ICS 71.040.10
N 53

中华人民共和国国家标准

GB/T 38113—2019

分析仪器物联规范

Specification for internet of things for analytical instruments

2019-10-18 发布　　2020-02-01 实施

国家市场监督管理总局
中国国家标准化管理委员会　发布

目　　次

前　言

本标准按照 GB/T 1.1—2009 给出的规则起草。

本标准由中国机械工业联合会提出。

本标准由全国工业过程测量控制和自动化标准化技术委员会(SAC/TC 124)归口。

本标准起草单位:上海市计算技术研究所、上海上科信息技术研究所、上海舜宇恒平科学仪器有限公司、杭州市中辉科学器材有限公司、北京雪迪龙科技股份有限公司、清谱(上海)分析仪器有限公司、山东山宇环境科技有限公司、长春吉大·小天鹅仪器有限公司、北京华夏科创仪器技术有限公司、上海软中信息技术有限公司、汉威科技集团股份有限公司、广州讯动网络科技有限公司、重庆创晖科技有限公司、南京分析仪器厂有限公司、南京霍普斯科技有限公司、钢研纳克检测技术有限公司、上海伍丰科学仪器有限公司、聚光科技(杭州)股份有限公司、上海磐合测控技术股份有限公司、深圳市麦斯达夫科技有限公司、苏州市计量测试院、河北先河环保科技股份有限公司、恩德斯豪斯(中国)自动化有限公司、上海纽钛测控技术有限公司、中国仪器仪表行业协会和上海产业技术研究院。

本标准主要起草人:张敬周、李钧、王志宏、吴华忠、郜武、王世立、董占勇、高德江、张新民、宋俊典、陈海永、陈新泉、郑杰、刘虎、陈海、顾潮春、赵英飞、徐伯元、赵忠欣、徐国平、杨玺、郑波、任豪、杨建虎、谭海玲、袁旭军、马雅娟、袁满满。

引　言

分析仪器种类多样、智能化水平各有差异，分析仪器相关的大数据和智能化应用需要一个通用的框架，来实现各类分析仪器的网络化测控、分析、共享、协同、运维、管理和数据服务等，本标准就提供了这样一个通用框架。

实施本标准，可：

——方便地实现不同分析仪器之间的联动、数据交互和共享；

——提高分析仪器开发、生产、管理和使用活动中 IT 部分的复用度，提高产品的智能化水平；

——减少分析仪器相关 IT 应用系统或平台的开发、运维和服务成本，降低项目实施风险；

——提高分析仪器相关的大数据建设的效率，提高数据管理、数据质量和大数据应用水平。

分析仪器物联规范

1 范围

本标准规定了分析仪器的术语和定义、缩略语、约定、物联框架、物联过程、物联模型、数据交换过程以及相关的形式化表达方法。

本标准适用于指导分析仪器物联相关的信息系统设计、开发、运维及数据服务等活动。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 13000—2010 信息技术 通用多八位编码字符集(UCS)

GB/T 13966—2013 分析仪器术语

3 术语和定义、缩略语

3.1 术语和定义

GB/T 13966—2013 界定的以及下列术语和定义适用于本文件。

3.1.1

实体分析仪器 entity analytical instrument

实体仪器

物联至网络的分析仪器。

3.1.2

虚拟分析仪器 virtual analytical instrument

虚拟仪器

实体仪器在网络中的代理。

3.1.3

仪器映射器 analytical instrument mapper

映射器

实现实体仪器与虚拟仪器之间数据绑定的软件密集型系统。

3.1.4

分析仪器物联 analytical instrument becoming things of internet

通过网络实现对分析仪器的测量分析、控制协同、数据共享及管理等。

注：一台分析仪器实现物联后应包含三个部分：实体仪器、虚拟仪器和映射器。

3.1.5

软件服务 software services

实现特定功能、通过访问接口在线响应外部调用请求的软件密集型系统。

3.1.6

仪器注册中心　analytical instrument registry

注册中心

为各分析仪器提供物联信息的发布、发现等服务的软件服务。

3.1.7

物联对象　objects of internet of things for analytical instruments

对分析仪器物联涉及的组件的统称。

注：主要有实体仪器、虚拟仪器、映射器和注册中心等。

3.1.8

物联标准件　formal objects of internet of things for analytical instruments

提供标准化接口、可网络化访问的分析仪器物联对象。

注：主要有虚拟仪器、注册中心。

3.1.9

物联信息模型　information models of internet of things for analytical instruments

物联模型

对描述分析仪器物联相关信息的若干个信息模型的统称。

注：主要分为四类：仪器信息模型、注册中心信息模型、访问接口信息模型和元数据信息模型。

3.1.10

物联默认模型　default information models of internet of things for analytical instruments

对描述分析仪器物联相关信息的共性部分的若干个信息模型的统称。

注：主要包括四个：仪器默认模型、注册中心默认模型、访问接口默认模型和元数据默认模型。特定的物联模型均应扩展自同类型的物联默认模型，如仪器模型应扩展自仪器默认模型。

3.1.11

仪器信息模型　information models of analytical instruments

仪器模型

用于描述虚拟仪器可对外交互的信息的数据模型。

3.1.12

仪器注册中心信息模型　information models of analytical instruments registry

注册中心模型

用于描述仪器注册中心的数据模型。

注：主要包括四类：注册中心入口信息模型、仪器黄页库信息模型、物联剖面库信息模型和元数据库信息模型。

3.1.13

访问接口信息模型　information model of access interface

访问接口模型

用于描述物联标准件提供的访问接口的数据模型。

3.1.14

元数据信息模型　information model of metadata

元数据模型

用于描述元数据的数据模型。

3.1.15

数据规约语言　data specification language

DSL 语言

用于描述分析仪器物联模型、剖面、元数据和访问接口等的形式化语言。

注：具体形式化定义参见附录 A。

3.1.16

剖面　profile of an information model

采用 DSL 语言、对信息模型的一个信息子集的形式化表达。

注：用于实现对该模型的特定共享目的。

3.1.17

模型主剖面　main profile of an information model

主剖面

采用 DSL 语言、对信息模型的信息全集的形式化表达。

3.1.18

物联剖面　profiles of internet of things for analytical instruments

对分析仪器物联信息模型的剖面的统称。

注：主要分四类：仪器剖面、注册中心剖面、访问接口剖面和元数据剖面。

3.1.19

物联默认剖面　default profiles of internet of things for analytical instruments

默认剖面

物联默认模型的主剖面。

3.1.20

默认访问接口　default access interface

由物联标准件提供的、符合访问接口默认剖面规定的对外访问接口。

3.1.21

模型的主类　main class of information model

主类

物联信息模型中的一个类，其名称作为相应物联剖面的键值对表达式中的“键”。

3.1.22

父/子模型　parent/child information model

在已有物联模型基础上可以扩展形成新的信息模型，新生成的信息模型为已有信息模型的子模型，已有信息模型为子模型的父模型。

3.1.23

键值对　key-value pair

由“键”和“值”通过符号“：”串接形成的一个字符串形式的数据表达式。

注：其中“键”是键值对的标识符，“值”是“键”对应的数值。

3.1.24

元数据　metadata

定义和描述其他数据的数据。

3.1.25

基础元数据　primary metadata

原子性的元数据。

注：在保持元数据语义不变的前提下，基础元数据不能再分解为其他元数据。

3.1.26

复合元数据　aggregate metadata

由一个或多个已定义的元数据，按一定的生成规则形成的元数据。

3.1.27

父/子元数据　parent/child metadata

由若干已定义的元数据生成新的元数据时，新生成的元数据为已定义元数据的父元数据，已定义的元数据为新生成的元数据的子元数据。

3.1.28

数值结构　value structure

用于描述物联剖面或元数据的数值构成及其类型约束的数据结构。

3.2　缩略语

下列缩略语适用于本文件。

DSL：数据规约语言(Data Specification Language)

ID：标识符(Identifier)

UML：统一建模语言(Uniform Modeling Language)

URI：统一资源标识符(Uniform Resource Identifiers)

4　约定

4.1　文档约定

表达形式方面遵循如下约定：

a) 类名。类名称以大写字母开头，字体加粗；类名称在文档第一次出现时，类中文名称放在英文名称之后并用括号“()”括起。

b) 属性名。属性名以用小写字母开头，字体为斜体；属性名称在文档第一次出现时，类中文名称放在英文名称之后并用符合“:”隔开。

c) 类或属性的名称若由多个单词组成，则在单词之间使用连字符“-”串接，连字符后面的单词以大写字母开头，如“product-Info”。

d) 键值对：键值对以“″键″:″值″”形式表示，若值部分为复合表达式，则值部分用符号对“{}”括起。

4.2　UML 建模约定

物联模型部分采用 UML 语言表达，并遵循如下约定：

a) 标识符属性。模型中类的标识符属性统一用 *id* 作为其英文名称。

b) 关联关系。本标准的模型中类与类之间主要有两种关联关系：组合关系、聚合关系。组合关系表示拥有者类包含了被组合的类的整体；聚合关系表示拥有者类包含被聚合的类的标识符属性 *id*。

c) 基数。类之间的关联基数使用“下限..上限”的表示方式。在上限为无限时，使用“*”表示；若基数为“0..*”，则可简写为“*”；若基数中的下限与上限相同，则可简写为一个数字，如基数为“1..1”，则可简写为“1”。

4.3　形式化表达

物联模型、物联剖面、元数据、访问接口及其具体的数据交换等，均采用 DSL 语言来表达。DSL 的形式化定义见附录 A。

5 物联框架

5.1 概述

本标准定义了分析仪器物联的三层架构：实体仪器层、映射层和虚拟仪器层，如图 1 所示。

实体仪器层的物联对象为各类具有特定数据通信接口的实体仪器。

映射层的物联对象为各类映射器，目的是建立实体仪器与虚拟仪器之间的动态映射，实现两者之间的数据绑定。

虚拟仪器层的物联对象主要为各物联标准件，主要包括各类虚拟仪器和注册中心。虚拟仪器是实体仪器的一个在线网络代理，对实体仪器的操控和共享通过对虚拟仪器的标准化访问来实现；仪器注册中心提供分析仪器的网络发布、发现服务。

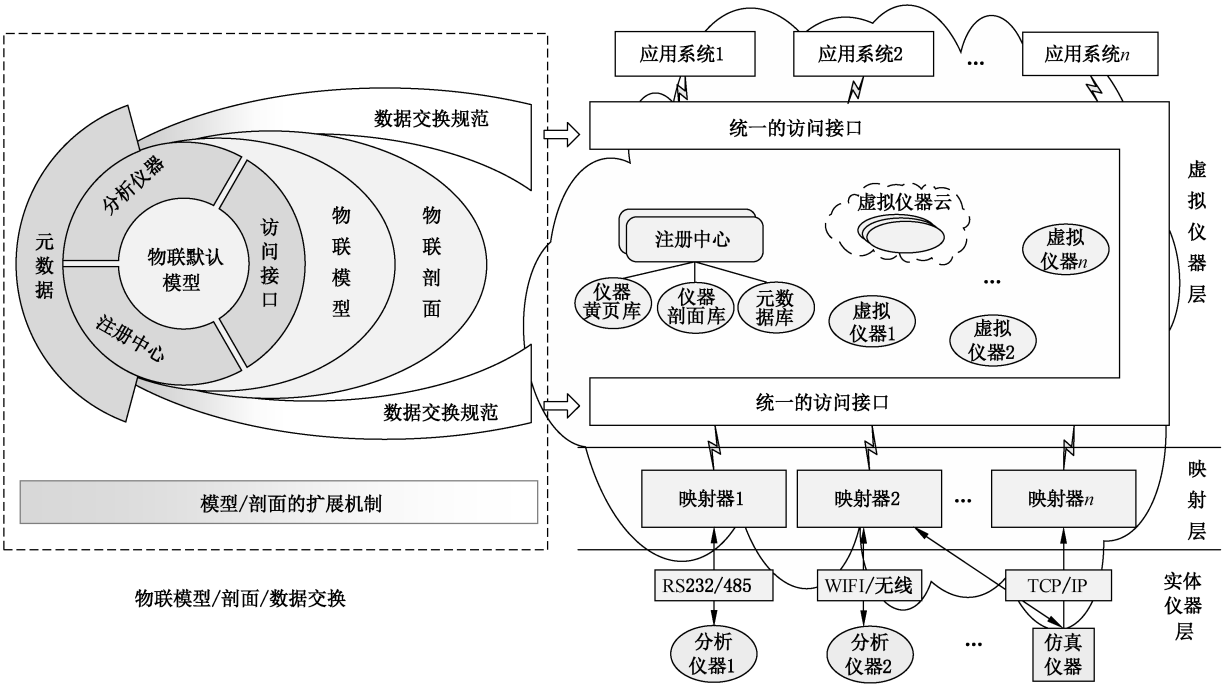

图 1 分析仪器物联总体框架

物联模型定义了分析仪器物联相关信息的数据结构和语义。其中：仪器模型描述虚拟仪器中的可共享数据；注册中心模型描述分析仪器网上发布、发现时所需的信息；访问接口模型描述虚拟仪器的访问接口；元数据模型描述分析仪器物联所用的元数据。本标准采用 UML 给出了物联默认模型的定义。

扩展机制定义从已有物联模型生成新模型时应遵循的约束。基于物联默认模型，可扩展得到各类分析仪器所需的特定物联模型。

物联剖面规约了物联标准件的对外数据交换。访问接口剖面规约了接口交互的数据帧结构，仪器剖面/注册中心剖面规约了数据帧内的数据内容。物联标准件均提供由物联默认剖面规约的默认访问接口，通过默认访问接口可获取其特定的物联信息，从而实现对其个性化的访问。

本标准定义的 DSL 语言为物联模型/剖面、元数据、访问接口和交互数据帧提供了统一的形式化表达方法。

本标准通过上述的架构、模型、扩展机制和形式化方法，为各类分析仪器物联提供了一个通用的框架。

5.2 实体仪器层

实体仪器层的物联对象为各类实体仪器。

对实体仪器的数据通信接口不作规定。

5.3 映射层

映射层中的映射器负责实体仪器与其虚拟仪器的数据绑定，绑定的数据由虚拟仪器的仪器剖面来规约。映射层可支持实体仪器与虚拟仪器之间的多对多映射关系。

映射器应实现以下功能：

a) 与实体仪器的交互。获取或设置实体仪器的可共享信息。

b) 与虚拟仪器的交互。将实体仪器的共享信息更新到虚拟仪器中；获取虚拟仪器对实体仪器的可设置信息；可实现对虚拟仪器的管控，如：虚拟仪器的创建、发布、开通/关闭、注销等。

c) 对上述交互数据进行必要的转换处理。保证实体仪器和虚拟仪器之间数据的语义一致性、同步的实时性。

对映射器的数据接口、交互方式和具体实现不作规定。

5.4 虚拟仪器层

5.4.1 虚拟仪器

虚拟仪器是实体仪器在网络中的一个在线代理，通过标准访问接口实现对外数据共享，共享数据由仪器模型的主剖面来规约，分为以下部分：

a) 实体仪器数据。实体仪器的生命周期相关的数据，主要包括：实体仪器的基本信息、状态与事件、控制与协同、测量分析过程、运维管理等的实时和历史数据。该部分主要通过映射器的数据绑定来产生。

b) 数据共享方法的描述。描述虚拟仪器对外如何进行数据共享的数据，主要包括：虚拟仪器可共享数据的范围、结构和语义的描述，如虚拟仪器的物联剖面、元数据等；虚拟仪器对外数据共享方式的描述，如虚拟仪器的访问接口、网络发布发现的数据；描述虚拟仪器提供数据共享时的服务质量的数据，如权限控制、共享模式、数据质量等。该部分主要由虚拟仪器来管理。

c) 数据共享的过程记录。对上述数据的交互过程的记录与统计数据。该部分主要由虚拟仪器来产生和管理。

虚拟仪器的实现应满足以下要求：

a) 提供符合访问接口默认剖面(见附录 B)规定的访问接口。

b) 实现与相应实体仪器的数据绑定，绑定的数据内容符合仪器主剖面的规定；仪器主剖面应直接或间接扩展自仪器默认剖面，扩展机制符合 7.4 的规定。

c) 提供的访问接口应符合其访问接口剖面的规定；接口交互的数据结构语义、服务质量应符合其仪器剖面的规定。

d) 各物联剖面、访问接口、元数据之间具备一致性、完整性。

对虚拟仪器的内部架构、运行平台和具体实现不作规定。

5.4.2 注册中心

仪器注册中心是一个在仪器物联域内可公开访问的软件服务，为分析仪器物联提供信息发布、发现功能。主要有以下组成部分：

a) 仪器黄页库。管理存储各实体仪器、虚拟仪器的基本信息及其映射关系。

b) 物联剖面库。管理存储分析仪器物联的各类剖面。

c) 元数据库。存储管理分析仪器物联所用的各类元数据。

注册中心的实现应满足以下要求：

a) 提供符合访问接口默认剖面(见附录 B)规定的访问接口。

b) 按注册中心默认剖面(见附录 B)规定的信息要求，实现分析仪器、物联剖面、物联元数据在注册中心的信息发布和发现。

c) 保证注册中心管理的各物联剖面、元数据的一致性和完整性。

本标准对仪器注册中心的内部架构、运行平台和具体实现不作规定。

6 物联过程

6.1 发布过程

仪器提供者根据实体仪器的物联要求，建立可供网络用户访问的虚拟仪器，称为仪器发布过程。主要包括以下活动：

a) 数据共享的需求分析。分析确定实体仪器对外共享的数据集合；分析确定数据共享的方式、范围和服务质量需求。

b) 定义仪器的物联模型/剖面。根据 a)中的需求分析，定义仪器特定的物联模型/主剖面；面向不同共享需求，定义相应的特定物联剖面；验证各物联剖面的有效性、一致性，及与默认剖面的扩展一致性；分析确定网络访问的角色权限及其他服务质量的指标；定义访问接口的具体实例。

c) 建立映射器。根据仪器的主剖面和服务质量，部署仪器映射器；建立仪器映射器与实体仪器的数据交互，建立仪器映射器与网络的连接。

d) 创建虚拟仪器。确定所用的仪器软件服务；根据仪器物联剖面创建虚拟仪器；建立映射器与虚拟仪器的连接，实现实体仪器与虚拟仪器的数据绑定。

e) 根据数据共享目的，将虚拟仪器发布到注册中心。

可根据实体仪器共享的实际情况，对上述活动进行相应的剪裁。

6.2 访问过程

网络用户根据所需访问的实体仪器，发现其相应的虚拟仪器并进行数据交互，称为仪器访问过程。主要包括以下活动：

a) 获取实体仪器相应的虚拟仪器信息。可通过注册中心查找发现实体仪器相应的虚拟仪器。

b) 获取虚拟仪器的访问信息。可通过物联默认访问接口，获取当前虚拟仪器的特定访问接口。

c) 访问所需的实体仪器。通过当前虚拟仪器的访问接口，实现对实体仪器的操控和共享。

7 物联模型

7.1 基本约定

本标准采用 UML 对物联模型中的各类信息模型进行建模。每个特定的物联模型应符合以下约定：

a) 采用 UML 中的组合、聚合和继承关系来建模模型中类之间的关系；

b) 任何特定的物联模型须直接或间接扩展自同类型的物联默认模型，扩展机制符合 7.4 的规定；

c) 模型中的类不能存在组合关系的回路；

注：组合关系的回路。举例说明：某模型中的三个类：类 A、类 B、类 C，类 A 组合了类 B，类 B 组合了类 C，若类 C 组合了类 A，则在同一个模型中形成了组合关系的回路。

d) 一个特定的物联信息模型有且只有一个主类；

e) 模型中每个被聚合的类应包含 *id* 属性；

f) 模型中类的属性数据类型应为元数据模型已定义的实例；

g) 模型中的每个类均可包含 **Description** 类，关联基数为“0..1”，用于对该类提供文字型说明或其他约束。

为便于阅读，在物联默认模型的 UML 图中，类属性的数据类型采用附录 C 中元数据的英文名称来表示。

7.2 物联默认模型

7.2.1 仪器默认模型

仪器默认模型如图 2 所示，模型的主类为 **Analyzer**（仪器）类，由以下子类组成：

a) **Classification**（分类描述）类，用于描述实体仪器的分类和功能；

b) **Structure**（组成结构）类，用于描述实体仪器的组成结构、部件及其之间的关系；

c) **Analysis-Process**（分析过程）类，用于描述使用分析仪器进行测量分析的具体过程；

d) **Related-Object**（相关对象）类，用于描述分析仪器的相关对象；

e) **Running**（运行信息）类，用于描述虚拟仪器的运行相关信息；

f) **Access-Info**（访问信息）类，用于描述外部访问者应如何访问虚拟仪器。

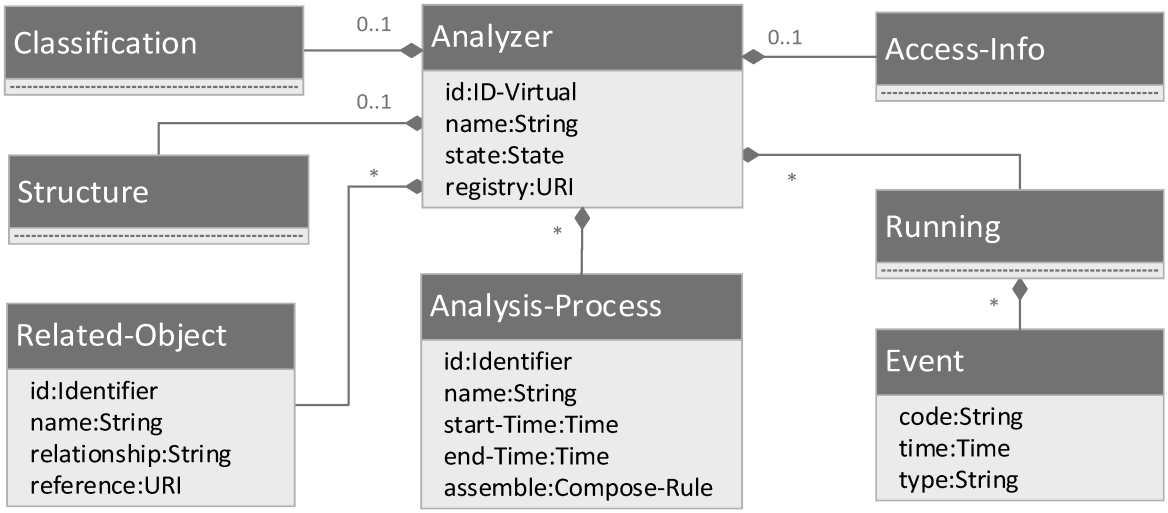

图 2 仪器默认模型的主要组成

7.2.1.1 Analyzer 类

Analyzer 类描述虚拟仪器的基本信息，有如下属性：

a) *id*：仪器 ID，属性值为当前虚拟仪器的唯一标识符；

b) *name*：仪器名称，属性值为当前虚拟仪器的名称；

c) *state*：仪器状态，属性值为当前虚拟仪器的实时状态；

d) *registry*：注册点，属性值为当前虚拟仪器的注册中心服务入口。

7.2.1.2 Classification 类

一个 **Classification** 类可包含多个 **Terms-Group**(术语组)类或 **Descriptor**(描述子)类。如图 3 所示。

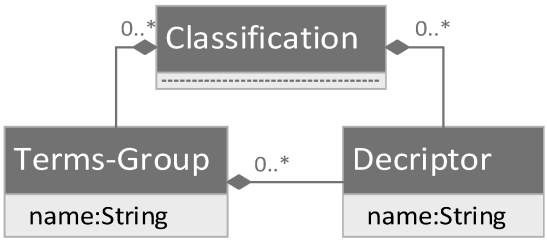

图 3 仪器默认模型 Classification 类的组成

Terms-Group 类描述某个分类维度的一组术语,其 *name* 属性表示当前分类维度的名称。一个 **Terms-Group** 类可包含多个 **Descriptor** 类。

Descriptor 类描述分析仪器相关的术语,其 *name* 属性值为特定的描述术语值。

7.2.1.3 Structure 类

一个 **Structure** 类可包含多个 **Component**(仪器部件)类,每个部件又可以分为若干个子部件。如图 4 所示。

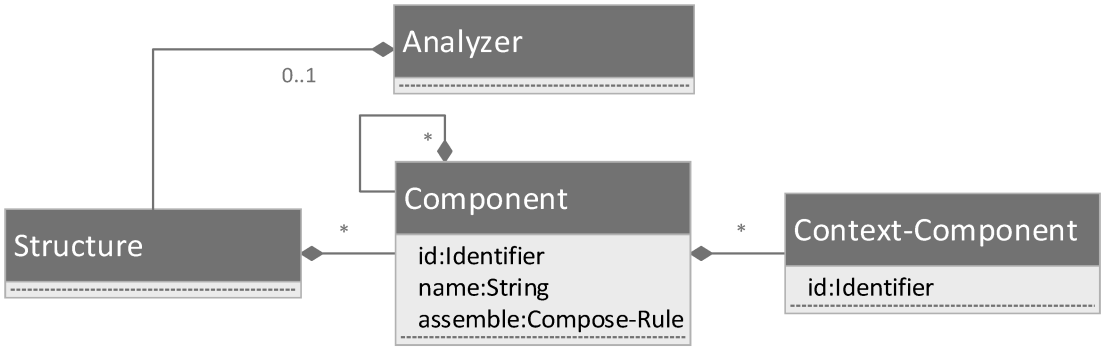

图 4 仪器默认模型 Structure 类的组成

Component 类定义了以下属性:

a) *id*:部件 ID,属性值为当前仪器部件的唯一标识符;

b) *name*:部件名称,属性值为当前部件的名称;

c) *assemble*:组合规则,属性值为一个组合表达式,用于描述当前部件的若干个子部件之间的关系,若当前部件没有子部件,则属性值为 null。组合规则表达式以对象标识符为基本元素,规约各子对象如何按规定的组合方式(顺序、选择、重复、同步等方式)组合为当前对象。

一个 **Component** 类可包含多个 **Context-Component**(部件周境)类,用于描述该仪器部件工作时的周境(如环境温湿度、供电要求等)。

7.2.1.4 Analysis-Process 类

一个 **Analyzer** 类可包含多个 **Analysis-Process** 类,表示一台实体仪器可以进行多次的测量分析,每次测量分析的过程信息由 **Analysis-Process** 类规约。

一个 **Analysis-Process** 类可包含：一个 **Object-Analyzed**（被分析对象）类，用于描述当前分析过程的被分析对象，即被检测物品；多个 **Analysis-Result**（分析结果）类，用于描述当前分析过程的分析结果；一个**Data-Realtime**（实时数据流）类，用于描述当前分析过程产生的实时数据；多个 **Context-Process**（分析周境）类，用于描述当前分析过程的周境（如温湿度、大气压等）；多个 **Measure-Activity**（测量活动）类，用于描述当前分析过程中的各个具体测量操作活动。如图 5 所示。

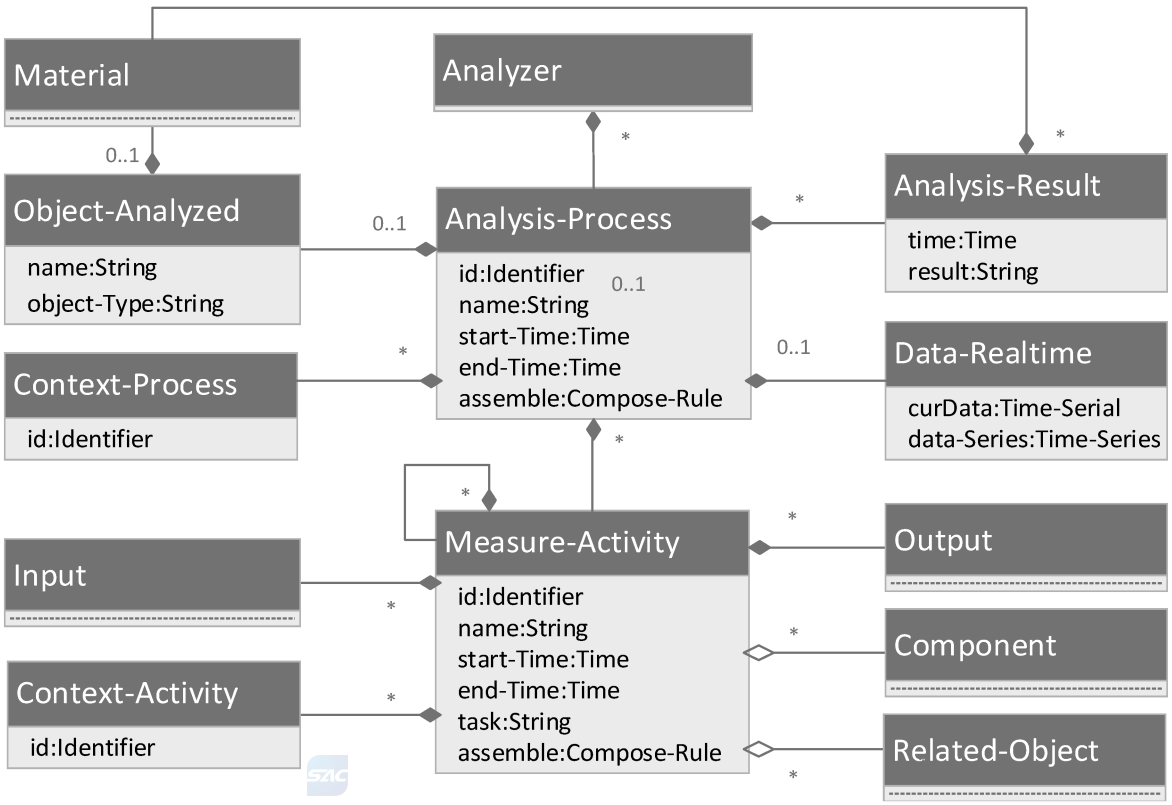

图 5　仪器默认模型 Analysis-Process 类的组成

Analysis-Process 类包含以下属性：

a）　*id*：分析过程 ID，属性值为当前分析过程的唯一标识符；

b）　*name*：分析过程的名称，属性值为当前分析过程的名称；

c）　*start-Time*：开始时间，属性值为当前分析过程的开始时间；

d）　*end-Time*：结束时间，属性值为当前分析过程的结束时间；

e）　*assemble*：组合规则，属性值为一个组合表达式，用于描述当前分析过程的若干个具体测量活动之间的关系。

Object-Analyzed 类包括以下属性：

a）　*name*：名称，属性值为被分析对象的名称；

b）　*object-Type*：分析品类型，属性值为被分析对象的类型。被分析对象有两种类型：一般物品和标准样品。

Analysis-Result 类包括以下属性：

a）　*time*：时间，属性值为当前分析结果的产生时间；

b）　*result*：分析结果，属性值为当前分析过程的分析结果。

Material(物品)类用于规约分析过程的输入或分析过程的分析结果(如:物质成分、化学结构或物理特性等数据)。**Object-Analyzed** 类可包含一个 **Material** 类,用于描述当前被分析的样品的相关数据;**Analysis-Result** 类可包含多个 **Material** 类,用于描述当前分析结果的数据。

Data-Realtime 类规约当前分析过程产生的实时数据。包括以下属性:

a) *curData*:当前数据,属性值为当前分析过程产生的最新数据;

b) *data-Series*:数据序列,属性值为当前分析过程自开始到当前时间产生的测量数据序列。

一个 **Measure-Activity** 类可包含:多个 **Input**(输入)类,用于表示当前测量活动的输入;多个 **Output**(输出)类,用于表示当前测量活动的输出;多个 **Context-Activity**(活动周境)类,用于描述当前测量活动的周境。一个测量活动中可与多个仪器部件(**Component** 类)或外部相关对象(**Related-Object** 类)发生关联。一个测量活动还可分解为若干个子测量活动。

Measure-Activity 类包含以下属性:

a) *id*:活动 ID,属性值为当前测量活动的唯一标识符;

b) *name*:活动名称,属性值为当前测量活动的名称;

c) *start-Time*:开始时间,属性值为当前测量活动的开始时间;

d) *end-Time*:结束时间,属性值为当前测量活动的结束时间;

e) *task*:任务,属性值描述当前测量活动的任务或目的;

f) *assemble*:组合规则,属性值为一个组合表达式,用于描述当前测量活动的若干个子测量活动之间的关系。

7.2.1.5 Related-Object 类

Related-Object 类包含以下属性:

a) *id*:相关对象的 ID,属性值为外部相关对象的唯一标识符;

b) *name*:对象名称,属性值为外部对象的名称;

c) *relationship*:关系,属性值描述当前分析仪器与外部对象的关系;

d) *reference*:参考,属性值为外部对象的资源标识符(URI),该资源一般作为外部对象相关的说明性资料。

7.2.1.6 Running 类

一个 **Running** 类可包含多个 **Event**(事件)类,用于描述当前实体仪器/虚拟仪器运行中发生的各类事件。

Event 类包含以下属性:

a) *code*:事件编码,属性值为发生的事件的具体编码值;

b) *time*:时间,属性值为事件发生的时间;

c) *type*:事件类型,属性值为发生的事件的类型,如:一般事件、报警事件、紧急处理事件等。

7.2.1.7 Access-Info 类

一个 **Access-Info** 类可包含:多个 **Self-Profile**(自身剖面)类,用于描述当前虚拟仪器可供外部访问的仪器剖面;多个 **Access-Interface**(访问接口)类,用于描述当前虚拟仪器对外提供的访问接口;多个 **Access-Qos**(数据访问的服务质量)类,用于描述与外部交互时的服务质量。如图 6 所示。

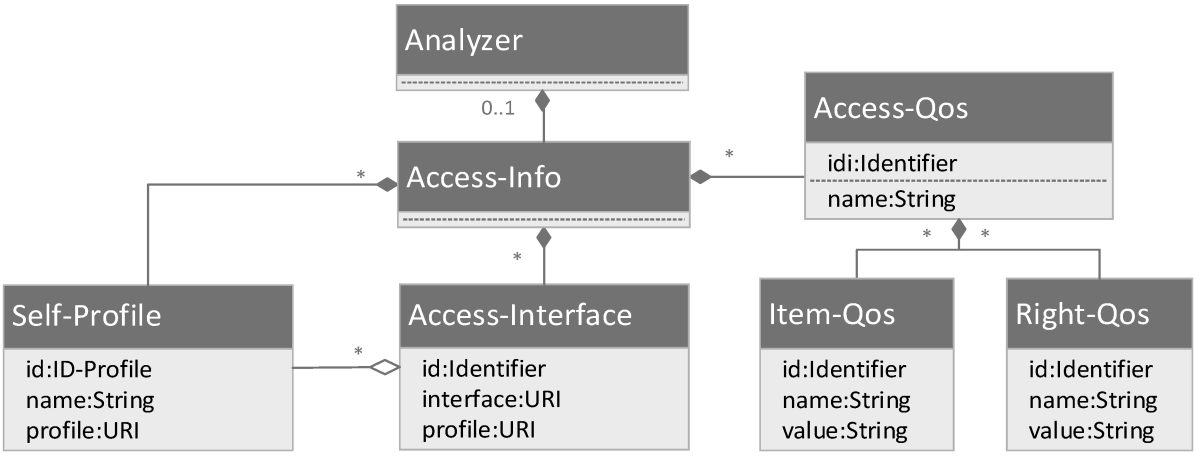

图 6　仪器默认模型 Access-Info 类的组成

Self-Profile 类包含以下属性：

a)　*id*：剖面 ID，属性值为仪器剖面的标识符；

b)　*name*：名称，属性值为当前仪器剖面的名称；

c)　*profile*：仪器剖面，属性值为当前仪器剖面的 URI。

Access-Interface 类包含以下属性：

a)　*id*：访问接口 ID，属性值为访问接口的标识符；

b)　*interface*：访问接口，属性值为访问接口的具体形式化表达式的 URI；

c)　*profile*：访问接口剖面，属性值为访问接口的剖面的 URI。

一个 **Access-Interface** 类可聚合多个 **Self-Profile** 类，表示一个访问接口可访问多个仪器剖面规约的数据集。

一个 **Access-Qos** 类包含：多个 **Item-Qos**（服务质量项）类，用于描述服务质量中的某个特定的服务质量项；多个 **Right-Qos**（服务质量权限）类，用于描述当前服务质量所要求的权限信息。

Access-Qos 类包含以下属性：

a)　*id*：服务质量 ID，属性值为当前服务质量的唯一标识符；

b)　*name*：名称，属性值为当前服务质量的名称。

Item-Qos 类包含以下属性：

a)　*id*：服务质量项 ID，属性值为服务质量项的唯一标识符；

b)　*name*：名称，属性值为服务质量项的名称；

c)　*value*：名称，属性值为服务质量项的具体取值。

Right-Qos 类包含以下属性：

a)　*id*：权限 ID，属性值为权限的唯一标识符；

b)　*name*：名称，属性值为权限的名称；

c)　*value*：名称，属性值为权限的具体取值。

7.2.2　注册中心默认模型

注册中心默认模型如图 7 所示，由四个子模型组成：注册中心入口模型，模型主类为 **Registry**（注册中心）类；仪器黄页库模型，模型主类为 **Lib-Yellow-Pages**（黄页库）类；物联剖面库模型，主类为 **Lib-Profile**（剖面库）类；元数据库模型，主类为 **Lib-Metadata**（元数据库）类。

Registry 类由 **Lib-Yellow-Pages** 类、**Lib-Profile** 类和 **Lib-Metadata** 类聚合而成，以上四个类均继承

自 **Services-Entry**(服务入口)类。

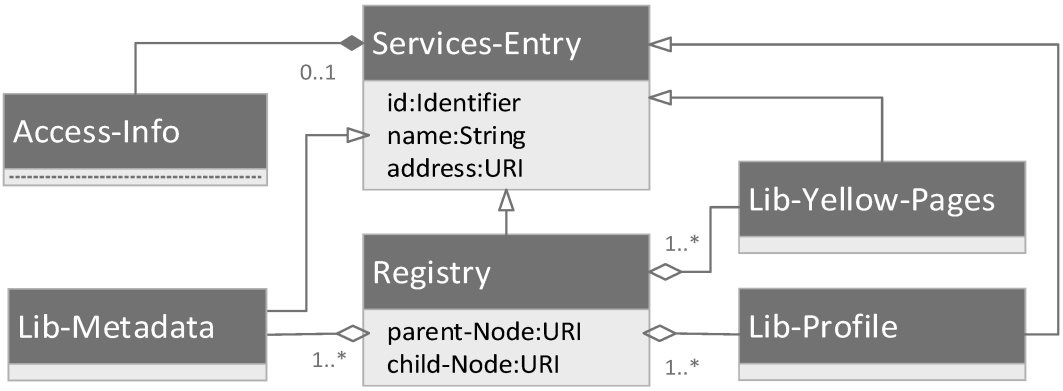

图 7　注册中心默认模型的主要组成

Services-Entry 类描述一个软件服务入口的基本信息,有以下属性:

a)　*id*:服务 ID,属性值为软件服务的唯一标识符;

b)　*name*:名称,属性值为软件服务的名称;

c)　*address*:服务地址,属性值为软件服务的访问地址。

Registry 类继承了 **Services-Entry** 类的所有元素。一个 **Services-Entry** 类可包含一个 **Access-Info** 类(见图 6),用于描述外部访问者应如何与当前软件服务进行数据交互。

Registry 类包含以下属性:

a)　*parent-Node*:父节点,属性值为上级注册中心节点的 URI;

b)　*child-Node*:子节点,属性值为下级注册中心节点的 URI。

7.2.2.1　仪器黄页库默认剖面

Lib-Yellow-Pages 类继承了 **Services-Entry** 类的所有元素。一个 **Lib-Yellow-Pages** 类可包含:多个 **Entity-Analyzer**(实体仪器)类,用于描述实体仪器的基本信息。**Entity-Analyzer** 类可包含多个 **Virtual-Analyzer**(虚拟仪器)类,用于描述当前实体仪器对应的虚拟仪器的基本信息。如图 8 所示。

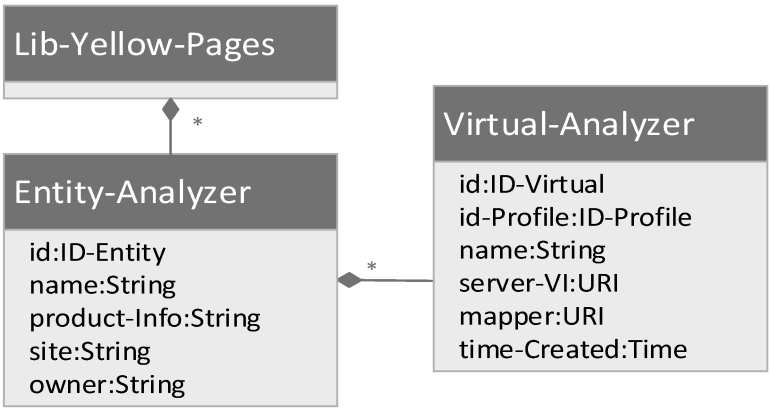

图 8　仪器黄页默认模型

Entity-Analyzer 类包含以下属性:

a)　*id*:实体仪器 ID,属性值为实体仪器的标识符;

b) *name*：名称，属性值为实体仪器的名称；
c) *product-Info*：产品信息，属性值为实体仪器的产品相关信息；
d) *site*：安装地点，属性值为实体仪器的安装地点；
e) *owner*：所有者，属性值为实体仪器的所有者或提供者信息。

Virtual-Analyzer 类包含以下属性：

a) *id*：虚拟仪器 ID，属性值为虚拟仪器的标识符；
b) *id-Profile*：物联剖面，属性值为虚拟仪器的物联剖面的标识符；
c) *name*：虚拟仪器名称，属性值为虚拟仪器的名称；
d) *server-VI*：运行服务器，属性值为虚拟仪器的软件服务 URI；
e) *mapper*：映射器，属性值为虚拟仪器的映射器的 URI；
f) *time-Created*：创建时间，属性值为虚拟仪器的创建时间。

7.2.2.2 物联剖面库默认剖面

Lib-Profile 类继承了 **Services-Entry** 类的所有元素。一个 **Lib-Profile** 类还可包含：多个 **Profile**（剖面）类，用于描述分析仪器物联剖面的基本信息。如图 9 所示。

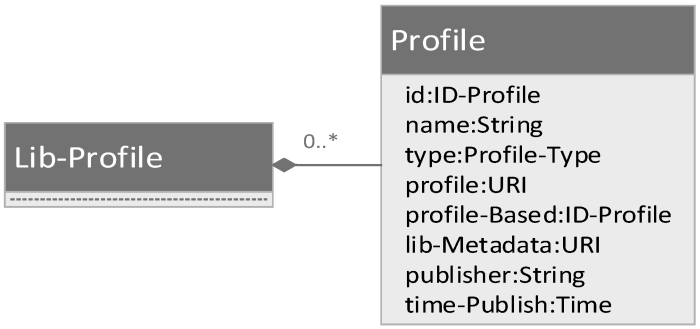

图 9 剖面库默认模型

Profile 类包含以下属性：

a) *id*：物联剖面 ID，属性值为特定物联剖面的标识符；
b) *name*：名称，属性值为当前物联剖面的名称；
c) *type*：剖面类型，属性值为当前物联剖面的具体类型；
d) *profile*：物联剖面，属性值为当前物联剖面的具体存放点的 URI；
e) *profile-Based*：父剖面，属性值为当前物联剖面的父剖面的标识符；
f) *lib-Metadata*：元数据库，属性值为当前物联剖面使用的元数据库的 URI；
g) *publisher*：发布者，属性值为当前物联剖面的发布者信息；
h) *time-Publish*：发布时间，属性值为当前物联剖面的发布时间。

7.2.2.3 元数据库默认剖面

Lib-Metadata 类继承了 **Services-Entry** 类的所有元素。一个 **Lib-Metadata** 类还可包含：多个 **Metadata**（元数据）类，**Metadata** 类具体阐述见 7.2.4。

7.2.3 访问接口默认模型

访问接口默认模型如图 10 所示，模型的主类为 **Interface**（访问接口）类，由以下子类组成：

a) **Request**（接口请求）类，用于描述访问接口交互中进行接口请求时的数据包；

b) **Response**(接口响应)类,用于描述访问接口交互中对接口请求的进行响应的数据包;

c) **Security**(接口安全)类,用于描述接口交互时数据包加密相关的信息;

d) **Frame-Header**(帧头)类,用于描述接口交互数据包的先导符;

e) **Frame-Tail**(帧尾)类,用于描述接口交互数据包的尾部添加符。

7.2.3.1 Interface 类

Interface 类描述一个访问接口的基本信息,包含以下属性:

a) *id-Interface*:接口 ID,属性值为访问接口的唯一标识符,用于对访问接口的信息管理;

b) *name*:接口名称,属性值为访问接口的名称;

c) *provider*:服务提供者,属性值为一个通用资源标识符,描述当前访问接口的网络访问入口点(包括网络地址、服务端口号等);

d) *timeout*:超时阈值,属性值为该接口请求的超时阈值。接口调用者根据该数值判断已发起的接口请求是否已超时。

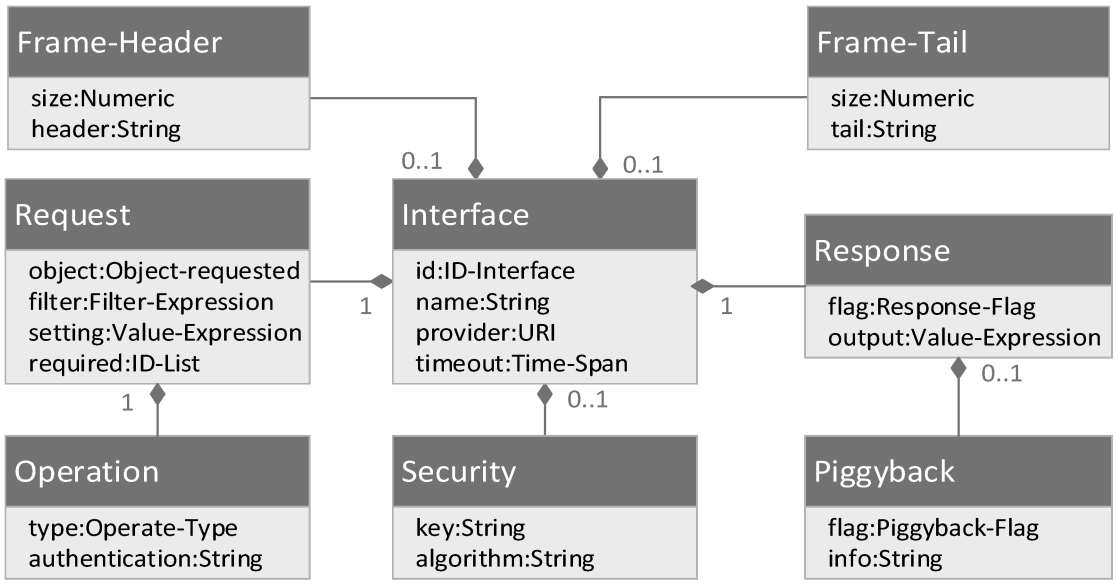

图 10　访问接口默认模型

Frame-Header 类包含以下属性:

a) *size*:大小,属性值为接口交互数据包的先导符的数据长度,单位为字节;

b) *header*:头部,属性值为接口交互数据包的先导符数据块。在接口数据交互时添加在接口请求/响应数据包的头部。

Frame-Tail 类包含以下属性:

a) *size*:大小,属性值为接口交互数据包尾部添加符的数据长度,单位为字节;

b) *tail*:头部,属性值为接口交互数据包尾部添加符数据块,在接口交互时添加在接口请求/响应数据包的尾部。

Security 类包含以下属性:

a) *key*:秘钥,属性值为一个秘钥字符串,用于接口交互中的数据加密/解密;

b) *algorithm*:加/解密算法,属性值为一个字符串,描述接口交互时数据加密/解密的算法。

7.2.3.2 Request 类

Request 类包含以下属性：

a) *object*：访问对象，属性值表示接口访问的物联标准件的相关信息，取值主要如下：
——“P”为剖面(Profile)，表示接口访问的是该物联标准件的剖面；
——“I”为实例集(Instances)，表示接口访问的是该物联标准件所管理的若干实例；
——某特定对象的唯一标识符，表示接口访问的是该物联标准件管理的特定对象。

根据 *object* 的属性取值，可确定访问对象及其剖面，该访问对象的剖面规定了访问接口模型中DSL 表达式类型的各个属性(**Request** 类的 *filters*、*setting*、*required* 属性，以及 **Response** 类 *output* 属性)的基本元素。

b) *filters*：过滤条件，属性值为一个 DSL 表达式，用于在访问对象中过滤出符合条件的具体对象。

c) *setting*：设置，属性值为一个 DSL 表达式。在接口的操控类型(**Operation** 类的 *type* 属性值)为“C”(新增)或“U”(修改)时，该属性值规约待新增或修改的各数据项及相应的数值；接口的操控类型为“R”“D”“A”时，该属性值表达式为 null。

d) *required*：请求项，属性值为一个以键值对中的“键”为基本元素的 DSL 表达式，描述当前接口响应需返回的具体数值项。当不需要返回具体数值时，该属性值表达式为 null。

一个 **Request** 类应包含一个 **Operation**(信息操控)类，用于描述本次接口请求的信息操控方式。**Operation** 类包含以下属性：

a) *type*：操控类型，属性值为当前接口请求的操控类型，本标准定义了 5 种操控类型：
——“C”为增加(Create)，增加新的信息；
——“R”为读取(Read)，读取相关信息；
——“U”为修改(Update)，当接口调用者为仪器映射器时，表示对仪器数据的修改；当调用者为一般网络用户时，表示对仪器的控制或设置操作；
——“D”为删除(Delete)，删除相关信息；
——“A”为验证(Authentication)，用以表示权限验证的操作。

b) *authentication*：验证码，属性值为一个字符串，用于对接口访问的数据的操控权限进行验证。

7.2.3.3 Response 类

Response 类包含以下属性：

a) *flag*：响应标志，属性值为当前接口调用执行结果的描述；

b) *output*：输出，属性值为一个 DSL 数值表达式，描述当前接口响应的具体输出结果。

Response 类可包含一个 **Piggyback**(信息捎带)类，用于在接口响应返回信息中捎带紧急信息(如紧急通知、异常事件等)。**Piggyback** 类有以下属性：

a) *flag*：捎带标志，属性值为捎带数据的简单描述；

b) *info*：捎带信息，属性值是一个字符串，描述当前接口响应所捎带的具体信息。

7.2.4 元数据默认模型

元数据默认模型如图 11 所示，模型主类为 **Metadata** 类，由以下子类组成：

a) **Value-Constraints**(数值约束)类，用于描述元数据的数值约束；

b) **Generate-Constraints**(生成约束)类，用于描述多个子元数据生成一个复合元数据时的约束；

c) **Aggregate-Constraints**(聚合约束)类，用于描述复合元数据的整体性约束。

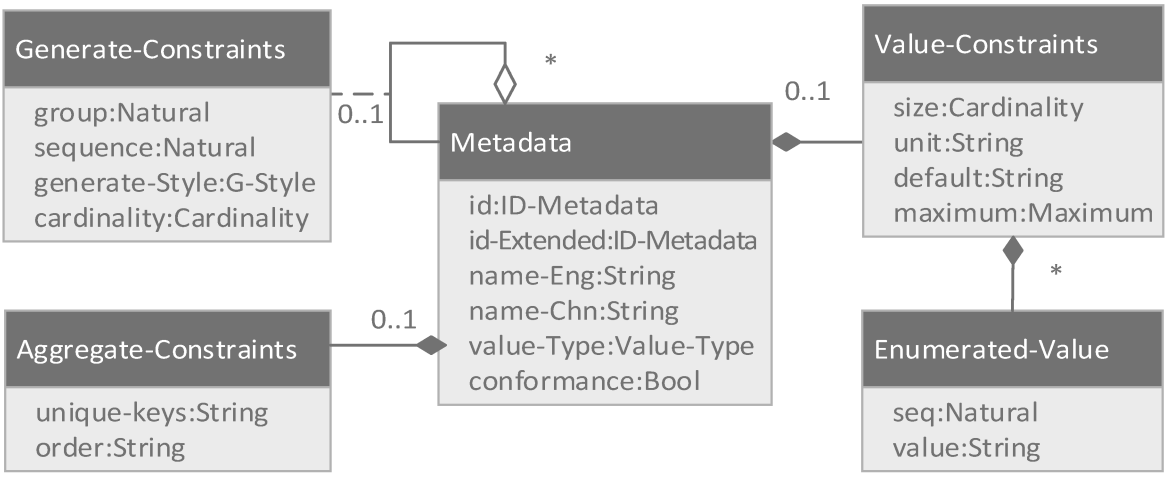

图 11　仪器元数据默认模型

元数据是元数据模型的实例。附录 C 中给出了本标准中所用的元数据的定义。

数值是元数据的实例。元数据的数值结构相当于编程语言中的数据结构，规定了数值的数据组成结构和类型。

7.2.4.1　Metadata 类

Metadata 类描述一个元数据的基本信息，包含以下属性：

a) *id*：元数据 ID，属性值为该元数据的唯一标识符。

b) *id-Extended*：被扩展的元数据 ID，属性值为被扩展的元数据的标识符。

c) *name-Eng*：英文名称，属性值为元数据的英文名称。

d) *name-Chn*：元数据中文名，属性值为元数据的中文名称。

e) *value-Type*：数值类型，属性值为元数据的数值类型。本标准定义了以下基本数值类型："S"表示字符型，"N"表示为数字型，"T"表示时间型，"B"表示布尔型，"U"表示资源型，"E"表示枚举型，"G"表示复合型。

f) *conformance*：符合性，属性值为 true 或 false，true 表示该元数据已得到本标准有关管理机构的认可，false 表示未得到认可。

7.2.4.2　Value-Constraints 类

一个 **Metadata** 类最多包含一个 **Value-Constraints** 类，用于规约基础元数据的数值约束。**Value-Constraints** 类包括以下属性：

a) *size*：长度，该属性值描述元数据的数值的长度，属性值中的下限表示字节长度的最小值，上限表示字节长度的最大值。若为 null 表示长度不限。

b) *unit*：单位，该属性值描述当前元数据的数值的计量单位。若元数据类型非数字型，则值为 null。

c) *default*：缺省值，该属性值描述当前元数据的数值的缺省值。若 *maximum* 属性值不为空，则缺省值须在 *maximum* 属性值的范围内。

b) *maximum*：最值约束，该属性值描述当前元数据的数值的最小值/最大值。若为 null 或"*"，表示无最值限制。

当 **Metadata** 类的实例为枚举型的元数据时，**Value-Constraints** 类应至少包含一个 **Enumerated-**

Value（枚举值）类，每个 **Enumerated-Value** 类的实例描述了枚举型元数据的一个枚举值。**Enumerated-Value** 类包括以下属性：

a） *seq*：枚举序号，该属性值描述枚举型的数值编号；

b） *value*：枚举值，该属性值为与枚举序号对应的具体数值。

7.2.4.3 Generate-Constrains 类

一个 **Metadata** 类嵌套多个 **Metadata** 子类时，每个被嵌套的子类应包含一个 **Generate-Constraints** 类。**Generate-Constraints** 类包括以下属性：

a） *group*：分组号，该属性值为子元数据的分组号。分组号相同的子元数据为一组，组内的子元数据按 *generate-Styles* 属性值规定的生成方式生成一个新元数据。

b） *sequence*：序号，该属性值为子元数据在分组内的序号，表示当前子元数据在分组内的前后关系。

c） *generate-Style*：组合方式，该属性值描述子元数据组装为一个新元数据时的组合方式。本标准定义了以下组合方式：

——“Sequential”顺序组合方式，表示将当前分组中 *generate-Style* 属性值为“Sequential”的所有子元数据组成一个有序向量的复合元数据，新元数据的数值结构是该组内所有子元数据的数值结构序列，顺序按该组内 *sequence* 属性值的升序排列；

——“Selective”选择组合方式，表示当前分组中 *generate-Style* 属性值为“Selective”的子元数据任选其一作为新元数据中的一个成员。新元数据的数值空间是所有该组中的子元数据的数值空间的并集；

——“Intersective”交集组合方式，表示当前分组中 *generate-Style* 属性值为“Intersective”的所有子元数据形成一个新元数据，新元数据的数值空间是所有该组中的子元数据的数值空间的交集；

——“Concatenative”串接组合方式，表示当前分组中 *generate-Style* 属性值为“Concatenative”的子元数据生成一个新元数据，新元数据的数值是该组内所有子元数据的数值的串接，顺序按其 *sequence* 属性值的升序排列。

d） *cardinality*：基数，该属性值为子元数据在生成新元数据时的基数关系，表示当前子元数据在新元数据中自身的复制次数。

7.2.4.4 Aggregate-Constraints 类

Aggregate-Constraints 类包含以下属性：

a） *unique-Key*：唯一键，该属性值描述元数据的实例的数值唯一键；

b） *order*：排序，该属性值描述元数据的实例的排序信息。

7.3 物联剖面

本标准采用 DSL 语言将物联模型形式化表达为相应的物联剖面。

将特定物联模型形式化表达为相应主剖面的方法如下：

a） 将物联模型中的每个类表示为一个键值对表达式，类名作为键值对中的“键”，“值”为由类中各属性的键值对组成的表达式；将类的每个属性表示为一个键值对，属性名作为键值对中的“键”，“值”为该属性数据类型对应的元数据标识符；

注：物联模型的 UML 图中类属性的数据类型，在剖面中以该元数据的标识符来代替。如：仪器默认模型 **Analyzer** 类中的 *name* 属性数据类型为 String，在剖面中的相应键值对应为″name″:″M00000″。

b） 按照类之间的关系，将物联模型的各个类形成的键值对表达式进行组装；

1） 组合关系的处理。对被组合的类的键值对表达式添加相应基数，一并添加到组合类的键值对的“值”表达式中，形成以组合类的类名为“键”的键值对表达式。

2） 聚合关系的处理。将被聚合的类的标识符属性作为一个键值对：该键值对的“键”采用“被聚合类的类名″＋″.″＋″被聚合的类的标识符属性名”的形式，“值”为被聚合的类的标识符属性的数据类型相应的元数据标识符。将以上键值对添加相应的基数，一并添加到聚合类的键值对的“值”表达式中。如：仪器默认模型中的 **Component** 类被聚合到 **Measure-Activity** 类，应在 **Measure-Activity** 类相应的键值对的“值”表达式中添加″Component.id″:[″M00100″](*)的表达式。

c） 按照 a)、b)逐级组装，直至形成以模型主类的类名为“键”的键值对表达式。

附录 B 给出了物联默认剖面的具体表达。

7.4 扩展机制

7.4.1 基本约定

本标准规定物联模型的扩展遵循与 UML 相一致的扩展机制，扩展是在已有物联模型的基础上增加新的约束限制和信息，从而得到一个新的物联模型。扩展关系的含义是：如果 B 是 A 的扩展，则对 A 及其实例的所有合法操作都可以适用于 B 及其实例，反之则不然。扩展关系具有以下性质：

a） 扩展关系具有自反性，即：每个模型对象都是自身的扩展；

b） 扩展关系具有传递性，即：若 C 为 B 的扩展，B 为 A 的扩展，则 C 也是 A 的扩展。

基于物联默认模型和扩展机制，可以逐级建立各类分析仪器物联模型，直至满足特定实体仪器物联的信息建模需要。以仪器模型为例，分析抽取色谱仪器的共性特征，基于默认模型扩展形成色谱仪器的信息模型；分析抽取气相色谱仪器的共性特征，基于色谱仪器的信息模型扩展形成气相色谱仪器的信息模型；基于气相色谱仪器的信息模型，仪器提供者可根据自己的需要，扩展形成某特定气相色谱仪器产品的信息模型。如图 12 所示。

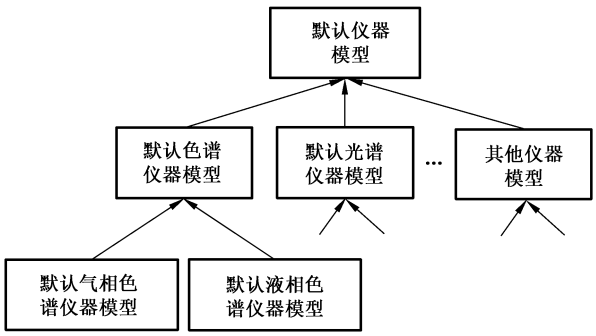

图 12 仪器模型扩展体系示意图

一个特定的物联模型必须直接或间接扩展自与其类型相同的物联默认模型。

7.4.2 信息模型的扩展机制

信息模型的扩展必须遵循以下原则：

a） 扩展应是同类型模型的扩展。如，仪器模型应扩展自仪器默认模型或其他已有的仪器模型。

b） 扩展应引入一些更为严格的语义和约束，但不能减少被扩展的模型中所规定的语义及约束。

信息模型扩展方法主要如下：

a） 在已有信息模型中增加新的类，但已有的类不能减少。扩展得到的新模型中类与类之间不能

形成组合关系的回路；新增的类不能成为新模型的主类；

b) 改变已有信息模型中类之间的关联关系：

1) 改变关联基数。新模型中类的关联基数应在已有模型相应类的关联基数范围内。

2) 增加关联关系。新模型中类与类之间不能形成组合关系的回路。

c) 在已有类中增加类的属性，但类的已有属性及其相关的约束不能减少。

d) 改变类属性的元数据。改变后的新元数据必须是改变之前的元数据的扩展。

7.4.3 主剖面的扩展机制

主剖面的扩展方法主要如下：

a) 在主剖面的某个键值对的“值”部分添加新的键值对表达式。

b) 改变主剖面中某个键值对的基数。改变后的基数须在原有的基数范围内。

c) 改变主剖面中某个键值对的“值”。改变后的“值”必须为已定义的元数据标识符，且改变后的新元数据必须是改变之前的元数据的扩展。

7.4.4 元数据扩展关系的判定

元数据扩展在其数值关系方面的含义是：若元数据 B 为 A 的扩展，则可以按照两者数值结构之间的映射关系，从 B 的任意实例数据中抽取数据形成 A 的实例数据。

元数据扩展关系的判定准则：

a) 所有元数据均为字符型元数据 String 的扩展；

b) A 和 B 为基础元数据，若 B 为 A 的扩展，则 A 和 B 须为同一类型（value-Type 属性值相同），且其数值约束（**Value-Constraints** 类的实例）满足以下条件：

1) B 的 *size* 属性值在 A 的 *size* 属性值范围内；

2) B 的 *maximum* 属性值在 A 的 *maximum* 属性值范围内；

3) B 的 *unit* 属性值与 A 的 *unit* 属性值相同；

4) 若 A 和 B 为枚举类型，B 的枚举值集合为 A 的枚举值集合的子集。

c) 通过元数据的 *id-Extended* 属性值和扩展关系的传递性来判定；

d) 根据元数据的数值结构表达式来判定。若 A 和 B 分别为两个元数据，其数值结构的表达式分别为 Adsl 和 Bdsl，对 Adsl 中任意一个键值对 kvpA，在 Bdsl 中有且只有一个键值对 kvpB，**key-DSL**（kvpB）＝**key-DSL**（kvpA），kvpB 的基数在 kvpA 的基数范围内，且 **Value-DSL**（kvpB）是 **Value-DSL**（kvpA）的扩展，则 B 为 A 的扩展。相关形式化定义见附录 A。

8 数据交换

8.1 基本约定

各物联标准件按其物联剖面的规定与外部进行数据交换。每个物联标准件的物联剖面应包括：

a) 访问接口剖面；

b) 描述该标准件自身信息的剖面。若该标准件为虚拟仪器，则该部分为仪器剖面；若该标准件为注册中心，则该部分为注册中心剖面；

c) 元数据剖面。

物联标准件对外数据交换应遵循以下约定：

a) 应支持按其相应的物联默认剖面的规定与外部进行数据交换；

b) 数据交换涉及的元数据应符合该标准件的元数据剖面的规定；

c) 通过默认访问接口，应能获取物联标准件的以下信息：

1） 该标准件特定的物联剖面；
2） 该标准件对外的访问接口；
3） 该标准件对外数据交换所用的元数据。

8.2 数据交换过程

外部与物联标准件的数据交换过程主要如下：

a） 建立网络连接。访问者与该标准件建立网络连接，连接端口由其访问接口信息模型中 **Interface** 类的 *provider* 属性值来确定。如连接成功，进行 b)；如连接不成功，则可再尝试建立连接，或反馈连接失败。
b） 通过默认访问接口，获取该标准件特定的物联剖面、访问接口、元数据等信息。
c） 采用该标准件的访问接口进行数据交互。
d） 数据交换完成，关闭网络连接。

8.3 访问接口的调用

8.3.1 接口交互数据帧

访问接口模型/剖面规定了接口交互的数据帧结构。默认访问接口的数据帧结构如图 13 所示，图中的虚线部分为可选部分，实线部分为必须存在的部分。

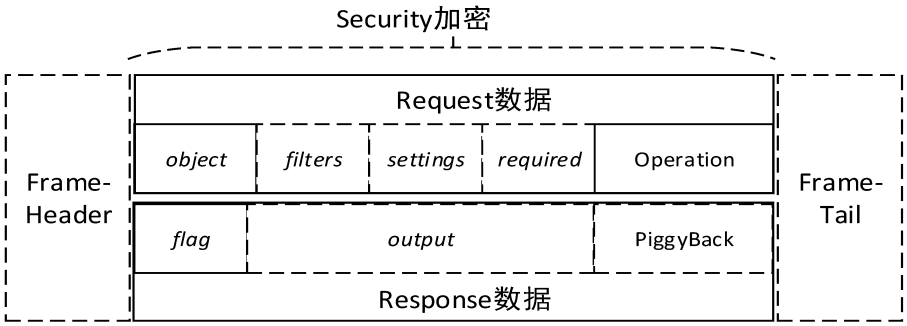

图 13 接口数据交互的帧结构

接口请求/响应数据包的生成：数据包为符合 DSL 语言的表达式，表达式中"{}"内的键值对分别按访问接口模型中 **Request** 类和 **Response** 类(参见 7.2.3 的描述)在访问接口剖面中相应的键值表达式的顺序生成；若 **Security** 类的实例数值不为空，则按其要求对数据包进行加密。

接口交互的数据帧的生成：将访问接口模型中 **Frame-Header** 类的实例数值作为先导符，添加在接口请求/响应的数据包的前面；将 **Frame-Tail** 类的实例数值作为结束符，添加在该数据包的后面。

访问接口涉及对超长数据(如多媒体、二进制文件等)的访问时，在接口请求/接口响应的数据包中只交换该数据的 URI 值，其二进制流的具体传输本标准不作限定。

8.3.2 接口的调用

访问接口的主要调用过程如下：

a） 调用者生成接口请求数据帧，并发送给物联标准件。
b） 物联标准件接收接口请求数据帧，验证、解析正确后，生成并发送接口响应数据帧给调用者。如验证、解析错误，则将该数据帧丢弃。

c) 如调用者在发出接口请求数据帧后的特定时间(访问接口模型中 **Interface** 类的 *timeout* 属性值)内,正确接收到接口响应数据帧,则当前访问正常;若超时,则本次访问失败。

8.3.3 接口交互数据包的数值结构

8.3.3.1 接口请求数据包的数值结构

访问接口默认剖面规定的接口请求数据的数值结构如下:

```
"Request":{
        "object":"M00150",
        "filters":"M00154",
        "settings":"M00155",
        "required":"M00101",
        "Operation":{
                "type":"M00156",
                "authentication":"M00000"
                }
        };
```

8.3.3.2 接口响应数据包的数值结构

访问接口默认模型规定的接口响应数据的数值结构如下:

```
"Response":{
        "flag":"M00157",
        "output":"M00155"
        "Piggyback":{
                "flag":"M00158",
                "info":"M00000"
                }
        };
```

附 录 A
（规范性附录）
数据规约语言

A.1 概述

本附录定义数据规约语言(Data Specification Language,简称 DSL)的符号、语法和关键字。

DSL 的形式化表达采用字符串形式,字符集采用 GB 13000—2010。

A.2 基本符号

letter_uppercase ="A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"|"N"|"O"|"P"|"Q"|
"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z";

letter_lowercase ="a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|"l"|"m"|"n"|"o"|"p"|"q"|
"r"|"s"|"t"|"u"|"v"|"w"|"x"|"y"|"z";

letter =letter_uppercase|letter_lowercase; //英文字母

digit ="0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"; //0-9 的数字

letter-digit-string =[*letter*| *digit*](*);.....//由英文字母或 0-9 的数字形成的字符串

identifier =*letter-digit-string*,[{"_"|"."|"-"},*letter-digit-string*](*);//标识符

cardinality-string ="*"|[*digit*](*)|{[*digit*](*),"..",{"*"| [*digit*](*)}};//基数

A.3 DSL 语法

DSL 形式化语法中的特殊符号见表 A.1。

表 A.1 形式化语法中的特殊符号

符号	符号名称	符号语法说明
=	等号(equal sign)	定义一个符号
|	单杠号(vertical line)	符号多选一
{}	花括号(curly bracket)	符号的起/止界定符
[]	直括号(square bracket)	符号的重复(或符号的数组)界定符
()	括号(bracket)	置于直括号[]后面,括号内的值为基数(cardinality),表示直括号内符号的可重复次数
,	逗号(comma)	符号的串接(或键值对的分隔符)
:	冒号(colon)	键值对中键与值两部分的连接符
;	分号(semicolon)	符号的结束
"	双引号(double quotes)	文本块的界定
注:若形式化表达式中直括号后没有给出基数值,则表示基数为 1。		

本形式化语法以键值对为基本元素，一个键值对可视为一个对象，键值对中的“键”是对象的唯一标识符；键值对中的“值”表示了该对象的值，一个键值对的“值”既可以是一个单一的字符串，也可以是一个由若干个键值对按 DSL 语法组成的表达式；键值对中的“键”和“值”之间由符号":"连接；每个键值对可以重复若干次，重复次数由其基数决定。一个 DSL 表达式可以用符号"{}"括起来，"{}"内的键值对由符号“,”分隔。

DSL 表达式的语法定义如下：

dsl-formula ＝*key-value-pair*｜ *value-formula*；

key-value-pair ＝*key-string*,":", *value-formula*；

key-string ＝ """, *identifier*, """；

value-formula ＝"{",*dsl-object*,[",",*dsl-object*](＊), "}"；

dsl-object ＝*key-value-pair*｜*value-string*｜*array-KVP*；

array-KVP ＝ "[",*key-value-pair* ,"](",*cardinality-string* ,")"；

value-string ＝{""",*string-text* ,"""}｜
digital-string｜
"*this*"｜
"*true*"｜
"*false*"｜
"*null*"｜
"*empty*"；

上述语法定义中，*string-text* 是一个字符串，与 IETF RFC 8259 中 string 的定义相同；*digital-string* 是由可计算的数学数值的字符串形式，与 IETF RFC 8259 中 number 的定义相同。

true(表示布尔值真)，false(表示布尔值假)，null(表示空值)均为表达式中的保留关键字。

this 也是 DSL 中的保留关键字，用于表示键值对的递归关系。this 只出现在键值对的值部分，若一个 DSL 表达式中出现了值为 this 的键值对，则说明该键值对被包含在一个键与其相同的更大的键值对中。例如，定义一个家族谱系的元数据：

{"House-Master":{ "Name": "String","Spouse": "String",
"Childs":[{"House-Master":this}](＊)
}}

empty 表示一个空的表达式。一个“值”为 empty 的键值对用于表达模型中的没有任何属性的类。

DSL 表达式中将“{}”和“[]”括起来的部分视为若干个键值对的一个序列，而不是键值对的集合。这样有利于表达的简洁性，可省略表达式中每“{}”符号对内的尾部连续的 null 值，而不影响数值的解析；在数据交互时，在上下文明确的情况下，键值对中的“键”部分也可省略，只保留“值”部分。

DSL 的语法定义中，“,”作为符号的串接；在 DSL 表达式中的“,”作为键值对的分割符。

采用 DSL 表达一个 UML 模型时，键值对中的“键”为模型中类或属性的标识符，“值”为模型中属性的数据类型或元数据标识符；表达一个具体数值时，键值对中的“键”为数据项的标识符，“值”为该数据项的具体数值。

为便于表达，本标准定义两个函数：

a) 函数 *key-DSL*(A)表示键值对 A 的“键”；

b) 函数 *value-DSL*(A)表示键值对 A 的“值”的表达式。

若 A 为一个键值对，定义以下 DSL 语法表达上的等价关系：

a) *key-DSL*(A):*value-DSL*(A)＝{*key-DSL*(A):*value-DSL*(A)}＝*key-DSL*(A):{*value-DSL*(A)}；

b) *key-DSL*(A):[*value-DSL*(A)](＊)＝[*key-DSL*(A):*value-DSL*(A)](＊)。

附 录 B
（规范性附录）
物联默认剖面

B.1 概述

本附录给出了分析仪器物联信息模型中的各默认模型的 DSL 形式化表达。本附录中使用到的元数据的定义见附录 C。

B.2 仪器默认剖面

```
"Analyzer":{                                          //模型主类
    "id":"M00105",                                    //"ID-Virtual"
    "name":"M00000",                                  //"String"
    "state":"M00121",                                 //"State"
    "registry":"M00005",                              //"URI"
    "Running":[{
        "Event":[{
            "code":"M00000",                          //"String"
            "time":"M00002",                          //"Time"
            "type":"M00000",                          //"String"
            "Description":[{empty}](0..1)
            }](*),
        "Description":[{empty}](0..1)
    }](*),
    "Access-Info":[{
        "Self-Profile":[{
            "id":"M00103",                            //"ID-Profile"
            "name":"M00000",                          //"String"
            "profile":"M00005",                       //"URI"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Interface":[{
            "id":"M00100",                            //"Identifier"
            "interface":"M00005",                     //"URI"
            "profile":"M00005",                       //"URI"
            "Self-Profile.id":["M00103"](*),          //"ID-Profile"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Qos":[{
```

```
            "id":"M00100",                          //"Identifier"
            "name":"M00000",                        //"String"
            "Item-Qos":[{
                "id":"M00100",                      //"Identifier"
                "name":"M00000",                    //"String"
                "value":"M00000",                   //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Right-Qos":[{
                "id":"M00100",                      //"Identifier"
                "name":"M00000",                    //"String"
                "value":"M00000",                   //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Description":[{empty}](0..1)
        }](*),
        "Description":[{empty}](0..1)
    }](0..1),
    "Classification": [{
        "Terms-Group": [{
            "name": " M00000",                      //"String"
            "Descriptor": [{
                "name": "M00000",                   //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Description":[{empty}](0..1)
        }](*),
        "Descriptor": [{
            "name": "M00000",                       //"String"
            "Description":[{empty}](0..1)
        }](*),
        "Description":[{empty}](0..1)
    }](0..1),
    "Structure": [{
        "Component": [{
            "id": "M00100",                         //"Identifier"
            "name": "M00000",                       //"String"
            "assemble": "M00127",                   //"Compose-Rule
            "Component": [this](*),
            "Context-Component": [{
                "id": "M00100",                     //"Identifier"
```

```
            "Description":[{empty}](0..1)
        }](*),
        "Description":[{empty}](0..1)
    }](*)
}](0..1),
"Related-Object": [{
    "id": "M00100",                                  //"Identifier"
    "name": "M00000",                                //"String"
    "relationship": "M00000",                        //"String"
    "reference": "M00005",                           //"URI"
    "Description":[{empty}](0..1)
}](0..*),
"Analysis-Process": [{
    "id": "M00100",                                  //"Identifier"
    "name": "M00000",                                //"String"
    "start-Time": "M00002",                          //"Time"
    "end-Time": "M00002",                            //"Time"
    "assemble": "M00127",                            //"Compose-Rule"
    "Data-Realtime": [{
        "curData": "M00122",                         //"Time-Serial"
        "data-Series": "M00123"                      //"Time-Series"
        "Description":[{empty}](0..1)
    }](0..1),
    "Context-Process": {
        "id": "M00100",                              //"Identifier"
        "Description":[{empty}](0..1)
    }](*),
    "Object-Analyzed": [{
        "name": "M00000",                            //"String"
        "object-Type": "M00000",                     //"String"
        "Material": [null](0..1),
        "Description":[{empty}](0..1)
    }](0..1),
    "Analysis-Result": [{
        "time": "M00002",                            //"Time"
        "result": "M00000",                          //"String"
        "Material": [null](*),
        "Description":[{empty}](0..1)
    }](*),
    "Measure-Activity": [{
        "id": "M00100",                              //"Identifier"
```

```
            "name": "M00000",                                   //"String"
            "start-Time": "M00002",                             //"Time"
            "end-Time": "M00002",                               //"Time"
            "task": "M00002",                                   //"String"
            "assemble": "M00127",                               //"Compose-Rule"
            "Measure-Activity": [this](*),
            "Input": [null](*),
            "Output": [null](*),
            "Context-Activity": [{
                "id": "M00100",                                 //"Identifier"
                "Description":[{empty}](0..1)
            }](0..1),
            "Component.id": [{"M00100"}](0..*),                 //"Identifier"
            "Related-Object.id": [{"M00100"}](0..*),            //"Identifier"
            "Description":[{empty}](0..1)
        }](*),
        "Description":[{empty}](0..1)
    }](0..*)}
    "Description":[{empty}](0..1)
}
```

B.3 注册中心默认剖面

B.3.1 注册中心入口默认剖面

```
"Registry": {                                                   //模型主类
    "id": "M00100",                                             //"Identifier"
    "name": "M00000",                                           //"String"
    "address": "M00005",                                        //"URI"
    "parent-Node": "M00005",                                    //"URI"
    "child-Node": "M00005",                                     //"URI"
    "Access-Info":[{
        "Self-Profile":[{
            "id":"M00103",                                      //"ID-Profile"
            "name":"M00000",                                    //"String"
            "profile":"M00005",                                 //"URI"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Interface":[{
            "id":"M00100",                                      //"Identifier"
            "interface":"M00005",                               //"URI"
            "profile":"M00005",                                 //"URI"
```

```
            "Self-Profile.id":["M00103"](*),                    //"ID-Profile"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Qos":[{
            "id":"M00100",                                      //"Identifier"
            "name":"M00000",                                    //"String"
            "Item-Qos":[{
                "id":"M00100",                                  //"Identifier"
                "name":"M00000",                                //"String"
                "value":"M00000",                               //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Right-Qos":[{
                "id":"M00100",                                  //"Identifier"
                "name":"M00000",                                //"String"
                "value":"M00000",                               //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Description":[{empty}](0..1)
        }](*),
        "Description":[{empty}](0..1)
    }](0..1),
    "Lib-Metadata.id": [{"M00100"}](1..*),                      //"Identifier"
    "Lib-Yellow-Pages.id": [{"M00100"}](1..*),                  //"Identifier"
    "Lib-Profile.id": [{"M00100"}](1..*)                        //"Identifier"
    "Description":[{empty}](0..1)
}
```

B.3.2 仪器黄页库默认剖面

```
"Lib-Yellow-Pages": {                                           //模型主类
    "id": "M00100",                                             //"Identifier "
    "name": "M00000",                                           //"String"
    "address": "M00005",                                        //"URI"
    "Access-Info"":[{
        "Self-Profile":[{
            "id":"M00103",                                      //"ID-Profile"
            "name":"M00000",                                    //"String"
            "profile":"M00005",                                 //"URI"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Interface":[{
```

```
            "id":"M00100",                          //"Identifier"
            "interface":"M00005",                   //"URI"
            "profile":"M00005",                     //"URI"
            "Self-Profile.id":["M00103"](*),        //"ID-Profile"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Qos":[{
            "id":"M00100",                          //"Identifier"
            "name":"M00000",                        //"String"
            "Item-Qos":[{
                "id":"M00100",                      //"Identifier"
                "name":"M00000",                    //"String"
                "value":"M00000",                   //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Right-Qos":[{
                "id":"M00100",                      //"Identifier"
                "name":"M00000",                    //"String"
                "value":"M00000",                   //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Description":[{empty}](0..1)
        }](*),
    "Description":[{empty}](0..1)
    }](0..1),
    "Entity-Analyzer": [{
        "id": "M00106",                             //"ID-Entity"
        "name": "M00000",                           //"String"
        "product-Info": "M00000",                   //"String"
        "site": "M00000",                           //"String"
        "owner": "M00000",                          //"String"
        "Virtual-Analyzer": [{
            "id": "M00105",                         //"ID-Virtual"
            "id-Profile": "M00103",                 //"ID-Profile"
            "name": "M00000",                       //"String"
            "server-VI": "M00005",                  //"URI"
            "mapper": "M00005",                     //"URI"
            "time-Created": "M00002",               //"Time"
            "Description":[{empty}](0..1)
        }](*),
        "Description":[{empty}](0..1)
```

```
    }](*),
    "Description":[{empty}](0..1)
}
```

B.3.3 物联剖面库默认剖面

```
"Lib-Profile": {                                        //模型主类
    "id": "M00100",                                     //"Identifier"
    "name": "M00000",                                   //"String"
    "address": "M00005",                                //"URI"
    "Access-Info":[{
        "Self-Profile":[{
            "id":"M00103",                              //"ID-Profile"
            "name":"M00000",                            //"String"
            "profile":"M00005",                         //"URI"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Interface":[{
            "id":"M00100",                              //"Identifier"
            "interface":"M00005",                       //"URI"
            "profile":"M00005",                         //"URI"
            "Self-Profile.id":["M00103"](*),            //"ID-Profile"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Qos":[{
            "id":"M00100",                              //"Identifier"
            "name":"M00000",                            //"String"
            "Item-Qos":[{
                "id":"M00100",                          //"Identifier"
                "name":"M00000",                        //"String"
                "value":"M00000",                       //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Right-Qos":[{
                "id":"M00100",                          //"Identifier"
                "name":"M00000",                        //"String"
                "value":"M00000",                       //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Description":[{empty}](0..1)
        }](*),
    "Description":[{empty}](0..1)
```

```
    }](0..1),
    "Profile":[{
        "id":"M00103",                          //"ID-Profile"
        "name":"M00000",                        //"String"
        "type":"M00128",                        //"Profile-Type"
        "profile":"M00005",                     //"URI"
        "profile-Based":"M00103",               //"ID-Profile"
        "lib-Metadata":"M00005",                //"URI"
        "publisher":"M00000",                   //"String"
        "time-Publish":"M00002",                //"Time"
        "Description":[{empty}](0..1)
    }](*),
    "Description":[{empty}](0..1)
}
```

B.3.4 元数据库默认剖面

```
"Lib-Metadata":{                                //模型主类
    "id":"M00100",                              //"Identifier"
    "name":"M00000",                            //"String"
    "address":"M00005",                         //"URI"
    "Access-Info":[{
        "Self-Profile":[{
            "id":"M00103",                      //"ID-Profile"
            "name":"M00000",                    //"String"
            "profile":"M00005",                 //"URI"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Interface":[{
            "id":"M00100",                      //"Identifier"
            "interface":"M00005",               //"URI"
            "profile":"M00005",                 //"URI"
            "Self-Profile.id":["M00103"](*),    //"ID-Profile"
            "Description":[{empty}](0..1)
        }](*),
        "Access-Qos":[{
            "id":"M00100",                      //"Identifier"
            "name":"M00000",                    //"String"
            "Item-Qos":[{
                "id":"M00100",                  //"Identifier"
                "name":"M00000",                //"String"
                "value":"M00000",               //"String"
```

```
                "Description":[{empty}](0..1)
            }](*),
            "Right-Qos":[{
                "id":"M00100",                          //"Identifier"
                "name":"M00000",                        //"String"
                "value":"M00000",                       //"String"
                "Description":[{empty}](0..1)
            }](*),
            "Description":[{empty}](0..1)
        }](*),
        "Description":[{empty}](0..1)
    }](0..1),
    "Description":[{empty}](0..1)
}
```

B.4 访问接口默认剖面

```
"Interface":[{                                          //模型主类
    "id": "M00107",                                     //"ID-Interface"
    "name": "M00000",                                   //"String"
    "provider": "M00005",                               //"URI"
    "timeout": "M00120",                                //"Time-Span"
    "Frame-Header":[{
        "size": "M00001",                               //"Numeric"
        "header": "M00000",                             //"String"
        "Description":[null](0..1)
    }](0..1),
    "Frame-Tail":[{
        "size": "M00001",                               //"Numeric"
        "tail": "M00000",                               //"String"
        "Description":[null](0..1)
    }](0..1),
    "Security": [{
        "key":"M00000",                                 //"String"
        "algorithm": "M00000",                          //"String"
        "Description":[null](0..1)
    }](0..1),
    "Request": {
        "object": "M00150",                             //"Object-Requested"
        "filter": "M00154",                             //"Filter-Expression"
        "setting": "M00155",                            //"Value-Expression"
```

```
        "required": "M00101",                          //"ID-List"
        "Operation": {
            "type": "M00156",                          //"Operate-Right"
            "authentication": "M00000",                //"String"
            "Description":[null](0..1)
        },
        "Description":[null](0..1)
    },
    "Response": {
        "flag": "M00157",                              //"Response-Flag"
        "output": "M00155",                            //"Value-Expression"
        "Piggyback": [{
            "flag": "M00158",                          //"Piggyback-Flag"
            "info": "M00000",                          //"String"
            "Description":[null](0..1)
        }](0..1),
        "Description":[null](0..1)
    },
    "Description":[null](1)
}
```

B.5 元数据默认剖面

```
"Metadata": {                                          //模型主类
    "id": "M00102",                                    //"ID-Metadata"
    "id-Extended":"M00102",                            //"ID-Metadata"
    "name-Eng":"M00000",                               //"String"
    "name-Chn": "M00000",                              //"String"
    "value-Type":"M00053",                             //"Value-Type"
    "conformance":"M00003",                            //"Bool"
    "Value-Constraints":[{
        "size":"M00051",                               //"Cardinality"
        "unit":"M00000",                               //"String"
        "default":"M00000",                            //"String"
        "maximum":"M00050",                            //"Maximum"
        "Enumerated-Value":[{
            "seq":"M00010",                            //"Natural"
            "value":"M00000",                          //"String"
            "Description":[{empty}](0..1)
        }](0*),
        "Description":[{empty}](0..1)
```

```
    }](0..1),
    "Metadata": [{this,
        "Generate-Constraints":[{
            "group": "M00010",                          //"Natural"
            "sequence":"M00010",                        //"Natural"
            "generate - Style":"M00054",                //"G-Style"
            "cardinality":"M00051",                     //"Cardinality"
            "Description":[{empty}](0..1)
        }](1)
    }](*),
    "Aggregate-Constraints":[{
        "unique-Keys": "M00000",                        //"String",
        "order":"M00000",                               //"String",
        "Description":[{empty}](0..1)
    }](0..1),
    "Description":[{empty}](0..1)
}
```

附 录 C
（规范性附录）
物联元数据

C.1 概述

本附录给出本标准中所用到的元数据的定义。

本标准中元数据的定义采用 DSL 形式化语言表达，元数据、元数据的数值值均采用 UCS 字符串表达。

元数据的标识符为 6 位定长字符串，以字母“M”开头，后 5 位为十进制数字。

本标准对元数据标识符的取值范围作了划分，具体如下：

——M00000-M00009 为最基本的元数据；

——M00010-M00049 为通用元数据；

——M00050-M00099 为物联模型中通用的元数据；

——M00100-M00119 为标识符相关的元数据；

——M00120-M00149 为仪器默认模型中所用的元数据；

——M00150-M00199 为访问接口默认模型中所用的元数据；

——M10000 及以上为可供用户扩展使用的范畴。

元数据标识符小于 M10000 且附录中未定义的，为本标准保留的部分。

C.2 基本的元数据

该部分为本标准中最基本的元数据集，本附录定义了 6 个元数据。见表 C.1。

表 C.1 元数据 ID 值的 M00000-M00009 部分

元数据 ID id	英文名 name-Eng	中文名 name-Chn	说明 Description
M00000	String	字符型	任意字符的串接，长度不限
M00001	Numeric	数字型	可进行数学运算的整数、浮点数，长度不限
M00002	Time	时间型	时间格式：年月日时分秒(yyyymmddhhmmss)
M00003	Bool	布尔型	值为 true(真)或 false(假)
M00004	Enumerate	枚举型	枚举类型
M00005	URI	资源型	统一资源标识符(Uniform Resource Identifier)

C.2.1 String 元数据

String 元数据的 DSL 定义：{

"id":"M00000","id-Extended":null,

"name-Eng":"String","name-Chn":"字符型",

"value-Type":"S","conformance":true,
"Description":{"任意字符的串接,长度不限。"}
}
String 数值结构:{"M00000":"M00000"}
String 型数值的例子:{"This is an example"}

C.2.2 Numeric 元数据

Numeric 元数据的 DSL 定义:{
"id":"M00001","id-Extended":"M00000",
"name-Eng":"Numeric","name-Chn":"数字型",
"value-Type":"N","conformance":true,
"Description": {"可进行数学运算的整数、浮点数,长度不限。"}
}
Numeric 数值结构:{"M00001":"M00001"}
Numeric 型数值的例子:{3.1415926}

C.2.3 Time 元数据

Time 元数据的 DSL 定义:{
"id":"M00002","id-Extended":"M00012",
"name-Eng":"Time","name-Chn":"时间型",
"value-Type":"T","conformance":true,
"Value- Constraints":{"size":"14"}
"Description":"时间格式:年月日时分秒(yyyymmddhhmmss)"
}
Time 数值结构:{"M00002":"M00002"}
Time 型数值的例子:{"20000101001030"} //2000 年 1 月 1 日 0 点 10 分 30 秒

C.2.4 Bool 元数据

Bool 元数据的 DSL 定义:{
"id":"M00003","id-Extended":"M00000",
"name-Eng":"Bool","name-Chn":"布尔型",
"value-Type":"B","conformance":true,
"Description":{"值为 true(真)或 false(假)。"}
}
Bool 数值结构:{"M00003":"M00003"}
Bool 型数值的例子:{true}

C.2.5 Enumerate 元数据

Enumerate 元数据的 DSL 定义{
"id":"M00004","id-Extended":"M00000",
"name-Eng":"Enumerate","name-Chn":"枚举型","value-Type":"E","conformance":true,

"Description":{"枚举类型。"}
}
Enumerate 数值结构:{"M00004":"M00004"}

注:Enumerate 为一个抽象类型的元数据,不能直接使用。

C.2.6 URI 元数据

URI 元数据的 DSL 定义:{
"id":"M00005","id-Extended":"M00000",
"name-Eng":"URI","name-Chn":"资源型",
"value-Type":"U","conformance":true,
"Description":{"统一资源标识符(Uniform Resource Identifier)"}
}
URI 数值结构:{"M00005":"M00005"}
URI 型数值的例子:{"file://a:1234/b/c/d.txt"}

C.3 通用的元数据

该部分主要为从最基本元数据集扩展出的通用元数据集,本附录定义了 4 个元数据。见表 C.2。

表 C.2 元数据 ID 值的 M00010-M00049 部分

元数据 ID id	英文名 name-Eng	中文名 name-Chn	说明 Description
M00010	Natural	自然数	自然数
M00011	Integer	整数	整数
M00012	Date	日期	格式:年月日(yyyymmdd)
M00013	Timestamp	时间戳	格式:年月日时分秒毫秒(yyyymmddhhmmssmmm)

C.3.1 Natural 元数据

Natural 元数据的 DSL 定义:{
"id":"M00010","id-Extended":"M00011",
"name-Eng":"Natural","name-Chn":"自然数型",
"value-Type":"N","conformance":true,
"Description":{"自然数"}
}

C.3.2 Integer 元数据

Integer 元数据的 DSL 定义:{
"id":"M00011","id-Extended":"M00001",
"name-Eng":"Integer","name-Chn":"整数型",
"value-Type":"N","conformance":true,

"Description":{"整数。长度不限。"}
}

C.3.3 Date 元数据

Date 元数据的 DSL 定义:{
"id":"M00012","id-Extended":"M00000",
"name-Eng":"Date","name-Chn":"日期型",
"value-Type":"T","conformance":true,
"Value-Constraints":{"size":"8"},
"Description":{"日期格式:年月日(yyyymmdd)。长度 8 字节。"}
}
Date 数值结构:{"M00012":"M00012"}
Date 型数值的例子:{"20170101"} //2017 年 1 月 1 日

C.3.4 Timestamp 元数据

Timestamp 元数据的 DSL 定义:{
"id":"M00012","id-Extended":"M00002",
"name-Eng":"Timestamp","name-Chn":"时间戳型",
"value-Type":"T","conformance":true,
"Value-Constraints":{"size":"17"}
"Description":{"时间戳格式:年月日时分秒毫秒(yyyymmddhhmmssmmm)。"}
}
Timestamp 数值结构:{"M00013":"M00013"}
Timestamp 型数值的例子:{"20000101001030020"} //2000 年 1 月 1 日 0 点 10 分 30 秒零 20 毫秒

C.4 模型通用的元数据

该部分主要为模型中通用的元数据,本附录定义了 5 个元数据,见表 C.3。

表 C.3 元数据 ID 值的 M00050-M00099 部分

元数据 ID id	英文名 name-Eng	中文名 name-Chn	说明 Description
M00050	Maximum	最值型	最值类型,描述数值的最小值、最大值
M00051	Cardinality	基数型	关联基数类型
M00052	Order	排序方式	值为"SL"表示排序从小到大;"LS"表示从大到小
M00053	Value-Type	数值类型	元数据的基本数值类型
M00054	G-Style	生成方式	子元数据组合形成复合元数据的生成方式

C.4.1 Maximum 元数据

Maximum 元数据的 DSL 定义:{

"id":"M00050","id-Extended":"M00000",
"name-Eng":"Maximum","name-Chn":"最值型",
"value-Type":"S","conformance":true,
"Description":{"描述数值的最小值、最大值。格式为"下限..上限",下限和上限均为实数。下限没有限制时,格式为"*..上限";在上限没有限制时,格式为"下限..*",在下限/上限均没有限制时,为"*"或 null"}
}
Maximum 数值结构:{"M00050":"M00050"}
Maximum 型数值的例子:{"-10..10"}

C.4.2 Cardinality 元数据

Cardinality 元数据的 DSL 定义:{
"id":"M00051","id-Extended":"M00050",
"name-Eng":"Cardinality","name-Chn":"最值型",
"value-Type":"S","conformance":true,
"Description":{"关联基数"}
}
Cardinality 数值结构:{"M00051":"M00051"}
Cardinality 型数值的例子:{"1..5"}

C.4.3 Order 元数据

Order 元数据的 DSL 定义:{
"id":"M00052","id-Extended":"M00004",
"name-Eng":"Order","name-Chn":"最值型",
"value-Type":"E","conformance":true,
"value-Constraints":{
"size":"2","unit":null,"default":null,"maximum":"0..2",
"Enumerated-Value":[
{"seq":0,"value":"SL","Description":"升序"},
{"seq":1,"value":"LS","Description":"降序"}]
"Description":{"值为"SL"表示排序从小到大;"LS"表示从大到小"}}
}
Order 数值结构:{"M00052":"M00052"}
Order 型数值的例子:{"SL"}

C.4.4 Value-Type 元数据

Value-Tyle 元数据的 DSL 定义:{
"id":"M00053","id-Extended":"M00004",
"name-Eng":"Value-Tyle","name-Chn":"数值类型",
"value-Type":"E","conformance":true,

```
    "value-Constraints":{
    "size":null,"unit":null,"default":null,"maximum":null,
    "Enumerated-Value":[
        {"seq":0,"value":"S","Description":"字符型"},
        {"seq":1,"value":"N","Description":"数字型"},
        {"seq":2,"value":"T","Description":"时间型"},
        {"seq":3,"value":"B","Description":"布尔型"},
        {"seq":4,"value":"U","Description":"资源型"},
        {"seq":5,"value":"E","Description":"枚举型"},
        {"seq":6,"value":"G","Description":"复合型"}]},
    "Description": {"元数据的数值类型"}
}
Value-Tyle 数值结构:{"M00053":"M00053"}
Value-Tyle 型数值的例子:{"S"} //字符型
```

C.4.5 G-Style 元数据

```
G-Style 元数据的 DSL 定义:{
    "id":"M00054","id-Extended":"M00000",
    "name-Eng":"G-Style","name-Chn":"生成方式",
    "value-Type":"G","conformance":true,
    "value-Constraints":{
    "size":null,"unit":null,"default":null,"maximum":null,
    "Enumerated-Value":[
            {"seq":0,"value":"Sequential","Description":"顺序生成方式"},
            {"seq":1,"value":"Selective","Description":"选择生成方式"}
            {"seq":2,"value":"Intersective","Description":"交集生成方式"},
            {"seq":3,"value":"Concatenative","Description":"串接生成方式"}]},
    "Metadata":{"id": "M00000"}, //当生成方式为"Concatenative"时,该部分串接时的串接符
    "Description":{"子元数据组合形成复合元数据的生成方式。"}
}
G-Style 数值结构:{"M00054":"M00054"}
G-Style 型数值的例子:{"Concatenative","."} //串接生成方式,串接符为"."
```

C.5 标识符相关的元数据

该部分主要为标识符相关的元数据,本附录定义了 8 个元数据,见表 C.4。

表 C.4　元数据 ID 值的 M00100-M00119 部分

元数据 ID id-Metadata	英文名 name-Eng	中文名 name-Chn	说明 Description
M00100	Identifier	标识符型	由英文字母开头的字符串,长度不限
M00101	ID-List	标识符链表	以标识符为基本元素的链表
M00102	ID-Metadata	元数据 ID	大写 M 开头,长 6 个字节,后 5 位为十进制数字
M00103	ID-Profile	剖面 ID	大写字母 P 开头,长 6 个字节,后 5 位为十进制数字
M00104	ID-TreeNode	树结构节点 ID	树形结构对象的内部元素的标识符
M00105	ID-Virtual	虚拟仪器 ID	大写 V 开头,长 6 个字节,后 5 位为十进制数字
M00106	ID-Entity	实体仪器 ID	大写 E 开头,长 6 个字节,后 5 位为十进制数字
M00107	ID-Interface	接口 ID	大写 I 开头,长 6 个字节,后 5 位为十进制数字

C.5.1　Identifier 元数据

Identifier 元数据的 DSL 定义:{
"id":"M00100","id-Extended":"M00000",
"name-Eng":"Identifier","name-Chn":"标识符型",
"value-Type":"S","conformance":true,
"Value-Constraints":{"size":"6"},
"Description":{"由英文字母开头的字符串,长度不限。"}
}
Identifier 数值结构:{"M00100":"M00100"}
Identifier 型数值的例子:{"E00123"}

C.5.2　ID-List 元数据

ID-List 元数据的 DSL 定义:{
"id":"M00101","id-Extended":"M00100",
"name-Eng":"ID-List","name-Chn":"标识符链表",
"value-Type":"G","conformance":true,
"Value-Constraints":null,
"Metadata":[{"id":"M00100"}](*),
"Description":{"以标识符为基本元素的链表。长度不限。"}
}
ID-List 数值结构:{"M00101":"M00101"}
ID-List 型数值的例子:{"E00011","E00012","E00013"}

C.5.3　ID-Metadata 元数据

ID-Metadata 元数据的 DSL 定义:{
"id":"M00102","id-Extended":"M00100",

"name-Eng":"ID-Metadata","name-Chn":"元数据 ID",
"value-Type":"S","conformance":true,
"Value-Constraints":{"size":"6"},
"Description":{"仪器元数据的标识符。长 6 个字节,大写 M 开头,后 5 位为十进制数字。"}
}
ID-Metadata 数值结构:{"M00102":"M00102"}
ID-Metadata 型数值的例子:{"M00011"}

C.5.4 ID-Profile(ID-Ventity、ID-Entity、ID-Interface)元数据

ID-Profile、ID-Virtual、ID-Entity 和 ID-Interface 元数据的定义与 ID-Metadata 类似,在此不一一给出。

C.5.5 ID-TreeNode 元数据

ID-TreeNode 元数据的 DSL 定义:{
"id":"M00104","id-Extended":"M00000",
"name-Eng":"ID-TreeNode","name-Chn":"树结构节点 ID",
"value-Type":"G","conformance":true,
"Value-Constraints":null,{
"Metadata":{"id": "M00000"},
"Generate-Constraints":{
"group": 0,
"sequence":0,
"generate - Style":{"Sequential","."} //串接符为"."
"cardinality":"*",
"Description": null}},
"Description":{"树形结构对象的内部元素的标识符。长度不限。"}
}
ID-TreeNode 数值结构:{"M00104":"M00104"}
ID-TreeNode 型数值的例子:{"Analyzer.name"}

C.6 默认剖面相关的元数据

该部分主要为默认剖面相关的元数据,本附录定义了 9 个元数据,见表 C.5。

表 C.5 元数据 ID 值的 M00120-M00149 部分

元数据 ID id-Metadata	英文名 name-Eng	中文名 name-Chn	说明 Description
M00120	Time-Span	时间长度	时间的长度,单位为毫秒
M00121	State	运行状态	仪器的当前状态。枚举值
M00122	Time-Serial	时序数据	一个基本的时间序列数据,格式:{时间戳,字符串}

表 C.5（续）

元数据 ID id-Metadata	英文名 name-Eng	中文名 name-Chn	说明 Description
M00123	Time-Series	时序数据流	时序数据形成的数据流
M00124	Stream-Type	字节流类型	字节流类型。枚举值
M00125	Stream-Data	字节流	超长数据的二进制流，如多媒体、二进制文件等
M00126	Compose-Type	合成方式	合成方式。枚举值：顺序，选择，循环和同步
M00127	Compose-Rule	合成规则	表示多个对象组合为一个对象时的组合规则。 组合方式主要是：顺序，选择，循环和同步
M00128	Profile-Type	剖面类型	枚举值：仪器，注册中心，访问接口和元数据

C.6.1 Time-Span 元数据

Time-Span 元数据的 DSL 定义：{

"id":"M00120","id-Extended":"M00010",

"name-Eng":"Time-Span","name-Chn":"时间长度",

"value-Type":"N","conformance":true,

"value-Constraints":{ "size": null,"unit":"ms"},

"Description":{"时间长度，单位为毫秒。"}

}

Time-Span 数值结构：{"M00120":"M00120"}

Time-Span 型数值的例子：{1200}

C.6.2 State 元数据

State 元数据的 DSL 定义：{

"id":"M00121","id-Extended":"M00004",

"name-Eng":"State","name-Chn":"仪器状态",

"value-Type":"E", "conformance":true,

"value-Constraints":{

"size": null,"unit":null,"default":null,"maximum": null,

"Enumerated-Value":[

{"seq":0,"value":"O","Description":"离线状态(Offline)，与网络断开"},

{"seq":1,"value":"P","Description":"失电状态(PowerOff)，电源失电"},

{"seq":2,"value":"T","Description":"电停机状态(TurnOff)，已关机但电源有电"},

{"seq":3,"value":"S","Description":"启动状态(Startup)，正在启动过程中"},

{"seq":4,"value":"R","Description":"就绪状态(Ready)，已就绪可用"},

{"seq":5,"value":"I","Description":"使用状态(InUse)，正在使用中"},

{"seq":6,"value":"A","Description":"异常状态(Abnormal)，有异常"},

{"seq":7,"value":"U","Description":"未知状态(Unknown)，状态未知"},

{"seq":8",value":"C","Description":"关闭状态(Close)，当前虚拟仪器已关闭"},

{"seq":9,"value":"N","Description":"其他状态(Others)"}]},
"Description": {"仪器的当前状态。"}
}
State 数值结构:{"M00121":"M00121"}
State 型数值的例子:{"I"}

C.6.3 Time-Serial 元数据

Time-Serial 元数据的 DSL 定义:{
"id":"M00122","id-Extended":"M00000",
"name-Eng":"Time-Serial","name-Chn":"时间序列数据",
"value-Type":"G","conformance":true,
"Value-Constraints":null,
{ "Metadata":{"id": "M00002"}, //时间戳
"Generate-Constraints":{
"group":0,
"sequence":0,
"generate - Style":"Sequential",
"cardinality":"1",
"Description":null}},
{ "Metadata":{"id": "M00000"}, //字符串
"Generate-Constraints":{
"group":0,
"sequence":1,
"generate - Style":"Sequential",
"cardinality":"1",
"Description": null}},
"Description":{"由{时间戳,字符串}组合成的数据结构,用于定义一个基本的时间序列数据。"}
}
Time-Serial 数值结构:{"M00002":"M00002","M00000":"M00000" }
Time-Serial 型数值的例子:{"20170101083005" ,"13657.8"}

C.6.4 Time-Series 元数据

Time-Series 元数据的 DSL 定义:{
"id":"M00123","id-Extended":"M00000",
"name-Eng":"Time-Series","name-Chn":"时序数据流",
"value-Type":"G","conformance":true,
"Value-Constraints":null,
{ "Metadata":{"id":"M00002"}, //时间戳,表达起始时间
"Generate-Constraints":{
"group":0,
"sequence":0,

"generate - Style":"Sequential",
"cardinality":"1",
"Description":null}},
{ "Metadata":{"id":"M00010"}, //数据流中的时序数据的数据量
"Generate-Constraints":{
"group":0,
"sequence":1,
"generate - Style":" Sequential ",
"cardinality":"1",
"Description":null}},
["Metadata":{"id":"M00122"}](*), //时序数据
"Description":{"时序数据形成的数据流。"}
}

Time-Series 数值结构:{{"M00002":"M00002","M00120":"M00120"}, [{"M00122":"M00122"}](0..*)}

Time-Series 型数值的例子:{{"20170101083005", 2}, { "20170101083005","13657.8"}, {"20170101083035", "13660.8"}}

C.6.5 Stream-Type 元数据

Stream-Type 元数据的 DSL 定义:{
"id":"M00124","id-Extended":"M00004",
"name-Eng":"Stream-Type","name-Chn":"字节流类型",
"value-Type":"E","conformance":true,
"value-Constraints":{
"size": null,"unit":null,"default":null,"maximum":null,
"Enumerated-Value":[
{"seq":0,"value":"Text","Description":"文本"},
{"seq":1,"value":"Audio","Description":"音频"},
{"seq":2,"value":"Images","Description":"图片"},
{"seq":3,"value":"Animations","Description":"动画"},
{"seq":4,"value":"Video","Description":"视频"},
{"seq":5,"value":"","Description":"其他"}]},
"Description": {"字节流类型。枚举值。"}
}

Stream-Type 数值结构:{"M00124":"M00124"}

Stream-Type 型数值的例子:{"Video"}

C.6.6 Stream-Data 元数据

Stream-Data 元数据的 DSL 定义:{
"id":"M00125","id-Extended":"M00000",
"name-Eng":"Stream-Data","name-Chn":"字节流",

```
    "value-Type":"G","conformance":true,
    "Value-Constraints":null,
    {   "Metadata":{"id":"M00005"},        //URI
        "Generate-Constraints":{
                    "group":0,
                    "sequence":0,
                    "generate - Style":"Sequential",
                    "cardinality":"1",
                    "Description":null}},
    {   "Metadata":{"id": "M00124"},  //字节流类型
        "Generate-Constraints":{
                    "group":0,
                    "sequence":1,
                    "generate - Style":"Sequential",
                    "cardinality":"1",
                    "Description":null}}
    "Description": {"长度大的二进制流,如多媒体数据。"}
}
Stream-Data 数值结构:{"M00005":" M00005", "M00124":"M00124"}
Stream-Data 型数值的例子:{"file://a:1234/b/c/d.txt","Text"}
```

C.6.7 Compose-Type 元数据

```
Compose-Type 元数据的 DSL 定义:{
    "id":"M00126","id-Extended":"M00004",
    "name-Eng":"Compose-Type","name-Chn":"合成方式",
    "value-Type":"E","conformance":true,
    "value-Constraints":{
        "size": null,"unit":null,"default":null,"maximum":null,
        "Enumerated-Value":[
            {"seq":0,"value":"Sequential","Description":"顺序"},
            {"seq":1,"value":"Selective","Description":"选择"},
            {"seq":2,"value":"Loop","Description":"循环"},
            {"seq":3,"value":"Synchronous ","Description":"同步"}]},
    "Description":{"合成方式。枚举值:顺序,选择,循环和同。"}
}
Compose-Type 数值结构:{"M00126":"M00126"}
Compose-Type 型数值的例子:{"Selective"}
```

C.6.8 Compose-Rule 元数据

```
Compose-Rule 元数据的 DSL 定义:{
    "id":"M00127","id-Extended":"M00000",
```

"name-Eng":"Compose-Rules","name-Chn":"合成规则",
"value-Type":"G","conformance":true,
"Value-Constraints":null,
{ "Metadata":{"id":"M00100"}, //对象 ID
"Generate -Constraints":{
"group":0,
"sequence":0,
"generate- Style":"Selective",
"cardinality":"*",
"Description":null}},
{ "Metadata":{this},
"Generate -Constraints":{
"group":0,
"sequence":1,
"generate- Style":"Selective",
"cardinality":"1",
"Description":null}},
{ "Metadata":{"id": "M00126"}, //合成操作符
"Generate -Constraints":{
"group":1,
"sequence":0,
"generate - Style":"Sequential",
"cardinality":"1",
"Description":null}}
"Description":{"表示多个对象组合为一个对象时的组合规则:顺序,选择,循环和同步。"}
}
Compose-Rule 数值结构:{[{"M00100":"M00100",this}](*),"M00126":"M00126"},
Compose-Rule 型数值的例子:{{{"测量活动 1","测量活动 2"},"Sequential"},
{{测量活动 3","测量活动 4"},"Selective"} ,"Sequential"}}

C.6.9 Profile-Type 元数据

Profile-Type 元数据的 DSL 定义:{
"id":"M00128","id-Extended":"M00004",
"name-Eng":"Profile-Type","name-Chn":"剖面类型",
"value-Type":"E","conformance":true,
"value-Constraints":{
"size": null,"unit":null,"default":null,"maximum": null,
"Enumerated-Value":[
{"seq":0,"value":"Instruments","Description":"仪器"},
{"seq":1,"value":"Registry","Description":"注册中心"},
{"seq":2,"value":"Interface","Description":"访问接口"},

{"seq":3,"value":"Metatdata","Description":"元数据"}]},
"Description": {"枚举值:仪器,注册中心,访问接口和元数据。"}
}
Profile-Type 数值结构:{"M00128":"M00128"}
Profile-Type 型数值的例子:{"Metatdata"}

C.7 访问接口相关的元数据

该部分主要为访问接口相关的元数据,本附录定义了 9 个元数据,见表 C.6。

表 C.6 元数据 ID 值的 M00150-M00199 部分

元数据 ID id-Metadata	英文名 name-Eng	中文名 name-Chn	说明 Description
M00150	Object-Requested	访问对象	实体仪器物联涉及的对象的标识符
M00151	Value-Matcher	数值匹配符	两个数值的匹配符。包括:"==" (等于),"<>" (不等于),">" (大于),"<" (小于),">=" (大于或等于),"<=" (小于或等于)
M00152	Logic-Operator	逻辑操作符	逻辑操作符:"&&" (与),"||" (或),"!" (非)
M00153	Filter	过滤子	过滤条件或查询语句的基本元素。由标识符、数值匹配符和匹配值组成
M00154	Filter-Expression	查询表达式	一个由过滤子和逻辑操作符组合成的 DSL 表达式
M00155	Value-Expression	数值表达式	一个 DSL 表达式
M00156	Operate-Type	操控方式	对访问对象的操控方式。枚举值:"C" (Create) 为增加;"R" (Read) 读取;"U" (Update) 修改;"D" (Delete) 删除;"A" (Authenticate) 验证;null 表示无任何操控
M00157	Response-Flag	接口响应标志	描述接口请求的处理结果的标志
M00158	Piggyback-Flag	捎带标志	接口响应中捎带标志。枚举值:"A" (Alarm) 报警;"S" (Setting) 设置

C.7.1 Object-Requested 元数据

Object-Requested 元数据的 DSL 定义:{
"id":"M00150","id-Extended": M00000",
"name-Eng":"Object-Requested","name-Chn":"访问对象",
"value-Type":"G","conformance":true,
{ "Metadata":{"id":"M00004",null,
"value-Constraints":{
"Enumerated-Value":[
{"seq":0,"value":"P","Description":"剖面"},
{"seq":3,"value":"I","Description":"实例"}]}},
"Generate-Constraints":{
"group":0,

"sequence":0,
"generate - Style":"Selective",
"cardinality":"1",
"Description":null}},
{ "Metadata":{"id":"M00100"} //id 为某物联标准件管理的特定对象标识符。
"Generate-Constraints":{
"group":0,
"sequence":1,
"generate - Style":"Selective",
"cardinality":"1",
"Description":null}},
"Description":{"被访问的对象的相关信息"}
}
Object-Requested 数值结构:{"M00150":"M00150"}
Object-Requested 型数值的例子:{"I"}

C.7.2 Value-Matcher 元数据

Valu e-Matcher 元数据的 DSL 定义:{
"id":"M00151","id-Extended": M00004",
"name-Eng":"Value-Matcher","name-Chn":"数值匹配符",
"value-Type":"E","conformance":true,
"value-Constraints":{
"size": null,"unit":null,"default":null,"maximum": null,
"Enumerated-Value":[
{"seq":0,"value":"==","Description":"等于"},
{"seq":1,"value":"! =","Description":"不等于"},
{"seq":2,"value":">","Description":"大于"},
{"seq":3,"value":"<","Description":"小于"},
{"seq":2,"value":">=","Description":"大于等于"},
{"seq":3,"value":"<= ","Description":"小于等于"},]},
"Description":{"两个数值的匹配符"}
}
Value-Matcher 数值结构:{"M00151":"M00151"}
Value-Matcher 型数值的例子:{"=="}

C.7.3 Logic-Operator 元数据

Logic-Operator 元数据的 DSL 定义:{
"id":"M00152","id-Extended":"M00004",
"name-Eng":"Logic-Operator","name-Chn":"逻辑操作符",
"value-Type":"E","conformance":true,
"value-Constraints":{

```
        "size": null,"unit":null,"default":null,"maximum": null,
        "Enumerated-Value":[
            {"seq":0,"value":"&&","Description":"与"},
            {"seq":1,"value":"||","Description":"或"},
        {"seq":2,"value":"!","Description":"非"}]},
    "Description": {"查询表达式的基本元素,过滤子间的逻辑操作符"}
}
Logic-Operator 数值结构:{"M00152":"M00152"}
Logic-Operator 型数值的例子:{"&&"}
```

C.7.4 Filter 元数据

```
Filter 元数据的 DSL 定义:{
    "id":" M00153","id-Extended":"M00000",
    "name-Eng":"Filter","name-Chn":"过滤子",
    "value-Type":"G","conformance":true,
    "Value-Constraints":null,{
        {"Metadata":{"id": "M00100"}, //标识符为某数据项的 ID
            "Generate-Constraints":{
                "group":0,
                "sequence":0,
                "generate- Style":"Sequential",
                "cardinality":"1",
                "Description":null}},
        {"Metadata":{"id": "M00151"},
            "Generate-Constraints":{
                "group":0,
                "sequence":2,
                "generate - Style":"Sequential",
                "cardinality":"1",
                "Description":null}}.
        {"Metadata":{"id": "M00000"},
            "Generate-Constraints":{
                "group":0,
                "sequence":2,
                "generate - Style":"Sequential",
                "cardinality":"1",
                "Description":null}}
    },
    "Description":{"过滤条件或查询语句的基本元素。由标识符、数值匹配符和匹配值组成。"}
}
Filter 数值结构:{"M00100":"M00100","M00151":"M00151","M00000":"M00000"}
```

Filter 型数值的例子：{ "Analyzer.name,"==","我的色谱仪"}

C.7.5 Filter-Expression 元数据

Filter-Expression 元数据的 DSL 定义：{

"id":"M00154","id-Extended":"M00000",

"name-Eng":"Filter-Express","name-Chn":"查询表达式",

"value-Type":"G","conformance":true,

"Value-Constraints":null,{

{"Metadata":{"id": "M00153"},

"Generate-Constraints":{

"group":0,

"sequence":0,

"generate- Style":" Selective",

"cardinality":"*",

"Description":null}},

{"Metadata":{this},

"Generate-Constraints":{

"group":0,

"sequence":1,

"generate - Style":"Selective",

"cardinality":"1",

"Description":null}},

{"Metadata":{"id": "M00152"},

"Generate-Constraints":{

"group":1,

"sequence":0,

"generate - Style":"Sequential",

"cardinality":"1",

"Description":null}}

},

"Description":{"过滤或查询表达式的类型。以 Filter 为基本元素组成的表达式。"}

}

Filter-Expression 数值结构：{[{"M00153":"M00153",this}](*),"M00152":"M00152}"

Filter-Expression 数值的例子：{{{"age",">","20"},{"age","<","30"},{"&&"}},{"salary",">","5000"},{"&&"}}}

C.7.6 Operate-Type 元数据

Operate-Type 元数据的 DSL 定义：{

"id":"M00156","id-Extended":"M00004",

"name-Eng":"Operate-Type","name-Chn":"操控方式",

"value-Type":"E","conformance":true,

```
    "value-Constraints":{
        "size": null,"unit":null,"default":null,"maximum":null,
        "Enumerated-Value":[
            {"seq":0,"value":"C","Description":"增加"},
            {"seq":1,"value":"R","Description":"读取"},
            {"seq":2,"value":"U","Description":"修改"},
            {"seq":3,"value":"D","Description":"删除"},
            {"seq":2,"value":"A","Description":"验证"},
            {"seq":3,"value":NULL,"Description":"无任何权限"}]},
    "Description": {"对访问对象的操控方式"}
}
Operate-Right 数值结构:{"M00156":"M00156}
Operate-Right 型数值的例子:{"C"}
```

C.7.7 Response-Flag 元数据

```
Response-Flag 元数据的 DSL 定义:{
    "id":" M00157", "id-Extended": " M00000",
    "name-Eng":"Response-Flag", "name-Chn":"接口响应标志",
    "value-Type":"G", "conformance":true,
    "Value-Constraints":null,{
        {"Metadata":{ "id": "M00003"},                    //布尔型
            "Generate-Constraints":{
                "group": 0,
                "sequence":0,
                "generate- Style":"Sequential",
                "cardinality":"1",
                "Description": null}},
        {"Metadata":{"id": "M00000"},                     //接口响应信息
            "Generate-Constraints":{
                "group": 0,
                "sequence":1,
                "generate - Style":"Sequential",
                "cardinality":"1",
                "Description": null}}
    },
    "Description":{"描述接口请求的处理结果的标志。"}
}
Response-Flag 数值结构:{{"M00103":"M00103"},{"M00000":"M00000"}}
Response-Flag 型数值的例子:{true,"接口响应成功!"}
```

C.7.8 Piggyback-Flag 元数据

```
Piggyback-Flag 元数据的 DSL 定义:{
```

"id":"M00158","id-Extended":"M00000",
"name-Eng":"Piggyback-Flag","name-Chn":"捎带标志",
"value-Type":"E","conformance":true,
"value-Constraints":{
"size": null, "unit":null, "default":null, "maximum": null,
"Enumerated-Value":[
{"seq":0,"value":"A","Description":"报警"},
{"seq":1,"value":"S","Description":"设置"},]},
"Description": {"接口响应中捎带信息类型。枚举值:"A"(Alarm)为报警;"S"(Setting)为设置。"}
}

Piggyback-Flag 数值结构:{"M00158":"M00158}

Piggyback-Flag 型数值的例子:{"A"}

参 考 文 献

[1] GB/T 18793—2002 信息技术 可扩展置标语言(XML)1.0

[2] GB T 25644—2010 信息技术 软件工程 可复用资产规范

[3] ISO/IEC 8824 信息技术 抽象语法标记(ASN.1)[Information technology—Abstract syntax notation one (ASN.1)]

[4] ISO/IEC 11404 信息技术通用数据类型(GPD)[Information technology—General purpose datatypes(GPD)]

[5] ISO/IEC 14977 信息技术 语法元语言(扩展的 BNF)(Information technology—Syntactic metalanguge—Extended BNF)

[6] IETF RFC 8259 The JavaScript Object Notation (JSON) Data Interchange Format

[7] OASIS Universal Description,Discovery and Integration(UDDI).https://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

[8] OMG Data Distribution Service.https://www.omg.org/omg-dds-portal/

[9] OMG Meta Object Facility(MOF)

[10] OMG Common Warehouse Metamodel(CWM)

[11] OLE for Process Control(OPC),OPC 基金会.http://opcfoundation.cn/default.aspx