeDimension	dataStructureDefinitionAgencyId: dataStructureDefinition (version). timeDimensionId	TFFS:EXT_DEBT(1.0).TIME_PERIOD
Measure Dimension	dataStructureDefinitionAgencyId: dataStructureDefinition (version). measureDimensionId	TFFS:EXT_DEBT(1.0).STOCK_FLOW
DataAttrribute	dataStructureDefinitionAgencyId: dataStructureDefinition (version). dataAttributeId	TFFS:EXT_DEBT(1.0).OBS_STATUS
PrimaryMeasure	dataStructureDefinitionAgencyId: dataStructureDefinition (version). primaryMeasureId	TFFS:EXT_DEBT(1.0).OBS_VALUE

Category Scheme	categorySchemeAgencyId: categorySchemeId(version)	IMF:SDDS(1.0)
Category	categorySchemeAgencyId: categorySchemeId(version). categoryId.categoryId categoryId.categoryId etc.	IMF:SDDS(1.0): level_1_category.level_2_category …
Reporting Taxonomy	reportingTaxonomyAgencyId: reportingTaxonomyId(version)	IMF:REP_1(1.0)
ReportingCategory	reportingTaxonomyAgencyId: reportingTaxonomyId(version) reportingcategoryId.reportingcategoryId	IMF:REP_1(1.0): level_1_repcategory.level_2_repcategory …
Categorisation	categorisationAgencyId: categorisationId(version)	IMF:cat001(1.0)
Organisation Unit Scheme	organisationUnitSchemeAgencyId: organisationUnitSchemeId(version)	ECB:ORGANISATIONS(1.0)
Organisation Unit	organisationUnitSchemeAgencyId: organisationUnitSchemeId(version). organisationUnitId	ECB:ORGANISATIONS(1.0).1F
AgencyScheme	agencySchemeAgencyId: agencySchemeId(version)	ECB:AGENCIES(1.0)
Agency	agencySchemeAgencyId: agencySchemeId(version). agencyId	ECB:AGENCY(1.0).AA
DataProvider Scheme	dataProviderSchemeAgencyId: dataProviderSchemeId(version)	SDMX:DATA_PROVIDERS(1.0)
DataProvider	dataProviderSchemeAgencyId: dataProviderSchemeId(version) dataProviderId	SDMX:DATA_PROVIDERS(1.0).PROVIDER_1
DataConsumer Scheme	dataConsumerSchemeAgencyId: dataConsumerSchemeId(version)	SDMX:DATA_CONSUMERS(1.0)
Data Consumer	dataConsumerSchemeAgencyId: dataConsumerSchemeId(version) dataConsumerId	SDMX:DATA_CONSUMERS(1.0).CONSUMER_1
Metadata Structure	MSDAgencyId:MSDId(version)	IMF:SDDS_MSD(1.0)
MetadataTarget	MSDAgencyId: MSDId(version).metadataTargetId	IMF:SDDS_MSD(1.0).AGENCY
Dimension DescriptorValues Target	MSDAgencyId: MSDId(version). metadataTargetId.keyDescriptorValueTa	IMF:SDDS_MSD(1.0).AGENCY.KEY

	rgetId	
Identifiable ObjectTarget	MSDAgencyId: MSDId(version).metadataTargetId.identifiable ObjectTargetId	IMF:SDDS_MSD(1.0).AGENCY.STR-OBJECT
DataSetTarget	MSDAgencyId: MSDId(version).metadataTargetId.dataSet TargetId	IMF:SDDS_MSD(1.0).AGENCY.D1101
PeportPeriod Target	MSDAgencyId: MSDId(version).metadataTargetId.reportPeriod TargetId	IMF:SDDS_MSD(1.0).AGENCY.REP_PER
ReportStructure	MSDAgencyId: MSDId(version).reportStructureId	IMF:SDDS_MSD(1.0).AGENCY_REPORT
Metadata Attribute	MSDAgencyId: MSDId(version).reportStructureId.metadataattri buteID	IMF:SDDS_MSD(1.0).AGENCY_REPORT.COMPILATION
Dataflow	dataflowAgencyId: dataflowId(version)	TFFS:CRED_EXT_DEBT(1.0)
Provision Agreement	provisionAgreementAgencyId:provisionAgreem entId(version)	TFFS:CRED_EXT_DEBT_AB(1.0)
Content Constraint	constraintAgencyId:ContentConstraintId(versio n)	TFFS:CREDITOR_DATA_CONTENT(1.0)
Attachment Constraint	constraintAgencyId: attachmentConstraintId(version)	TFFS:CREDITOR_DATA_ATTACHMENT_CONSTRAINT_ON E(1.0)
Metadataflow	metadataflowAgencyId: metadataflowId(version)	IMF:SDDS_FLOW(1.0)
StructureSet	structureSetAgencyId: structureSetId(version)	SDMX:BOP_STRUCTURES(1.0)
StructureMap	structureSetAgencyId: structureSetId(version). structureMapId	SDMX:BOP_STRUCTURES(1.0).TABLE1_TABLE2
Component Map	structureSetAgencyId: structureSetId(version). structureMapId. componentMapId	SDMX:BOP_STRUCTURES(1.0).TABLE1_TABLE2. REFAREA_REPCOUNTRY
CodelistMap	structureSetAgencyId: structureSetId(version).	SDMX:BOP_STRUCTURES(1.0).CLREFAREA_CLREPCOU

	codelistMapId	NTRY
CodeMap	structureSetAgencyId: structureSetId(version). codeListMapId. codeMapId	SDMX:BOP_STRUCTURES(1.0).CLREFAREA_CLREPCOUNTRY. DE_GER
Category SchemeMap	structureSetAgencyId: structureSetId(version). categorySchemeMapId	SDMX:BOP_STRUCTURES(1.0).SDMX_EUROSTAT
CategoryMap	structureSetAgencyId: structureSetId(version). categorySchemeMapId. categoryMapId	SDMX:BOP_STRUCTURES(1.0).SDMX_EUROSTAT.TOURISM_MAP
Organisation SchemeMap	structureSetAgencyId: structureSetId(version). organisationSchemeMapId	SDMX:BOP_STRUCTURES(1.0).DATA_PROVIDER_MAP
Organisation Map	structureSetAgencyId: structureSetId(version). organisationSchemeMapId. organisationMapId	SDMX:BOP_STRUCTURES(1.0).DATA_PROVIDER_MAP.IMF_1CO
Concept SchemeMap	structureSetAgencyId: structureSetId(version). conceptSchemeMapId	SDMX:BOP_STRUCTURES(1.0).SDMX_OECD
ConceptMap	structureSetAgencyId: structureSetId(version). conceptSchemeMapId. conceptMapId	SDMX:BOP_STRUCTURES(1.0).SDMX_OECD.COVERAGE_AVAI LABILITY
Reporting TaxonomyMap	structureSetAgencyId: structureSetId(version). reportingTaxonomyMapId	SDMX:BOP_STRUCTURES(1.0).TAXMAP
Reporting CategoryMap	structureSetAgencyId: structureSetId(version). reportngCategoryId	SDMX:BOP_STRUCTURES(1.0).TAXMAP.TOPCAT
HybridCodelist Map	structureSetAgencyId: structureSetId(version). hybridCodelistMapId.	SDMX:BOP_STRUCTURES(1.0).COUNTRY_HIERARCHYMAP
HybridCodeMap	structureSetAgencyId: structureSetId(version). hybridCodelistMapId. hybridCodeMapId	SDMX:BOP_STRUCTURES(1.0).COUNTRY_HIERARCHYMAP.CO DEMAP1
Process	processAgencyId: processId{version]	BIS:PROCESS1(1.0)
ProcessStep	processAgencyId:	BIS:PROCESS1(1.0).STEP1

	processId(version). processStepId	
Transition	processAgencyId: processId(version). processStepId transitioned	BIS:PROCESS1(1.0).STEP1.TRANSITION1
Subscription	订阅本身不是一个可识别的内容，因此它并不遵循 URN 结构规则，URN 的名称是 registryURN，它没有预先确定的格式。	当没有授权给维护代理创建订阅时，尽管可维护性一定意义上意味着能提交或删除订阅，但并不能由共同机制生成订阅，也不能进行版本控制。因此，目标注册表的责任是为订阅生成唯一 id,为应用程序创建订阅用以储存 registryURN，从订阅反馈信息的注册表中返回 registryURN。

6 实施指引

6.1 元数据的结构化定义

6.1.1 引言

SDMX注册表必须能够支持机构定义和分发结构元数据，包括数据结构定义、代码表，概念等，且在SDMX实施指南中有详细的说明。一个经认证的维护机构可能会提交有效“结构元数据”定义，这些定义必须存储在注册表中。注意，术语“结构元数据”属于所有结构元数据的通用术语（数据结构定义、元数据结构定义、代码表、概念方案等等）。

至少，存储于注册表中的结构元数据定义必须能够被HTTP/HTTPS GET以SDMX-ML结构报文方式实现访问，或者作为注册表接口的一部分被提交。建议使用SOAP协议（简单对象访问协议），该内容见《SDMX WEB服务指南》，同时也支持其他的协议。报文可包含用于整个注册表的所有结构元数据项，用于一个维护机构的结构元数据项以及单一结构元数据项。

结构元数据项：

只能由创建它的维护机构来修改；

只能被创建它的维护机构所删除；

当被注册表中其他结构引用时，不能被删除。

注册表中关于SDMX结构元数据对象维护的粒度水平是可维护的产物。换言之，任何函数，比如增加、修改、删除操作都是与此可维护产物在同一层的。例如，如果一段代码被添加到一个代码表中，或者一份代码的名称变了,注册表就会用已提交的含有维护机构、代码表ID以及版本的代码表代替原有代码表。

下面的表格（表7）列出了可维护类，描述了结构化定义元数据的可维护对象。

表 7

可维护的对象		内容
抽象类	具体类	
Item Scheme	Codelist	代码
	Concept Scheme	概念
	Category Scheme	目录
	Organisation Unit Scheme	组织单元

	Agency Scheme	代理
	Data Provider Scheme	数据提供者
	Data Consumer Scheme	数据使用者
	Reporting Taxonomy	报告分类
Structure	Data Structure Definition	维度描述符 组维度 描述符 维度 度量维度 时间维度 属性描述符 数据属性 度量描述符 主要度量
	Metadata Structure Definition	元数据目标 维度描述符 值的目标身份识别 对象目标 报告期目标 数据集目标 报表结构 元数据属性
Structure Usage	Dataflow Definition	
	Metadataflow Definition	
None	Process	处理步骤
None	Structure Set	组件图 概念格式图 代码列表图 目录格式图 报表分类标准图 组织格式图 概念图 代码图 目录图 组织图 报表分类图 混合代码列表图 混合代码图
None	Provision Agreement	
None	Hierarchical Codelist	层级 层级代码

6.1.2 项目方案、结构

结构化定义中包含的产物有：

——各种类型的项目方案（代码表、概念方案、类别方案、组织方案—机构方案，数据提供者方案，数据使用者方案，组织机构方案）；

——各种类型的结构（数据机构定义、元数据结构定义）；

——各种类型的结构用法（数据流定义、元数据流定义）。

6.1.3 结构用法

6.1.3.1 结构用法的基本概念

结构用法在数据流定义和元数据流定义的实体类中定义了一种对应关系，即哪些数据流和元数据流使用哪些特定的结构；并且重要地是为了支持数据和元数据的发现，结构用法可以被链接到一个或多个类别方案中的一个或多个类别中。这使得一个应用程序能够通过挖掘类别方案来发现数据和元数据。

6.1.3.2 结构用法 schematic。

结构用法见图9。

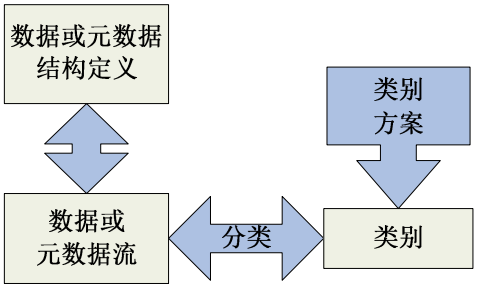

图 9 结构用法图

6.1.3.3 结构用法模型

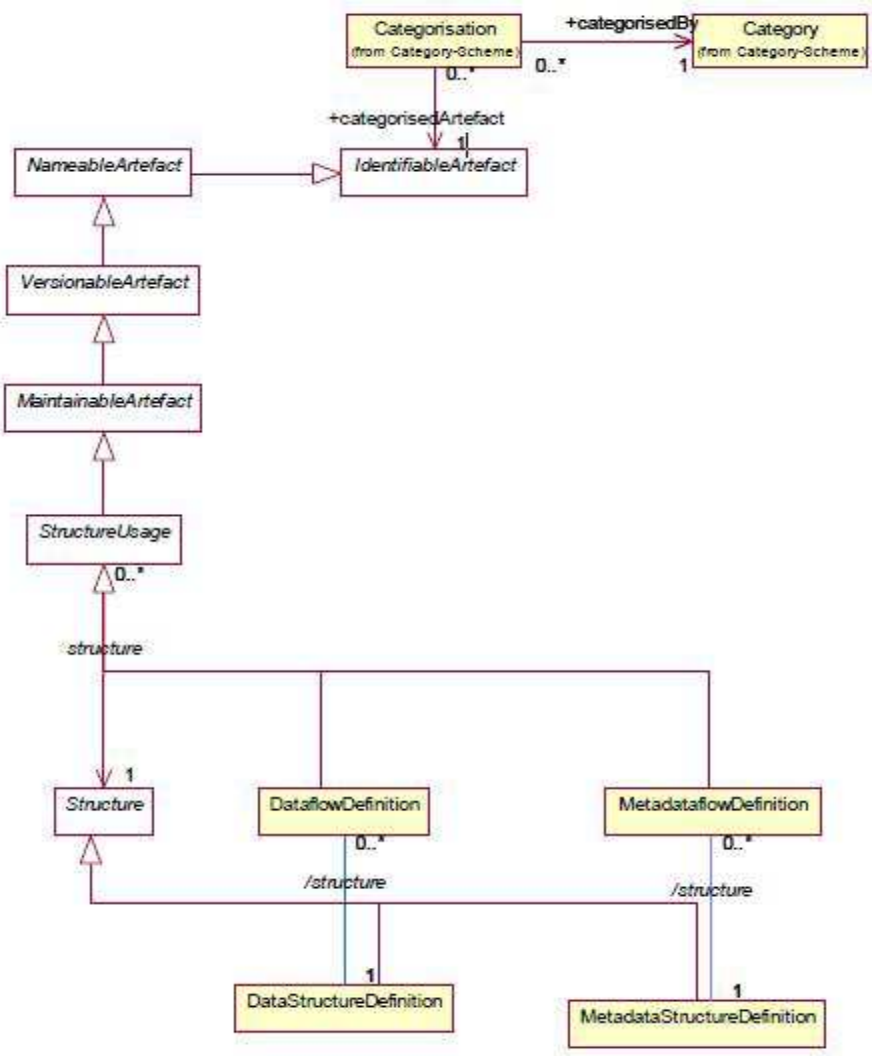

图 10 SDMX-IM 的结构使用和目录关联

除了数据流定义和元数据流定义的维护以外，在注册表中还要维护下列的链接：

——到数据结构定义的数据流定义；

——到元数据结构定义的元数据流定义。

以下链接靠分类的各种方法实现：

——数据流定义和类别的分类；

——元数据流定义和类别的分类。

6.2 数据和元数据供应

6.2.1 供应协议的基本概念

数据供应定义了一个框架，在这个框架中由各种数据提供者提供的不同类型的统计数据和元数据能够被指定和控制。有了这个框架做基础，才能让使用SDMX的组织了解到存在的数据，因此才能发现数据。这样一个框架能通过调整数据的内容来促进智能应用程序的建立。它也可以用于把服务等级协议，或其他供应协议加入到那些基于合法指令的方案中。另外，它还支持高质量和及时的信息，这使信息供应链的监控能够发挥实际作用。

注意，在SDMX信息模型（SDMX-IM）中，“数据供应商”类同时包括数据和元数据的供应商，而且“数据供应”在这里同时包括供应数据和元数据。

尽管，提供协议直接支持数据共享“Pull”模型，但它在“push”交换（双边和网关的情况），或发布环境中也同样有用。需要注意的是，在任何交换的情形下，注册表的作用相当于一个结构化元数据存储库。

6.2.2 配置协议模型-pull 用例

一个发布统计数据并希望让所发布的数据能够被已注册的SDMX团体使用的组织称为数据供应商。根据SDMX信息模型， 数据供应商维护数据提供者方案。

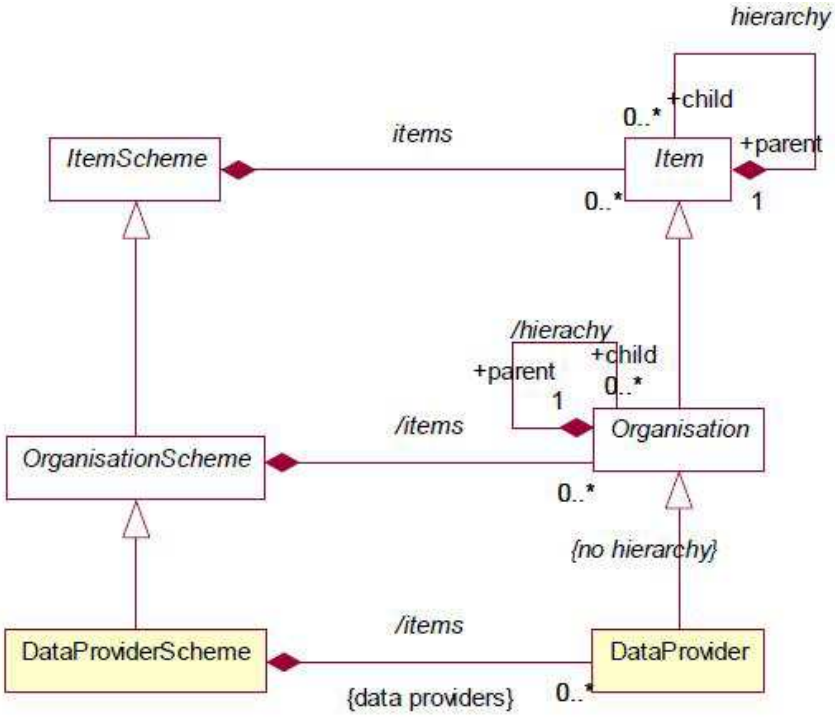

图11 SDMX-IM 数据提供者

注：数据提供者没有继承分层关系，下面是用来维护配置协议的数据模型类的逻辑原理图。

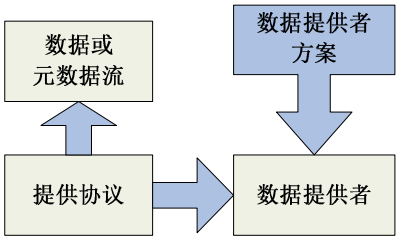

图 12 配置协议原理图

下图是用来维护提供协议的数据逻辑表示图。

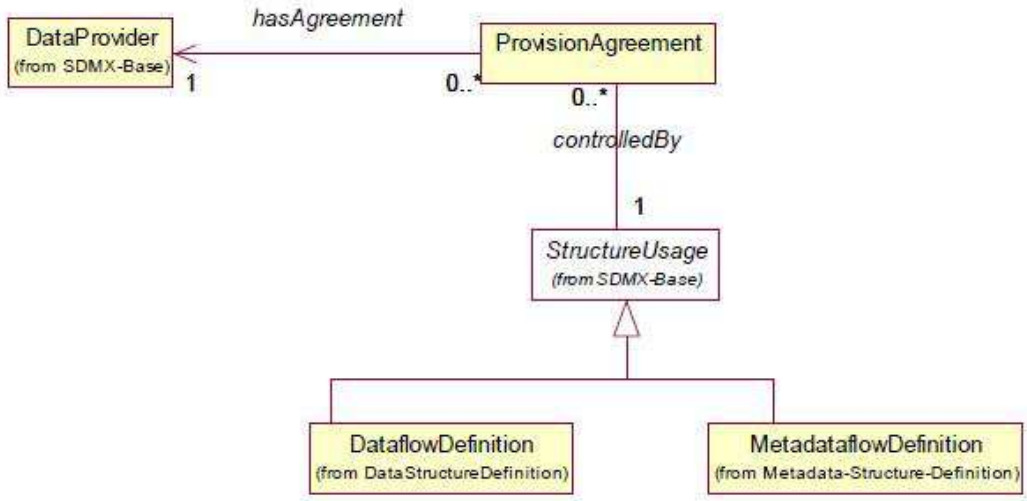

图 13 配置协议信息逻辑类图

提供协议是结构化元数据。每个提供协议必须引用数据提供者和数据流或元数据流定义。定义提供协议前，必须已存在数据提供者和数据流/元数据流的定义。

6.3 数据和元数据约束

6.3.1 数据和元数据约束：基本概念

数据约束是关于上述定义的数据供应产物（数据供应商，数据流和/或供应协议）的元数据，它限制和定义了这些产物。数据约束规范增强了数据供应产物的语义，使“信息供应链”过程的更加自动化。

数据约束包括囊括在数据源中或是数据流或元数据流定义提供的键值取值或属性取值的子集的规范。它是很重要的元数据，因为关键字族定义（如，有效值的完备集是代码表中所有取值在每一维度的笛卡尔积）中隐含的信息系统的可能性通常比实际出现在特定数据源中的信息系统的可能性要高，或者是比特定数据流定义要应用的信息系统的可能性要高。

通常一个数据供应商不能够为所有的关键字组合提供数据，一方面可能是因为该组合自身没有意义，另一方面可能仅仅是由于它没有该组合所需的数据。在这种情况下，数据供应商将通过提供定义关键字组合和可访问空间区域的元数据来限定数据源（在供应协议层次和数据供应商方面）。

而且，定义在代码表中限制了取值范围的关键字族的子集和视图通常是很有用的，特别是当很多这样的子集限制了该关键字族定义时。这样的一个视图被称作数据流定义，对于任意一个关键字族是可以定义一个或多个这样的视图。

无论数据何时被发布，或者数据供应商何时将其设置为可访问，它都必须与数据流定义保持一致（因此也须与关键字族保持一致）。因此数据流定义是基于内容处理的一种实现方式。

无论数据提供商什么时候发布数据或可用化数据，都必须进行数据流定义（随后是数据结构定义）。因此，数据流定义是认证内容一种处理方法。

而且，约束在数据可视化系统中非常有用，如网站上发布统计数据。在这样一个系统中，一个立方体可以指定实际在数据源中的维度码（可以用来编制相关选择表），并且关键集能指定数据源中的主键（可以用来指导用户选择维度码值，可以根据此维度值返回数据）。

6.3.2 数据和元数据约束：原理图

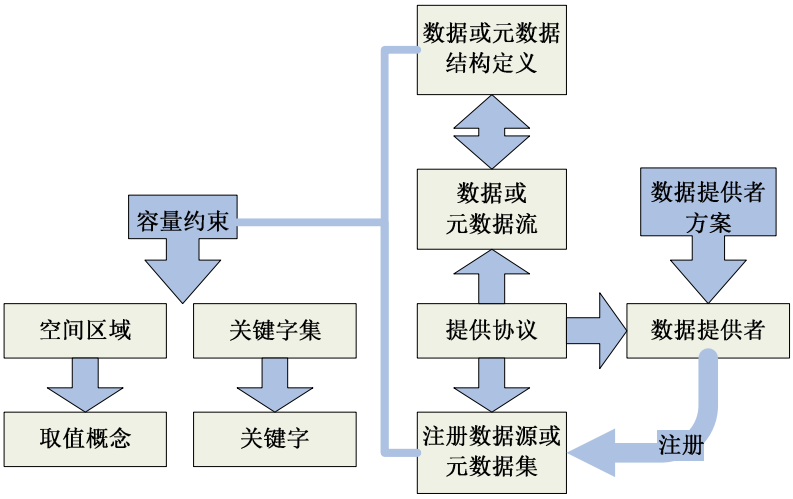

图14 约束和被约束内容原理图

6.3.3 数据和元数据约束：模型

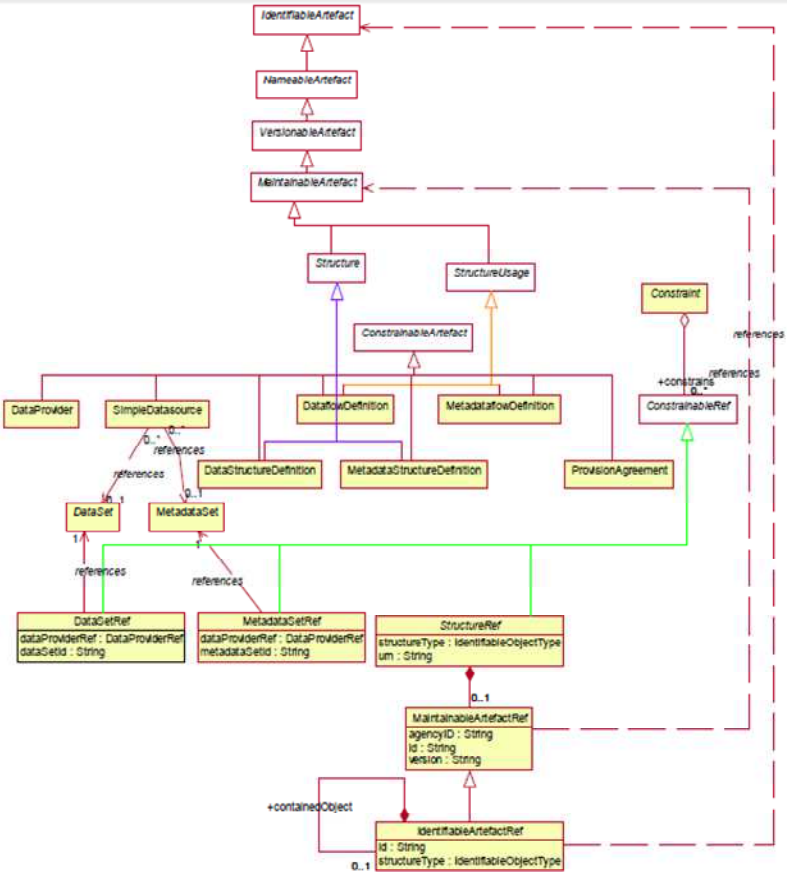

图15 显示继承关系并参考可约束的artifacts的逻辑类图

以上类图显示数据提供商、数据流定义、元数据流定义、提供协议、数据结构定义、元数据结构定义、简单数据源和查询数据源都是ConstrainableArtefact的具体子类，因此都有特定的约束。注意，实际提交的约束与参考类有联系，约束继承了ConstrainableRef：这些都用来关联约束适用的类。

关于约束的相关知识点，可以参见SDMX信息模型文档。

6.4 数据和元数据注册

6.4.1 基本概念

数据供应商发布了与已存的数据流定义一致（因而属于关键字族）的数据集。这既可以以一个网络可访问的SDMX-ML文件的形式完成，也可以在一个具有网络服务接口的数据库中完成，该网络服务接口要能够对一个SDMX-ML查询响应一个SDMX-ML数据流。

数据供应商希望不用实际发布数据而让新数据拥有更多的读者。为了做到这一点，数据供应商用一个或多个满足SDMX规范的注册表来注册这一新的数据集，所用的注册表已经在用结构化的供应元数据进行了供应。换句话来讲，注册表“认识”数据供应商并且“知道”什么数据流是供应商同意让它被访问的。

同样的机制可以用来报告或提供元数据集。

SDMX-RR支持通过Registration Request（注册请求）进行数据集和元数据集的注册，Registration Request（注册请求）能够由数据供应商创建（须给数据供应商最高控制权限），也能够由注册表服务代表数据供应商来生成。注册表通过注册响应来回应注册请求，有注册响应表示已注册成功。如果出现错误，则返回注册异常。

6.4.2 注册请求

6.4.2.1 注册请求原理

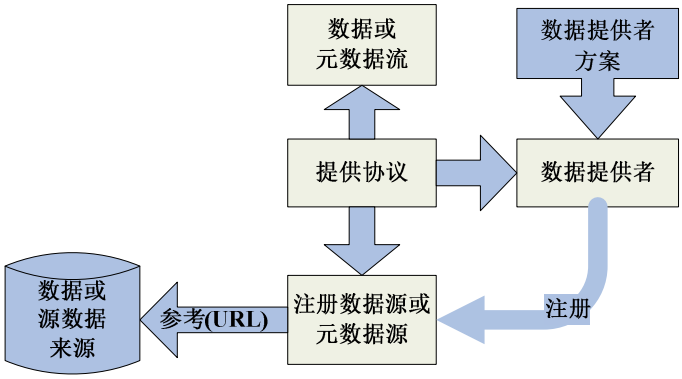

图16 对象注册原理图

6.4.2.2 注册请求模型

下面的UML图展示了注册请求的构成，每一个请求都是由一个或多个注册（Registration）和一个待注册的数据集组成。每个数据集或元数据集都要注册。可以选择性的从注册表提取信息：

——ValidFrom；

——validTo；

——lastUpdated。

注册有一个行为属性，该属性从下面三个值中取值：见表8。

表 8

操作属性值	操作
追加	在注册表中增加注册
替换	识别提交的注册请求中所包含的注册信息 id，并替换已存在的相同 id 的注册信息
删除	识别提交的注册请求中所包含的注册信息 id，并在注册表中删除相同 id 的注册信

	息

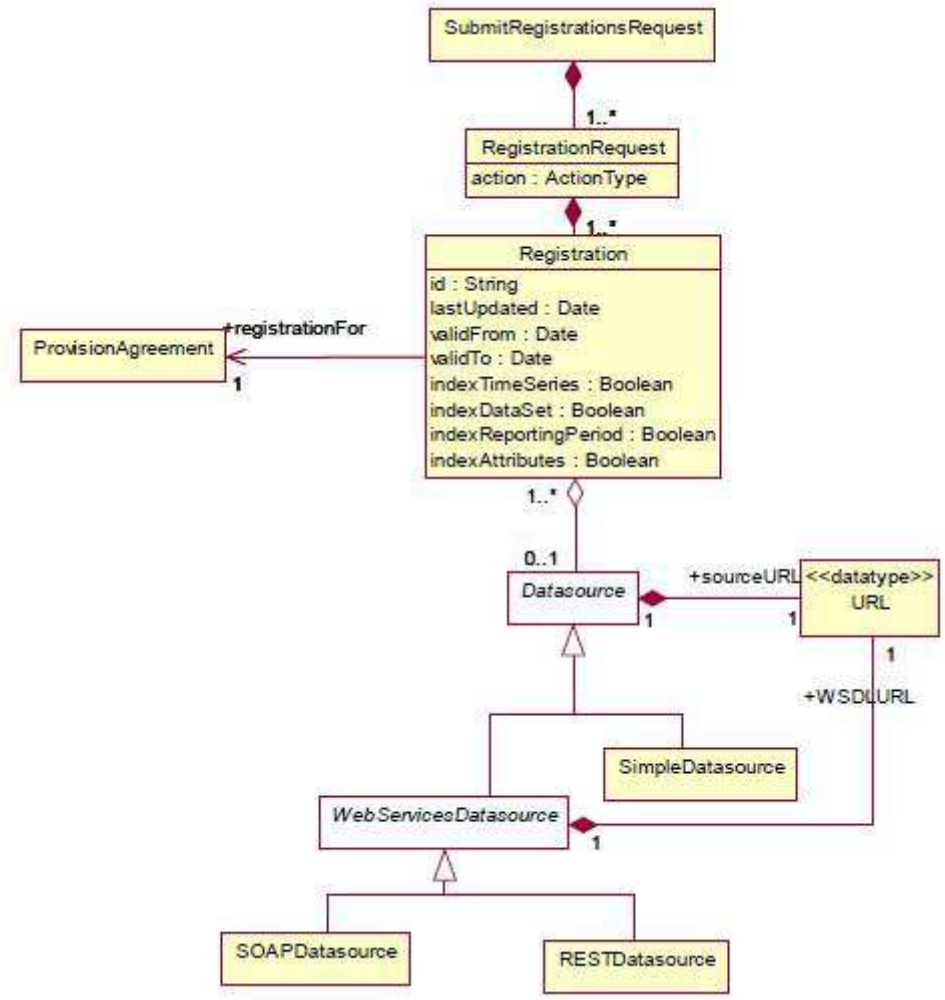

图17 数据、元数据注册的逻辑类图

查询数据源(Query Datasource)是一个代表数据源的抽象类，该数据源能解释SDMX-ML查询（SOAPDatasource）或RESTful查询（RESTDatasource），并能做出正确的响应。每个不同的数据源都从数据源中继承了dataURL，QueryDatasource拥有一个额外的URL来指向一份WSDL或WADL文档，此文档描述如何访问数据源。所有其他支持协议被假定为使用简单数据源URL。

简单数据源可以参考从URL访问的SDMX-ML文件的物理位置。

注册请求中有一个用来定义是否是增加、替换或删除的注册的操作属性，注册请求中仅提供要替换或删除的注册id，对于新增的注册，注册表会为它分配唯一的id。

注册还有一个属性用来说明，简单数据源(Simple Datasource)在注册时是如何索引的。注册表中进行注册必须按照如下方式进行：

数据或元数据集信息被抽出放在一个或多个内容约束中（参见SDMX信息模型中的约束模型—SDMX标准第2章）。抽取出的信息在配置协议中以布尔值集方式显示，如表9所示：

表 9

需要的索引	注册过程
indexTimeSeries	抽取所有关键字序列，并创建关键字集约束
indexDataSet	抽取所有代码以及数据集中其它关键字值内容，并

	创建一个或多个包含 SDMX-IM 模型维度组件成员选项的立方体和相关的选项值。
indexReportingPeriod	这仅适用于已注册的数据集。从包含数据集报文头部抽取报告开始和结束，并定义参考期约束。
indexAttributes	**数据集** 抽取数据集中的属性值，并创建一个或多个包含 SDMX-IM 约束模型数据属性组件成员选项的立方体和相关的选项值。 **元数据集** 通过创建一个或多个包含 SDMX-IM 约束模型元数据属性组件成员选项的立方体，来表示已报告属性的存在。

指定查询数据源(Query Datasource)内容的约束要提交到提交结构请求(Submit Structure Request)的注册表中。

注册必须参考相关的提供协议。

6.4.3 注册响应

在一个注册请求提交给注册表之后，将会返回给提交者成功或失败的响应信息。假定一个注册请求能够同时申请多个注册动作（Registrations），则对应地，必须给每一个请求一个回应。注册响应（Registration Result）类有一个状态域，该域只会设置为“成功”或者“失败”两种状态。

如果某次注册成功了，一个供应协议参考类（ProvisionAgreementRef，ConstrainableRef的子类）将会被返回——这个类将包含新注册的数据集的URN,还有具有一个能够访问该数据集和元数据集的URL链接的数据源（Datasource）信息。

注册会根据注册消息反馈一系列注册状态（每个注册项一个），以标识成功或失败. 当注册失败时，一组注册异常信息（Registry Exceptions）就会被返回，它将给出注册过程中的出错信息。当注册一批数据集的时候，完全有可能出现这种情况：注册响应中同时包含一些成功信息和一些失败信息。注册响应的UML（统一建模语言）描述如下：

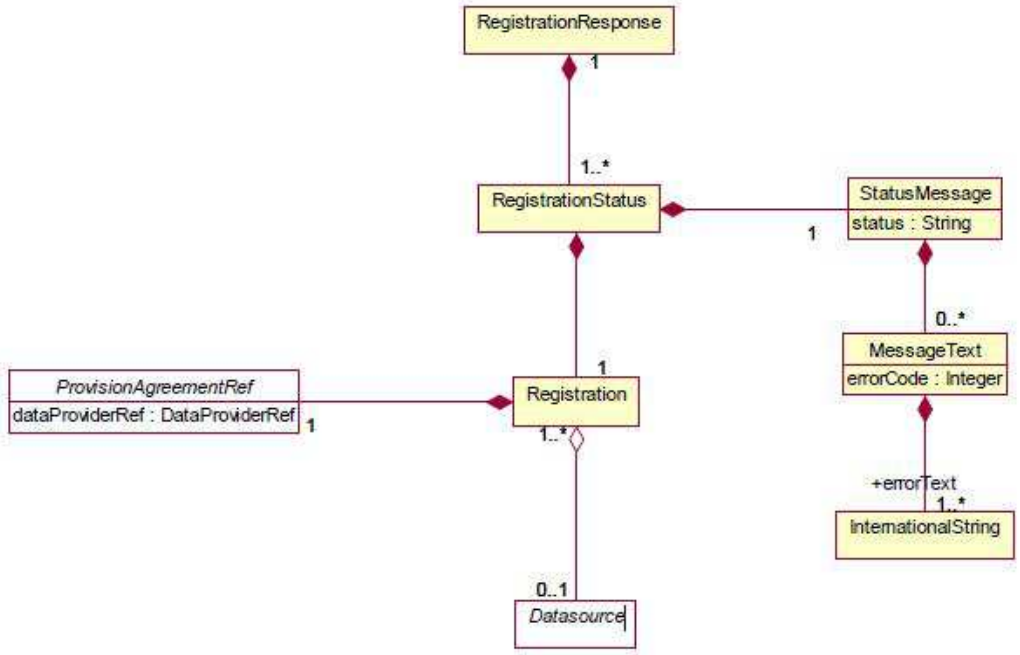

图18 注册反馈逻辑类图

6.5 订阅和通知服务

SDMX注册表/存储库的内容会经常发生变化：新的代码表和关键字族会被发布，新的数据集和元数据集会被注册。为了使用户不必多次重复地查询注册表是否有新的可用信息，相关机制会在这些情况发生时给用户自动提供通知。

一个用户能够在注册表中存储一个订阅，该订阅可由用户定义了哪些事项是其感兴趣的，以及当有符合要求的事项发生时进行通知投递的email或者HTTP地址。在注册表中每个订阅都会被一个URN(统一资源名称)标识，这个标识会在用户申请创建该订阅时返回给用户。如果之后用户想删除该订阅，这个URN会被用作一个标识符(来删除它)。订阅有一个有效期限，该期限用一个持续日期来表示(startDate，endDate)，注册表可以自由地删除过期的订阅，并会通知订阅者其订阅已过期。

当一个注册表/存储库产物被修改时，任何监视该对象的订阅都会被激活，此时一封email或者HTTP POST会被投递，以便将变化的细节报告给订阅中所指定的用户，这就叫做一个“通知”。

6.5.1 订阅逻辑类图

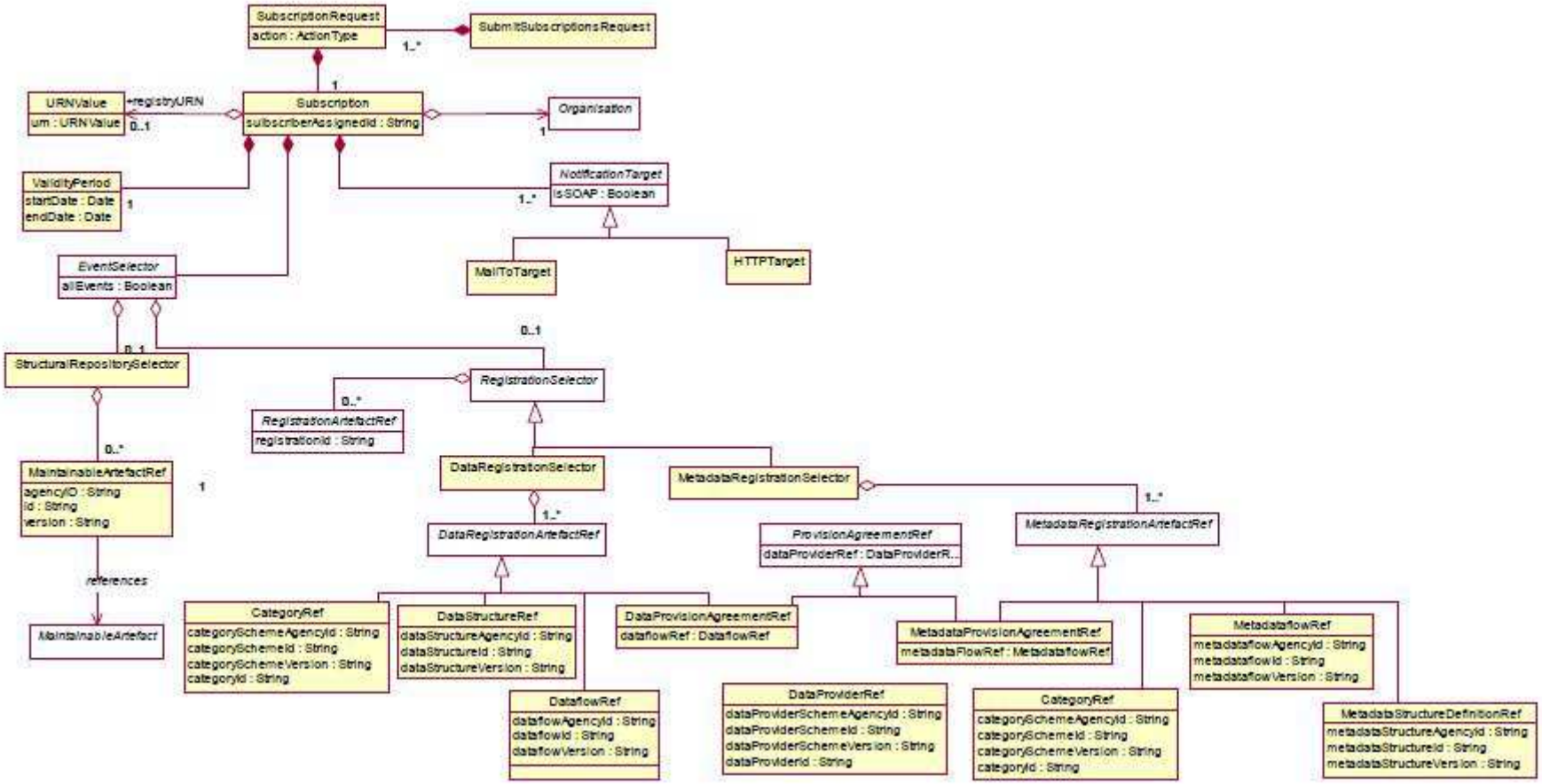
SubscriptionRequest
action : ActionType
SubmitSubscriptionsRequest
1..*
1
URNValue
urn : URNValue
+registryURN
0..1
Subscription
subscriberAssignedId : String
Organisation
1
ValidityPeriod
startDate : Date
endDate : Date
1
NotificationTarget
isSOAP : Boolean
1..*
MailToTarget
HTTPTarget
EventSelector
allEvents : Boolean
0..1
0..1
StructuralRepositorySelector
RegistrationSelector
0..*
RegistrationArtefactRef
registrationId : String
0..*
MaintainableArtefactRef
agencyID : String
id : String
version : String
1
DataRegistrationSelector
MetadataRegistrationSelector
1..*
DataRegistrationArtefactRef
ProvisionAgreementRef
dataProviderRef : DataProviderR..
1..*
MetadataRegistrationArtefactRef
references
MaintainableArtefact
CategoryRef
categorySchemeAgencyId : String
categorySchemeId : String
categorySchemeVersion : String
categoryId : String
DataStructureRef
dataStructureAgencyId : String
dataStructureId : String
dataStructureVersion : String
DataProvisionAgreementRef
dataflowRef : DataflowRef
MetadataProvisionAgreementRef
metadataFlowRef : MetadataflowRef
MetadataflowRef
metadataflowAgencyId : String
metadataflowId : String
metadataflowVersion : String
DataflowRef
dataflowAgencyId : String
dataflowId : String
dataflowVersion : String
DataProviderRef
dataProviderSchemeAgencyId : String
dataProviderSchemeId : String
dataProviderSchemeVersion : String
dataProviderId : String
CategoryRef
categorySchemeAgencyId : String
categorySchemeId : String
categorySchemeVersion : String
categoryId : String
MetadataStructureDefinitionRef
metadataStructureAgencyId : String
metadataStructureId : String
metadataStructureVersion : String

图 19 订阅逻辑类图

6.5.2 订阅信息

无论监视什么类型的注册表/存储库事件，一个订阅总是包括：

——一些通用资源标识符（URI）的集合，它们描述了当订阅被激活之后通知必须要送到哪些结点。这些 URI 可能是电子邮件地址或者是 HTTP 协议地址。如果是前者就会发一封电子邮件，如果是后者就会发一封 HTTP POST 通知；

——一个在订阅请求响应中的由用户定义的标识符，这可以用于异步进程；

——一个有效时段，这规定了订阅何时生效及何时过期。当其过期之后，订阅者将会收到一份通知；

——一个选择器，它指定了哪些类型的事项是感兴趣的。事项类型的集合如下（见表 10）：

表 10

事件类型	解释
STRUCTURAL_REPOSITORY_EVENTS	结构化元数据库中可维护内容的生命周期变化
DATA_REGISTRATION_EVENTS	已发布的数据集可以随时注册。数据集可以是 SDMX-ML 数据文件或符合 SDMX 的数据库
METADATA_REGISTRATION_EVENTS	已发布的数据集可以随时注册。数据集可以是 SDMX-ML 参考元数据文件或符合 SDMX 的数据库
ALL_EVENTS	EventType 中指定的所有事件

6.5.3 通配符

订阅通知支持通配符标识符组件 URN，标识符中的一部分或全部可以用通配符%来代替。这些标识符为：

——agencyID

——id

——version

Codelist 中识别对象类型中可以通配的标识符，其举例如下：

AgencyID = %

Id = %

Version = %

上例是为订阅了所有代理维护的所有版本的代码表。

AgencyID = AGENCY1

Id = CODELIST1

Version = %

上例为订阅了由代理 AGENCY1 维护的 Codelist CODELIST1 的所有版本

AgencyID = AGENCY1

Id = %

Version=%

上例为订阅了由代理 AGENCY1 维护的所有 Codelist 对象的所有版本

AgencyID=%

Id=CODELIST1

Version=%

上例为订阅了由代理 AGENCY1 维护的 Codelist CODELIST1 的所有版本

注意，如果订阅了最新版本，则可以用字符*表示

如 Version=*

注意，使用 URN 机制的订阅不能使用通配符。

6.5.4 结构化数据库通知

这种通知将包含触发该事件（数据结构定义、概念表、代码表、元数据结构定义、分类表等）的可维护类（MaintainableArtefact）。

6.5.5 注册通知

这种通知将包含注册类（Registration），它含有最后更新时间（lastUpdated），有效期开始时间（validFrom），有效期结束时间（validTo）等属性。该通知同样包含任何以URL形式与该注册联系的数据源（Datasource）。同时可以选择性地将其约束一并返回（在某些情况下它们是非常大的。）

6.6 通知

6.6.1 通知类图

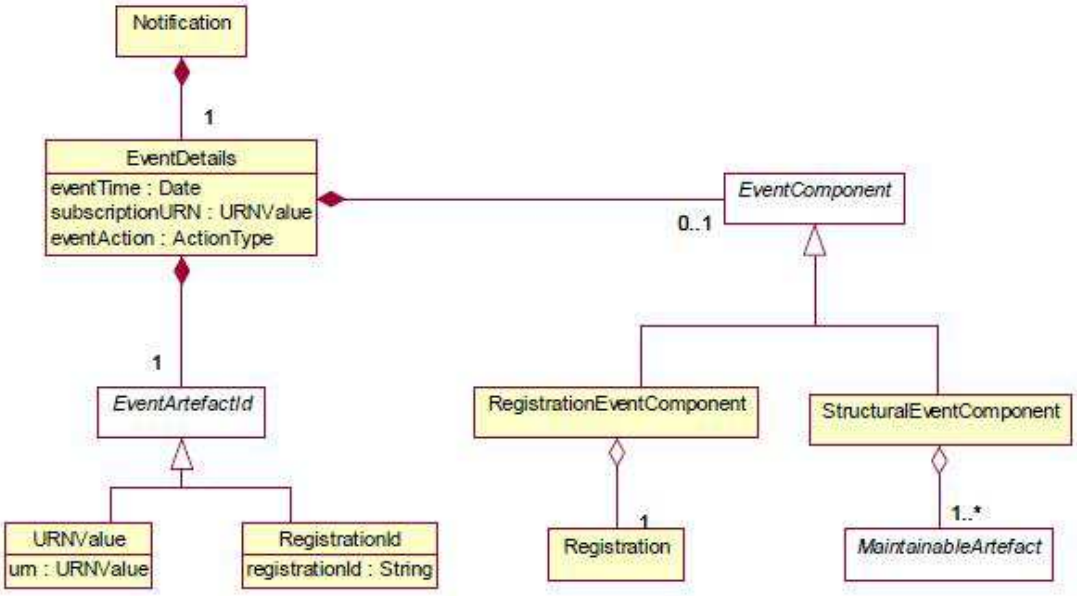

图 20 通知逻辑类图

当订阅激活后就会通过邮件或 http 端口将通知以 XML 文档的形式发送给用户，这是一个异步单向的信息。

无论导致事件被触发的注册组件是什么，以下公共信息都会在消息中：

——事件发生的日期和时间；

——导致事件发生的对象的 URN；

——产生通知的订阅的 URN；

——事件操作：增加、替换，或删除。

除此之外，通知中还会包含一些补充资料。

6.6.2 结构化事件组件

通知包含触发事件的 Maintainableartefact，通知的格式和 SDMX-ML 结构化信息（使用元素命名空间）比较类似。

6.6.3 注册事件组件

通知包含注册。