ICS 35.110
L 79

中华人民共和国国家标准

GB/T 36451—2018/ISO/IEC/IEEE 18880:2015

信息技术　系统间远程通信和信息交换 社区节能控制网络协议

Information technology—Telecommunications and information exchange between systems—Community energy-saving control network protocol

(ISO/IEC/IEEE 18880:2015, Information technology—Ubiquitous green community control network protocol, IDT)

2018-06-07 发布　　2019-01-01 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 …… Ⅲ
引言 …… Ⅳ
1　概述 …… 1
1.1　范围 …… 1
1.2　目的 …… 1
2　规范性引用文件 …… 1
3　术语和定义、缩略语 …… 2
3.1　术语和定义 …… 2
3.2　缩略语 …… 2
4　架构 …… 3
4.1　UGCCNet 架构 …… 3
4.2　典型通信过程 …… 4
4.3　网络设计 …… 5
4.4　系统模型和部署 …… 5
4.5　点 …… 7
5　通信协议 …… 8
5.1　综述 …… 8
5.2　组件与组件之间的通信协议 …… 8
5.3　组件与注册器的通信协议 …… 10
6　应用程序编程接口 …… 11
6.1　综述 …… 11
6.2　传输的数据结构 …… 11
6.3　组件的访问接口 …… 12
6.4　注册器的访问接口 …… 14
7　data 和 query 模型 …… 17
7.1　总则 …… 17
7.2　点集合树结构的点管理 …… 17
7.3　点集合树结构的查询模式 …… 17
8　数据结构 …… 19
8.1　总则 …… 19
8.2　对象类和 XML 元素的命名规则 …… 19
8.3　组件之间通信协议的数据结构 …… 19
8.4　组件与注册器之间通信协议的数据结构 …… 25
9　协议绑定 …… 31
10　安全考虑 …… 31

附录 A（资料性附录） UGCCNet 通信的典型序列 …… 32
附录 B（资料性附录） 典型的设施网络系统部署 …… 34
附录 C（资料性附录） query 方法和 data 方法应用指南 …… 35
附录 D（资料性附录） 组件访问接口的 Web 服务描述语言 …… 39
附录 E（资料性附录） 注册器访问接口的 Web 服务描述语言 …… 45
附录 F（资料性附录） 组件与组件通信的错误类型 …… 51
附录 G（资料性附录） 组件与注册器通信的错误类型 …… 52

前　言

本标准按照 GB/T 1.1—2009 给出的规则起草。

本标准使用翻译法等同采用 ISO/IEC/IEEE 18880:2015《信息技术　泛在绿色社区控制网络协议》。

与本标准中规范性引用的国际文件有一致性对应关系的我国文件如下：

——GB/T 7408—2005　数据元和交换格式　信息交换　日期和时间表示法(ISO 8601:2000，IDT)

本标准还做了下列编辑性修改：

——为与国家标准体系协调一致，对标准名称进行了改变，改为《信息技术　系统间远程通信和信息交换　社区节能控制网络协议》。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准由全国信息技术标准化技术委员会(SAC/TC 28)提出并归口。

本标准起草单位：中国电子技术标准化研究院、北京天地互连信息技术有限公司、深圳赛西信息技术有限公司、中国电信股份有限公司北京研究院、下一代互联网关键技术和评测北京市工程研究中心有限公司、北京交通大学。

本标准主要起草人：寇宏、宋阳、刘东、卓兰、李文杰、张宏科、郜帅、余晖。

引　言

本标准定义了建设数字社区的重要系统组件，包括用于现场总线网络的网关，用于搭建数据共享平台的数据存储器以及应用单元，可用于建设楼宇级及城市范围内的泛在设施网络基础设施。本标准定义了在IPv4/IPv6网络环境下，组件(网关，存储，应用单元)之间互联和数据交换的通信协议，能够将多样设备、存储器及包括集中管理、节能降耗、环境监测、警报系统等在内的应用服务进行集成。

目前普遍认为，在建筑、房屋、工厂内部部署设施网络是实现能源管理以及节能目标的有效工具。基于TCP/IP协议对设施进行组网则可以实现建筑物范围内甚至是全市范围内的能源管理。然而，目前大多数的系统是独立研制开发，独立部署运营的，这使得安装和运行成本相当高。

通常情况下，为了更好地在现场总线中通过互联网连接传感器以及执行器，我们引入了网关的设计。然而，当前我们要求这种规模的设施网络应用不能仅仅是简单地连接设备，这些新兴应用在实际部署实现中，通常要具备(1)大容量的存储器，用于存储传感器读数的历史数据，(2)支持交互操作的用户接口，(3)上报系统，以及(4)数据分析器。

这些系统组件之间能够相互协作，特别是在具有能源感知功能的设施网络中。然而由于这些系统组件是独立开发并独立集成的，因此如果没有对各系统之间进行专门分析、集成和操作处理，它们之间很难直接实现协作或互操作。

如果定义一个通用的通信协议，能够实现这些系统组件之间的互操作，提高设施网络部署的效率，同时可以减少系统集成和互操作性管理的成本，使设施网络能够应用于中小规模的建筑中，甚至部署在房屋内。对于产品供应商，生产的组件无需做任何定制化的改动，仍可销往世界各地，有时用合理的成本就可以实现大规模的生产。

为了实现整幢建筑物范围内，甚至是城市范围的能源管理，IEEE P18880工作组启动了泛在绿色社区网络控制协议项目(UGCCNet)，它规范了设施网络的远程控制架构。本项目的范围和目的是通过规范设施网络组件之间(即设备接入网关，数据存储器和应用单位)的通信协议，实现组件之间的互操作，从而构建基于互联网模式的设施网络基础架构。制定本标准是为了支持设施网络组件之间的互操作性和组件开发的开放性。首先，通过简单的组件模型抽象出通用的设施网络组件。然后，本标准定义了组件之间的通信协议。为了支持组件之间的自主协作，本标准还引入了注册机制。

本标准定义了通信的基础框架，旨在构建一个新的网络用于设施的更新，下一代的设施管理和中小型规模设施网络的节能。本标准以实现节能和管理平台的集成化为目的，从过去的设施管理扩展到运营级管理。此外，本标准定义的基础框架还可用于系统级的协作。

信息技术　系统间远程通信和信息交换 社区节能控制网络协议

1　概述

1.1　范围

本标准定义了在IPv4/IPv6网络环境下，组件(网关、存储、应用单元)之间互联和数据交换的通信协议，在通用的数字化基础设施上采用开放的应用接口实现对多厂商设备的兼容。本标准定义了建设数字社区的重要系统组件，包括用于现场总线网络的网关，用于搭建数据共享平台的存储器以及应用单元，可用于建设楼宇间及城市范围内的泛在设施网络基础设施。本标准允许多家服务提供商和集成商对基础设施进行分布式运营，定义了支持分布式基础设施互操作的组件管理协议。本标准也考虑了相应的安全要求，以保证数据的安全性和完整性。

1.2　目的

本标准适用于建设能够对能源的使用情况进行良好管理且利用率高的绿色社区，并且允许包括中小规模的不同现场网络、数据共享平台和应用单元在内的多楼宇设施实现互连。基于此标准的产品能够实现泛在信息的感知、存储和展示，例如，产能和用能情况、环境状态及信息、人体活动、暖通空调的工作状态、电灯系统、天气、报警、数据分析、数据预测等。通过系统组件之间的集成互操作，本标准可提供并分享数据平台，包括实现与控制系统的协同。

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。

ISO 8601　数据元交换格式　信息交换　日期和时间的表示(Data elements and interchange formats—Information interchange—Representation of dates and times)[1)]

IETF RFC 793　传输控制协议[Transmission Control Protocol (TCP)][2)]

IETF RFC 2460　互联网协议第六版[Internet Protocol, Version 6 (IPv6) Specification]

IETF RFC 3986　统一资源标识符：通用语法[Uniform Resource Identifiers (URI): Generic Syntax]

W3C　可扩展置标语言[Extensible Markup Language (XML) 1.0][3),4)]

W3C　XML的命名空间1.1版本(Namespaces in XML 1.1)

W3C　SOAP协议1.2版本　第1部分：消息传递框架(SOAP Version 1.2—Part 1: Messaging framework)

W3C　XML概要　第1部分：结构(XML Schema—Part 1: Structures)

1)　ISO出版物可从ISO官方网站获取(http://www.iso.org/)。

2)　IETF文本(RFC文档)可在网站http://www.rfc-archive.org/下载。

3)　W3C出版物可从W3C万维网联盟获取(http://www.w3.org/)。

4)　W3C是万维网联盟的商标(在多个国家注册)；W3C的商标由它的总部机构美国麻省理工大学(MIT)、欧洲数学与信息学研究联盟(ERCIM)、日本庆应大学(Keio University)和中国北京航空航天大学注册和持有。

W3C XML概要 第2部分:数据类型(XML Schema—Part 2: Datatypes)

3 术语和定义、缩略语

3.1 术语和定义

下列术语和定义适用于本文件。本条款未定义的术语应查阅IEEE线上标准词典。[5)]

3.1.1

访问控制 access control

允许经授权的访问以及对资源使用的方法。

3.1.2

执行器 actuator

接收数据信号,将其转换成物理动作的装置。

3.1.3

可扩展置标语言命名空间 eXtensible Markup Language (XML) namespace

一种区分具有相同名称但不同含义的XML元素及属性的方法。URL作为"本地名称"的前缀,可确保元素或属性名称的唯一性。URL仅作为一种创建唯一前缀的方法,不一定真正指向互联网上的某个实际页面。

3.1.4

传感器 sensor

一种将物理、生物或化学参数转化为数字信号的转换器。

3.1.5

全局唯一标识符 universally unique identifier

UUID

在某些已定义空间内保证唯一性的标识符。

注:在本标准中,除非另有说明,传输数据结构中的查询表达式和查找表达式都由UUID标识。[6)]

3.2 缩略语

下列缩略语适用于本文件。

AAA:认证,授权和计费(Authentication, Authorization, and Accounting)

API:应用程序编程接口(Application Programming Interface)

EPR :端点引用(End-Point Reference)

FTP:文件传输协议(File Transfer Protocol)

HTTP:超文本传输协议(Hypertext Transfer Protocol)

HTTPS:超文本传输安全协议(Hypertext Transfer Protocol Secure)

IP:互联网协议(Internet Protocol)

IPv6:互联网协议第六版本(Internet Protocol Version 6)

NAT: 网络地址转换(Network Address Translation)

RPC:远程过程调用(Remote Procedure Call)

SIP:会话发起协议(Session Initiation Protocol)

5) IEEE线上标准词典订阅地址:http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html。

6) 标准文本、表格和图形中的注释仅供参考,不包含实施本标准的必然要求。

SMTP:简单邮件传输协议(Simple Mail Transfer Protocol)
SOAP:简单对象访问协议(Simple Object Access Protocol)
SSH:安全外壳协议(Secure Shell)
SSL:安全套接层(Secure Sockets Layer)
TCP:传输控制协议(Transmission Control Protocol)
TTL:生存时间(Time To Live)
UGCCNet:泛在绿色社区控制网络(Ubiquitous Green Community Control Network)
URI:统一资源标识符(Uniform Resource Identifier)
UUID:全局唯一标识符(Universally Unique Identifier)
VPN :虚拟专用网络(Virtual Private Network)
W3C:万维网联盟(World Wide Web Consortium)
WSDL :网络服务描述语言(Web Services Description Language)
XML:可扩展置标语言(eXtensible Markup Language)

4 架构

4.1 UGCCNet 架构

本标准适用于基于 TCP/IP 协议的设施网络架构,见图 1。本标准的主要目标之一是实现设施网络组件之间的互操作性。因此,网关、存储器和应用单元,即本标准中定义的"组件",具有相同的通信接口。注册器与这些组件有不同通信接口,在图 1 中以虚线框表示。组件属于数据平面,注册器则属于控制平面。在图 1 中还强调了 AAA,提供运营管理支持。本标准并没有对 AAA 进行定义,但相关工作可能在未来必要时启动。

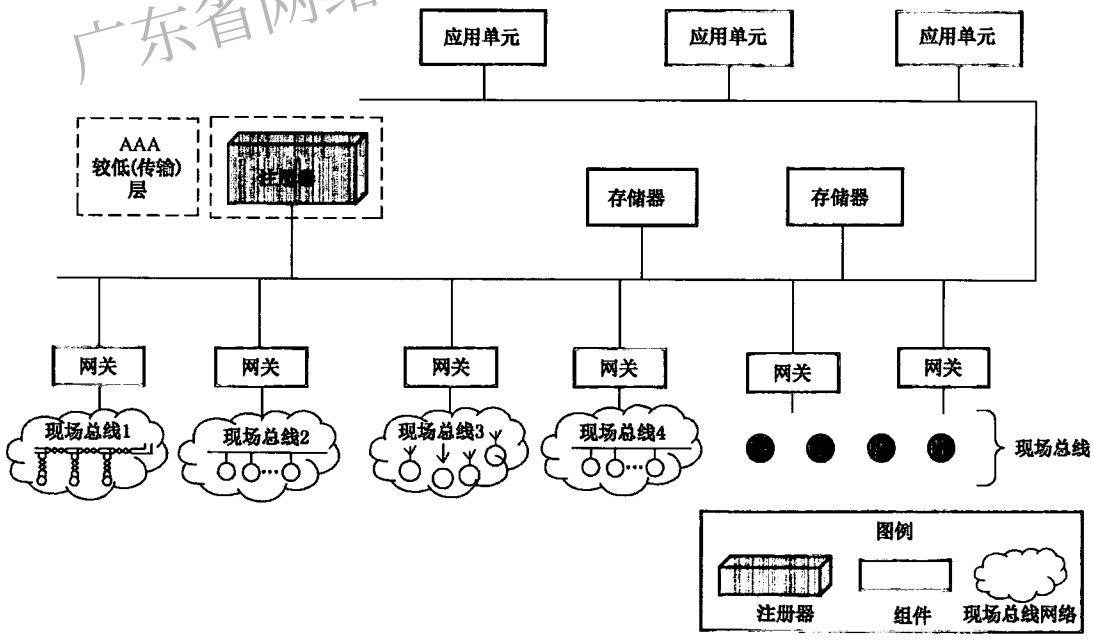


图 1 泛在绿色社区控制网络组网架构

在泛在绿色社区控制网络环境中,注册器是组件的代理,管理元信息,例如,组件充当的角色以及点标识的语义等,从而实现组件之间的自主绑定和管理。本标准详细描述了泛在绿色社区控制网络的组件之间相互协作的过程。

4.1.1 网关

网关组件连接传感器和执行器，为这些设备的数据模型和接入方法进行统一抽象，并对物理层（现场总线）的数据模型和访问方法进行封装。网关根据其他组件写入的值对执行器进行相应操作，并向其他组件（存储器和应用单元）提供传感器读取的数值。

4.1.2 存储器

存储器组件存储历史数据，其他组件写入的数值被永久保存在后端磁盘，当其他组件请求获取数据时，存储器还负责提供相应的值。

4.1.3 应用单元

应用单元组件针对传感器读取的数据和执行器的指令提供相关的应用。应用单元可以提供用户接口以显示当前环境状况，支持用户输入对执行器的设定策略，还可以作为一个虚拟设备实时分析传感器获取的数据，并提供分析结果。

4.1.4 注册器

注册器作为网关、存储器及应用单元的代理，主要负责实现组件之间主动适当的绑定。注册器不属于数据平面，不会对传感器和执行器进行直接操作。本标准允许系统在没有注册器的情况下运行。

4.2 典型通信过程

组件和注册器之间典型的通信协议过程见图 2，实线表示组件之间的通信，虚线表示组件与注册器之间的通信。

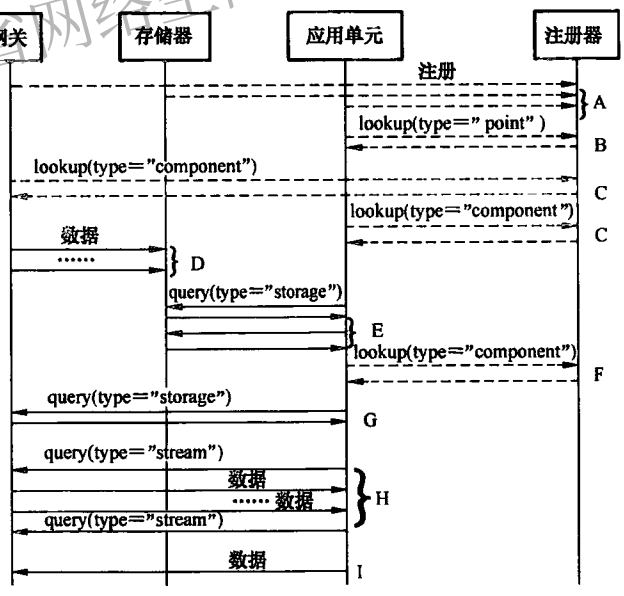


图 2 典型的泛在绿色社区控制网络通信过程

注册器管理组件的角色以及相应的点标识。当组件需要访问某个点标识对应的数据或接口时，首先查询注册器。注册器会在返回的响应中携带管理该点标识对应的数据或接口的组件的接入 URI。组件与注册器的交互是可选的，如果请求组件通过配置方式知道访问对象的 URI，就不必向注册器查询。下面给出了从过程 A 到过程 I 的通信说明。

注：每个通信过程的详细说明参见附录 A。

a) 组件的注册。注册信息包括(1)网关管理的点信息,(2)存储器负责存储的点标识对应的数据,(3)应用单元读取或提供的点数据。具体内容见 5.3.2。
b) 通过语义查询搜索相应的点。具体内容见 5.3.3。
c) 搜索某指定点对应的存储器,返回解析得到的组件的接入 URI(见 IETF RFC 3986)[7]。具体内容见 5.3.3。
d) 数据的传输。网关将传感器获取的数据发送到存储器。具体内容见 5.2.3。
e) 数据的获取。应用单元从存储器中获取数据。如果返回的数据很大,支持数据按序分段,逐段传输。具体内容见 5.2.2。
f) 搜索管理某指定点的网关组件。返回解析得到的组件的 URI。具体内容见 5.3.3。
g) 数据的获取,应用单元从网关获取数据。如果不是需要分段的超大数据,则可以在一个远程过程调用中读取完成。具体内容见 5.2.2。
h) 数据的通知。应用单元周期性的向网关设定事件查询条件,网关在满足触发条件时向应用单元主动提交更新的数据。具体内容见 5.2.4。
i) 写入数据。应用单元向网关发送对某执行器的指令。具体内容见 5.2.3。

4.3 网络设计

所有组件都可以充当一个 TCP(见 IETF RFC 793)连接的发起方和接收方,组件之间是对等的关系,可组成扁平化的网络,因此为了不对直接进行的双向通信造成影响,应避免引入 NAT 路由器或防火墙之类的中间设备。

为解决该问题,本标准强烈建议采用 IPv6 网络(见 IETF RFC 2460),除此以外还可采用基于 HTTP 代理或 NAT 穿越的解决方案,然而,这些方案通常都依赖于网络配置需求。本标准并不排斥这类解决方案,但要求他们与其他基于泛在绿色社区控制网络构建的系统可以互操作,具体的除 IPv6 之外的解决方案不在本标准讨论的范围内。

4.4 系统模型和部署

泛在绿色社区控制网络的系统模型见图 3。

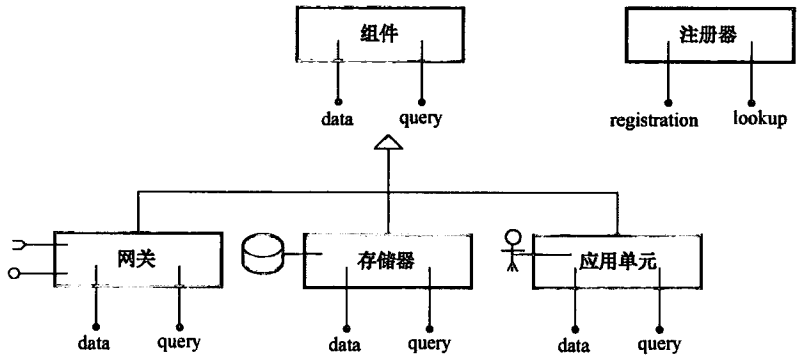


图 3 系统模型

作为泛在绿色社区控制网络的基本单元,组件是对网关、存储器、应用单元的统一抽象,组件接口提供两种方法:data[8]和 query[9]。由于网关、存储器和应用单元都是组件的继承类,因此,它们具有相同的接口(data 和 query),并使用相同的协议通信。

query 是一种从组件中获取数据(包括基于事件的数据传输)的方法;

7) 标准文本、表格和图形中的注释仅供参考,不包含实施本标准的必然要求。
8) 本标准中用"data"表示向组件写入数据的接口方法。
9) 本标准中用"query"表示从组件中获取数据(包括基于事件的数据传输)的方法。

data 是一种向组件中写入数据的方法。

注册器作为组件的代理，具有不同的接口，提供 registration[10] 和 lookup[11] 方法。

registration 是一种记录组件角色和点语义的方法；

lookup 是一种搜索特定组件或点的方法。

在网关、存储器、应用单元和注册器的实际实现中：

网关通过 query 和 data 方法实现对现场总线的封装，提供对物理设备的输入/输出访问；

存储器将通过 data 方法获取的数据进行保存，并通过 query 方法实现历史数据的查询；

应用单元实现其他功能。例如，提供用户门户，实现数据处理等；

注册器管理并维护组件与相关点标识之间的关系，通过 registration 方法提供组件角色和点标识语义间的绑定，通过 lookup 方法提供针对组件和点标识的查询。

注：本标准也允许调用 APP 系统不通过 quary 和 data 方法访问其他组件。

上述的统一抽象保证了任何厂商都可以开放的自主研发设施网络组件（即网关、存储器和应用单元）。与此同时，在客户楼宇部署设施网络系统时，无需针对用户进行额外的改动，见图 4。

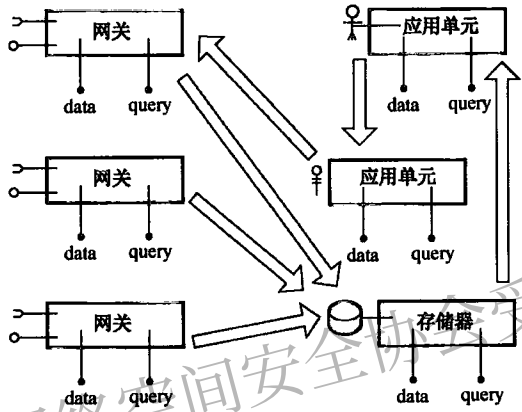


图 4　设施网络系统的实现

注册器的作用是增加组件之间协作的自治程度。在可操作范围内，注册器允许组件通过分享角色信息进行协作（事实上，不仅仅适用于可操作范围内，对于非可操作域同样适用），见图 5。

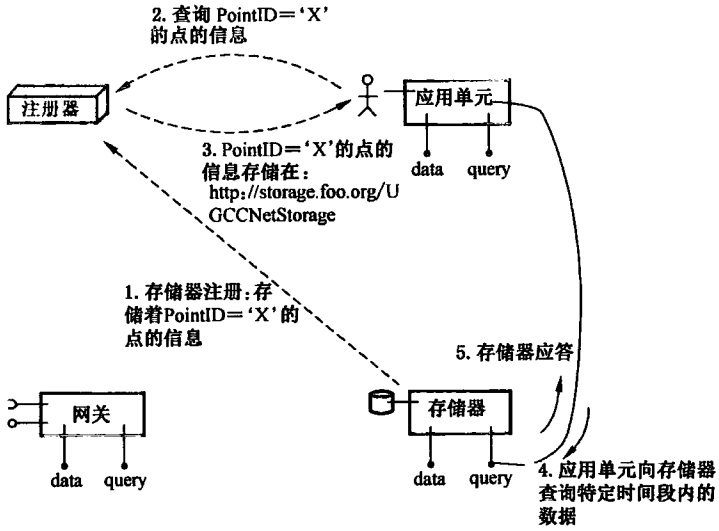


图 5　组件之间的协同

10）　本标准中用“registration”表示组件或点注册的接口方法。

11）　本标准中用“lookup”表示查询组件或点的接口方法。

4.5 点

4.5.1 概述

本条定义“点”的概念，一个点由基于 URI 的全局唯一标识符进行标识，它确定了组件之间交换数据的数据流(如传感器读数、执行器命令及控制信号等)。

4.5.2 定义

点描述了用于在组件之间传输某特定数据序列的消息通道，一系列的传感器读数，执行器命令等(如虚拟的传感器读数、元控制信号)都应与点绑定。在一个点(来自于传感器或到执行器中去)中通过数据值来表示信息，点中的数据值可以是任何的对象类型。

通过调用其他组件的接口方法，可以在组件之间传递点的数据值。提供的方法是：

query：从某指定点读取对象；

data：向某指定点写入对象。

通过使用这些方法，一个组件可以从另一个组件得到指定点的数据值，同时也可以将某点相关的数据值传输到另一个组件。

注：本标准扩展了“点”在设施网络中的传统概念。传统上，“点”代表一个可实现直接访问(通过读写方式)的特定设备。当目标组件是网关时，此定义仍然适用。然而，在本标准中，存储器和应用单元具有与网关相同的接口，因此我们扩展了其定义，不仅可以访问它们，还可以管理与它们相关的数据序列。在向一个点写入数据值时，如果组件具有网关功能，则与之相关联的物理执行器将根据写入值进行操作。如果组件具有存储器功能，则该值将被归档在其磁盘中。

4.5.3 基于 URI 的标识

一个点对应于一个全球唯一标识的数据序列。数据可能是来自某特定传感器的读取数据，或是发往某特定执行器的指令数据。因此，为了区分全球的数据序列，每个点都应该有一个全球唯一的标识符。当然对于局部范围的私有应用，无需全局唯一，但本标准并不推荐使用该方式。

在泛在绿色社区控制网络中，每个点都需要一个 URI 作为标识符。在实际应用中，可以首先给物理传感器和执行器分配标识符，然后基于该标识符生成点标识，这种操作方式可以在传统的设施网络运营中很好的应用。

通过使用 URI 作为标识符，可以实现该点的全局访问，如 X(=http://gw.foo.org/sensor1) 可作为一个点标识。

如果组件不知道管理点标识 X 的注册服务器的地址，组件应尝试直接访问 X，然后，URI 能够重定向到注册服务器。如果组件已经获知管理 X 的注册服务器，点标识则不需要是可达的。然而，为了操作方便，提高可读性，URI 的主机部分应该是网关的主机名称(因为物理传感器和执行器连接到网关)。例如典型的 URI 格式是：point ID = http://〈GW host name〉/〈any format to identify the Point in the GW〉。

4.5.4 点集合

本标准定义了点集合，点集合支持点的分级管理。一个点集合聚集多个点和多个点集合，这个定义允许对点进行分组分层的常规操作。点集合功能是可选功能，所有的组件应允许在没有点集合的情况下进行操作。点集合的具体定义见 8.3.2.5。

5 通信协议

5.1 综述

本标准针对组件和注册器定义了两种类型的通信协议，包括组件与组件之间的通信协议和组件与注册器之间的通信协议，两种通信协议基于 SOAP 协议来实现（见 W3C，SOAP 协议 1.2 版本第 1 部分：消息传递框架）。

5.2 组件与组件之间的通信协议

5.2.1 组件与组件之间通信协议的类型

本条定义和描述了组件之间通信的 3 种子协议类型，组件指的是网关、存储器和应用单元。对于注册器的接入方法参见 5.3。

FETCH[12] 协议——用于从远程组件获取数据。

WRITE[13] 协议——用于向远程组件传输数据。

TRAP[14] 协议——用于基于事件查询的注册和基于事件的数据传输。

下面给出了这 3 种协议的详细定义和描述。

5.2.2 FETCH 协议

FETCH 协议用于从远程组件获取数据。本标准将请求获取数据的组件称为“信息请求方”，将负责应答提供数据的组件称为“信息提供方”。FETCH 协议的示意图见图 6。

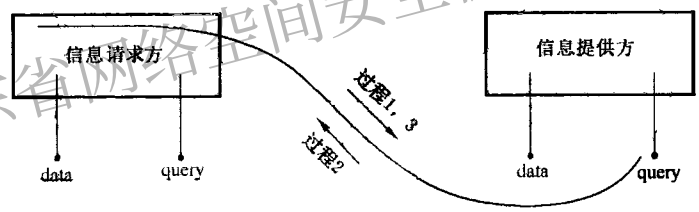


图 6 FETCH 协议示意图

过程 1：信息请求方调用信息提供方的 query 方法。信息请求方发送查询请求信息（例如，限定感兴趣的数据集的范围），并指明在远程过程调用响应时可接受的最大数据集大小，即可同时接收数据值的最大数目，默认值为 100。

过程 2：信息提供方通过远程过程调用响应返回数据集。如果返回数据集的大小超过了信息请求方最大可接受的数据的大小，或是需要消耗信息提供方过多的计算资源，则信息提供方仅返回整个数据集的一个子集，并提供一个指针用于后续的数据获取过程。

过程 3：如果在接收到的响应中有一个指针，则信息请求方再次调用信息提供方的 query 方法，重复过程 2。

过程 4：如果接收到的响应中没有指针，表示所有的数据已经传递完毕，FETCH 过程也完成了。

注 1：指针应具有有效时间。如果信息提供方收到的请求消息中的指针已经过期失效，则返回报错信息。指针有效期推荐值为 60s。但是如果网络带宽不够，可能导致 RPC 耗时变长，因此需要适当增加指针的有效期时间。

注 2：信息提供方遇到任何错误时都应主动返回错误信息，如访问控制策略错误，XML 的格式不正确的，或其他系统错误。

12) 本标准中用“FETCH”表示数据获取协议的名称。

13) 本标准中用“WRITE”表示数据推送协议的名称。

14) 本标准中用“TRAP”表示基于事件触发协议的名称。

5.2.3 WRITE 协议

WRITE 协议是用于向远程组件传输数据。提交数据的组件为“信息提交方”，接收数据的组件为“信息接收方”。WRITE 协议示意图见图 7。

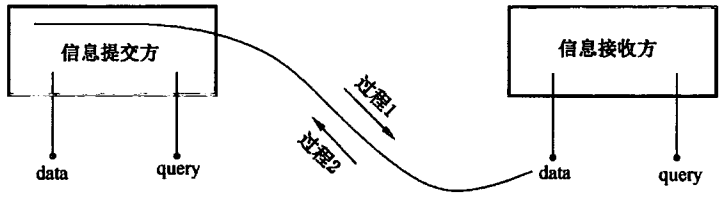


图 7 WRITE 协议示意图

过程 1：信息提交方调用信息接收方的 data 方法，携带待发送数据的内容。

过程 2：信息接收方向信息提交方返回结果，表明此次 WRITE 操作是成功或失败。

5.2.4 TRAP 协议

TRAP 支持基于事件的查询注册和数据传输。组件命名如下：

信息请求方——向信息提供方设置基于事件的查询条件的组件；

信息提供方——负责当收到与触发查询条件匹配的更新时，发送数据的组件；

数据信息回送方——从信息提供方接收数据的组件；

控制信息回送方——从信息提供方接收控制信息的组件。

本条描述了组件之间的协作（见图 8）。尽管组件的角色一般都有明确的分类，但在大多数实际应用中，数据信息回送方、控制信息回送方和信息请求方通常是同一个组件（见图 9）。

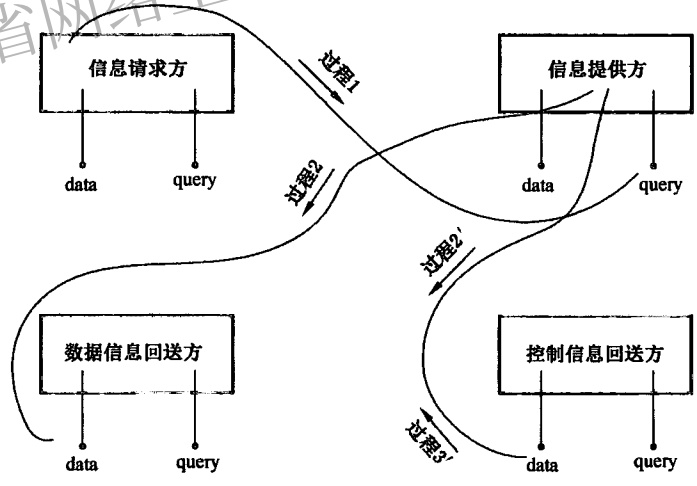


图 8 TRAP 协议（通用部署）

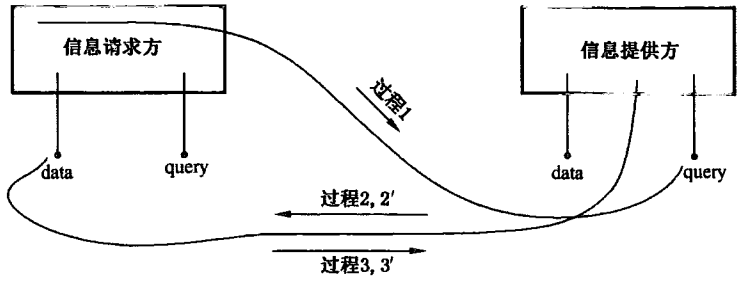


图 9 TRAP 协议（实际部署）

过程 1:信息请求方调用信息提供方的 query 方法。除了查询表达式之外,同时传输的参数还包括查询的有效时间(TTL),以秒(s)为单位,数据信息回送方组件的 URI 和控制信息回送方组件的 URI。

注 1:在该过程中,需要周期性的更新触发该过程,以更新信息提供方记录的 TTL,从而使信息提供方能够继续接收该通知消息,更新频率推荐为 TTL 的一半或三分之一。为了使信息提供方可以区分接收到的是更新请求还是新的设置请求,需要为更新的触发请求设置相同的标识。

注 2:如果信息提供方无法接受该请求,则返回错误消息。

过程 2:信息提供方调用数据信息回送方的 data 方法,推送与查询表达式匹配的更新数据,同时信息提供方发出请求。

过程 3:数据信息回送方回送成功或错误信息。

过程 2′:信息提供方调用控制信息回送方的 data 方法,推送通信错误信息(例如,过程 3 中遇到的错误)。

过程 3′:控制信息回送方回送成功或错误信息。

注 1:在 TRAP 过程中,每过 1s,信息提供方就将查询表达式的 TTL 减 1。如果 TTL 达到"0",信息提供方拒绝该查询,并将其从查询表项中删除。

注 2:如果信息请求方想显示的删除某查询请求,也可以通过指定 TTL=0 的方法实现。

5.3 组件与注册器的通信协议

5.3.1 组件与注册器的通信协议类型

本条定义了组件和注册器间通信的两种类型的子协议:

REGISTRATION[15] 协议——组件角色和点对象语义信息的注册。

LOOKUP[16] 协议——查找相应的组件和点对象。

注:注册器的接口提供两种方法,registration 和 lookup,见 6.4。REGISTRATION 协议调用 registration 方法,LOOKUP 协议调用 lookup 方法。

下面给出了这两种协议的详细描述。

5.3.2 REGISTRATION 协议

REGISTRATION 协议使组件注册自己的角色以及点对象的语义信息,提交注册申请的组件为"注册发起方",示意图见图 10。

图 10 REGISTRATION 示意图

过程 1:注册发起方调用注册器的 registration 方法,提供组件信息(如:名称,接入 URI,支持的协议类型)和角色信息(如:组件管理的点对象信息)。

注:本标准不要求注册发起方支持 data 和 query 方法,例如对于点属性的注册过程,注册的是点语义信息,并非是一个组件的角色信息。见 6.4.2。

过程 2:注册器返回成功或失败结果。如果未成功,注册器应返回错误信息给注册发起方。

---

15) 本标准中用"REGISTRATION"表示组件或点注册协议的名称。

16) 本标准中用"LOOKUP"表示组件或点查询协议的名称。

5.3.3 LOOKUP 协议

通过 LOOKUP 协议，组件可以查询需要接入的组件(对于组件之间的通信)，或是通过语义查询寻找指定的点。发起查询请求的组件为“查找请求方”，示意图见图 11。

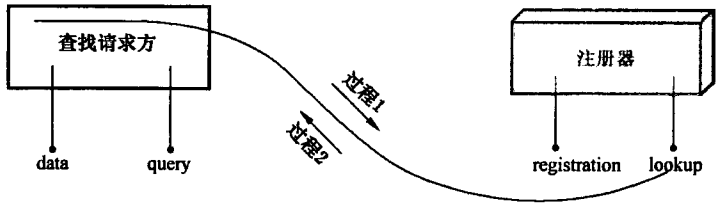


图 11 LOOKUP 示意图

过程 1：查找请求方调用注册器的 lookup 方法，其中的查找表达式指明查询对象的类型(如是查询组件还是查询点信息)。

过程 2：注册器返回解析的组件或点的接入 URI。

6 应用程序编程接口

6.1 综述

本条定义了两种类型的应用程序编程接口(API)。

组件访问接口——用于组件到组件的通信过程。

注册器访问接口——用于组件和注册器之间的通信过程。

6.2 传输的数据结构

本标准中无论是组件之间的通信还是组件与注册器之间的通信，都通过远程过程调用 RPC 实现接入方法，见图 12。调用方调用被调用方的接口方法，发起请求消息，而被调用方返回响应消息。请求消息和响应消息具有相同的数据结构，包括消息头和消息体。消息头包含控制信息，如用于组件之间通信的查询表达式，用于组件和注册器之间通信的查找表达式，确认信息，失败信息等。消息体包含携带数值的点或点集合对象，例如来自传感器的读数和对执行器的指令(用于组件之间的通信)，或是组件角色和对应点标识(用于组件和注册器之间的通信)。点或特定数据值相关的控制信息也是消息体的一部分。

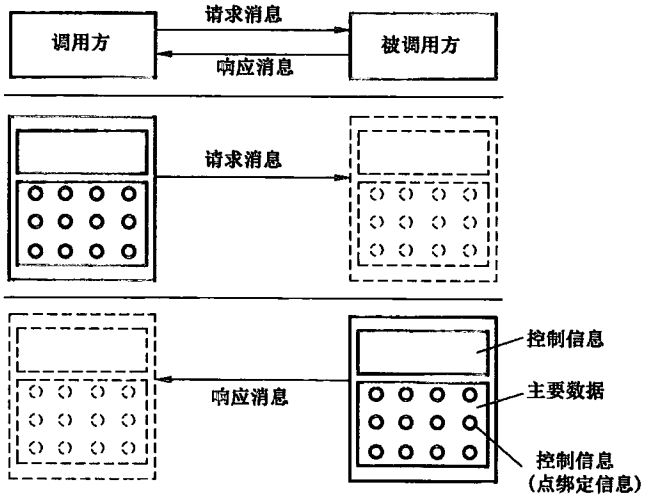


图 12 传输消息的数据结构

6.3 组件的访问接口

6.3.1 方法类型

组件(网关、存储器和应用单元)调用此接口,接口定义了 query 和 data 方法,用于其他组件的访问。接口方法的定义以 WSDL 形式提供,具体内容参见附录 D。

query 方法:用于从组件获取或订阅数据;

data 方法:用于向组件推送数据。

在 5.2 中定义的如下三种子协议通过 query 和 data 两种方法实现其功能:

FETCH 协议;

WRTIE 协议;

TRAP 协议。

协议的实际应用内容参见附录 C。

注:所有组件都应支持这些方法,然而,一些组件(特别是 APP)并不总是需要支持所有功能(例如,一些组件仅支持 FETCH 信息请求方的功能)。

6.3.2 query 方法

6.3.2.1 简介

6.3.2.1.1 格式

"Transport query(Transport t)"。

6.3.2.1.2 介绍

信息请求方通过 query 方法,指定查询表达式,从信息提供方获取数据。查询表达式应包含在 t 的消息头中。

本标准定义了两种类型请求:类型为"storage"和"stream"。根据请求的类型,信息提供方应通过以下方式工作。

类型为"storage"时,表明应用 FETCH 协议。

类型为"stream"时,表明应用 TRAP 协议。

下面给出了上述情况下 query 方法的详细描述。

6.3.2.2 query(类型为"storage")

6.3.2.2.1 请求消息

查询表达式应放在 t 的头部,而 t 的消息体部分应被忽略。

6.3.2.2.1.1 查询表达式的属性

FETCH 协议的查询表达式具有如下属性:

id:用于标识该请求;

type:设置为"storage";

acceptableSize:信息请求方在一个远程过程调用响应中,一次所能接收的数据值元素的最大数目;

cursor:用于获取未传完的后续数据,cursor 属性不会出现在信息请求方发送的第一个请求消息中,其值由信息提供方根据响应消息的发送情况设定。

6.3.2.2.2 响应消息

6.3.2.2.2.1 消息头

在响应的消息头部分应携带有查询表达式和“OK”或“Error”。如果查询表达式有指针属性，就意味着有后续的提供方的数据集。如果没有，这说明已经从提供者获取了所有数据。

OK 对象表示在提供者的程序已顺利完成。Error 对象表示同时开展的程序发生了错误，错误信息应包括在 Error 对象中。

6.3.2.2.2.2 消息体

如果消息传输成功(消息头中包含 OK 对象)，则点集合或点的对象应包含在消息体部分。如果未成功，消息体对象应被忽略。

6.3.2.3 query(类型为“stream”)

6.3.2.3.1 请求消息

查询表达式应放在 t 的头部，t 的消息体部分应被忽略。

6.3.2.3.1.1 查询表达式的属性

TRAP 协议的查询表达式具有如下属性：

id：用于标识该查询(UUID 格式)；

type：设置为“stream”；

ttl：设置该请求在信息提供方处的生存时间，单位为秒(s)；

callbackData：当数据与查询匹配时，数据发送的目标组件的 URI；

callbackControl：当需要发送控制类信息时，接收该类信息的目标组件的 URI；

acceptableSize：可选，callbackData 标识的组件在一次数据传输过程中，一次所能接收的数据值元素的最大数目。

6.3.2.3.2 响应消息

6.3.2.3.2.1 消息头

在响应的消息头部分，包含查询表达式和“OK”或“Error”对象。

OK 对象表示提供者侧的流程已成功完成，Error 对象表示执行过程中发生了错误，错误信息应包含在 Error 对象里。

6.3.2.3.2.2 消息体

消息体对象应被忽略。

6.3.2.3.3 其他

信息提供方调用 callbackData 所标识组件的 data 方法，从而传输与查询匹配的数据。信息提供方查询表达式添加到参数 t 的头部，以便 callbackData 可以识别数据传输的上下文。

注：当信息请求方更新信息提供方的查询 TTL 时，信息请求方应使用相同的查询标识(TRAP 协议)。

6.3.3 data 方法

6.3.3.1 格式

Transport data(Transport t)。

6.3.3.2 介绍

信息提交方将向信息接收方传送的点或点集合的相关信息封装在 data 请求消息 t 的消息体中。如果在 TRAP 协议中调用 data 方法，从信息提交方向数据信息回送方组件传送数据，则 t 的消息头中复制有相应的 query 方法消息头中携带的查询表达式。

6.3.3.3 请求消息

数据对象应放在消息体部分。在消息头部分，TRAP 协议中应包含相应的查询表达式和控制信号。

6.3.3.4 响应消息

6.3.3.4.1 消息头

在响应的消息头部分应包含“OK”或“Error”对象。

OK 对象表示提供者侧的流程已成功完成，Error 对象表示执行过程中发生了错误，错误信息应包含在 Error 对象里。

6.3.3.4.2 消息体

消息体部分应被忽略。

6.4 注册器的访问接口

6.4.1 方法类型

注册器应支持此访问接口，访问接口定义了 registration 和 lookup 方法，从而支持组件（网关、存储器和应用单元）访问注册器。此接口功能对于组件是可选的。WSDL 格式的注册器访问接口的定义参见附录 E。

registration：注册组件角色和点对象语义；

lookup：查找相应的接入组件，通过语义查询查找点对象。

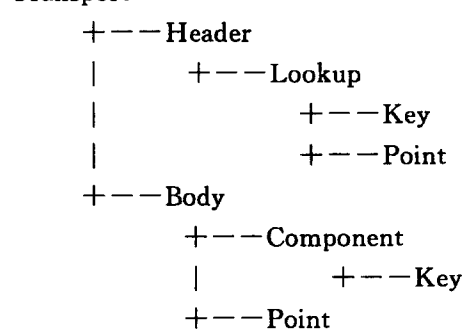
在 5.3 中定义的如下两种子协议通过 registration 和 lookup 两种方法实现其功能：

REGISTRATION 协议；

LOOKUP 协议。

6.4.1.1 注册器访问接口的数据结构

对于注册器支持的协议 REGISTRATION 和 LOOKUP 而言，具有相同的数据结构，都包括消息头 Header 和消息体 Body。其中消息头中包含有查找表达式，以及成功或失败的状态信息。消息体部分则描述了组件的实现角色，以及对应的点对象信息，见 8.4。



6.4.2 registration 方法

6.4.2.1 简介

6.4.2.1.1 格式

Transport registration(Transport t)。

6.4.2.1.2 介绍

当注册发起方调用注册器的 registration 方法时，完成自身的实现角色以及对应的点信息的注册。基于该方法可以实现 REGISTRATION 协议(见 5.3.2)。在本标准中，注册器应管理两种类型的注册信息：

描述组件角色的信息；

描述点对象的信息。

6.4.2.2 组件的注册过程

6.4.2.2.1 请求消息

当处理组件相关的注册信息时，请求消息 t 的消息体中携带关于组件的信息。

6.4.2.2.2 响应消息

####### 6.4.2.2.2.1 消息头

在响应的消息头部分应包含“OK”或“Error”对象。

OK 对象表示注册器侧的流程已成功完成，Error 对象表示执行过程中发生了错误，错误信息应包在 Error 对象内。

####### 6.4.2.2.2.2 消息体

响应消息的消息体应被忽略。

6.4.2.3 点的注册

点对象语义相关的信息应携带于 t 的消息体中。

在本标准中，对于点对象，除了点标识之外，不指定其他的任何属性。系统操作员可根据不同的应用背景，设计适当的点配置文件，并提交配置文件到注册表访问接口，使系统的部署和操作更灵活。

6.4.3 lookup 方法

6.4.3.1 简介

6.4.3.1.1 格式

Transport lookup (Transport t)。

6.4.3.1.2 概述

lookup 方法用于组件查找某特定点对象的归属组件，或者查找符合一定约束条件的匹配点对象。

基于该方法可以实现 LOOKUP 协议。当信息请求方通过调用注册器的 lookup 方法时，在消息实例 t 的消息头中指明查找表达式。查找表达式的一个重要属性是查找的对象，用“type”表示，有两种不同的取值，分别为“component”和“point”。根据不同的类型，注册器会采用不同的处理方法。

当类型 type=“component”时，找到匹配的一个或多个组件来访问。注册器返回解析得到的一个或多个组件的接入 URI；

当类型 type=“point”时，用于完成一个或多个点的语义查询。注册器返回一个或多个匹配点的信息，包括配置文件等。

6.4.3.2 lookup(类型为“component”)

6.4.3.2.1 请求消息

lookup 请求消息 t 的消息头部分携带查找表达式，t 的消息体部分应被忽略。

6.4.3.2.1.1 查找表达式的属性

查找表达式包括如下属性：

id：用于标识该查询；

type：设置为“component”。

6.4.3.2.2 响应消息

6.4.3.2.2.1 消息头

在响应的消息头部分，应包含查找表达式和“OK”或“Error”对象。

OK 表明注册器成功的完成了 lookup 方法对应的组件查找过程，Error 表示注册器在执行 lookup 方法调用时出现了错误，Error 也可提供具体的错误描述信息，供信息请求方检查。

6.4.3.2.2.2 消息体

如果 lookup 响应携带 OK，则其消息体中携带解析得到的接入组件的 URI，否则该消息体被接收方忽略。

6.4.3.3 lookup(类型为“point”)

6.4.3.3.1 请求消息

请求消息 t 的消息头部分携带有查找表达式，t 的消息体部分应被忽略。

6.4.3.3.1.1 查找表达式的属性

查找表达式包括如下属性：

id：用于标识该查找(UUID 格式)；

type：设置为“point”。

6.4.3.3.2 响应消息

6.4.3.3.2.1 消息头

在响应的消息头里应包含查找表达式和“OK”或“Error”对象。

OK 对象表示在注册器中的程序已顺利完成，Error 对象表示执行过程中发生了错误，错误信息应包括在 Error 对象里。

6.4.3.3.2.2 消息体

如果它是成功的(如果一个 OK 对象包含在消息头)，则其消息体中携带与查找匹配的点对象的属性文件，否则该消息体被接收方忽略。

7 data 和 query 模型

7.1 总则

本部分定义组件到组件通信的 data 和 query 模型，这个数据模型基本上采用传统的树状数据结构。

7.2 点集合树结构的点管理

一个典型点集合数据树结构见图 13。一个点集合聚集多个点和点集合，这种结构实现了分级管理。每个点通过一系列的数据值元素来描述，传感器的一个读数或执行器的一条指令都是一个数据值元素。数据值元素本身不需要全局唯一的标识符，但具有时间属性，从而标识该数据值何时从传感器提取，或者应在何时将该数据值发往执行器上。

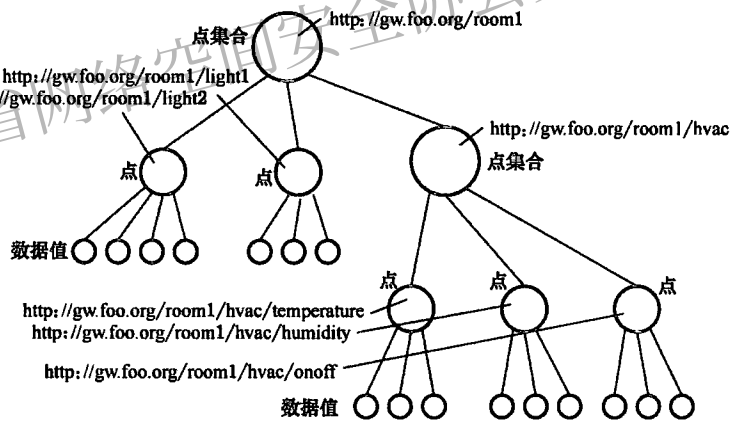


图 13 点集合树数据模型

本标准定义了点集合的概念，但点集合的功能是可选的。所有的组件应允许在没有点集合下进行操作。

在本标准中，仅对点和点集合定义了 id 属性，对数据值定义了时间属性，其他的属性可以根据具体的节点应用和现场网络进行灵活的定义和扩展。

7.3 点集合树结构的查询模式

FETCH 和 TRAP 的请求方通过 query 和 key 元素指定了数据集的范围，然后信息提供方应提供相应的点集合，点和数据值对象。信息提供方将指定的点集合和点对象作为响应消息 body 元素的子集，点集合和点对象在点集合树上不应有两个以上的子节点。

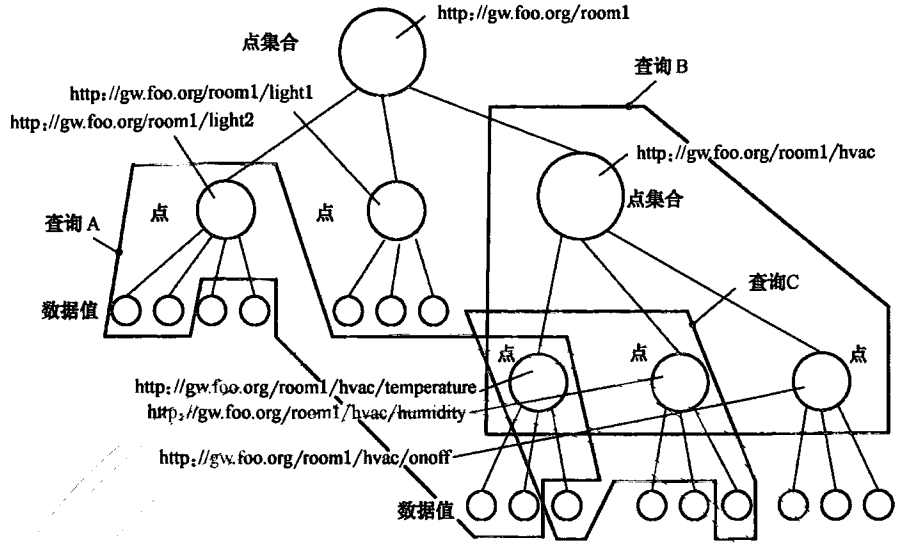


图 14 基于树状结构的 query 接口查询

7.3.1 查询 A

图 14 提供了点集合树的简单请求模型，以下描述了针对指定数据集范围的查询。

描述：查询标识为 http://gw.foo.org/room1/light2 或 http://gw.foo.org/room1/hvac/temperature 在 2009 年 10 月 1 日到 2009 年 10 月 2 日之间的值。

```
〈query〉
  〈key id="http://gw.foo.org/room1/light2" attrName="time"
  gteq="2009-10-01T00:00:00.00000000+08:00"
  lteq="2009-10-02T00:00:00.00000000+08:00" /〉
  〈key id="http://gw.foo.org/room1/hvac/temperature" attrName="time"
  gteq="2009-10-01T00:00:00.00000000+08:00"
  lteq="2009-10-02T00:00:00.00000000+08:00" /〉
〈/query〉
```

7.3.2 查询 B

描述：查询 http://gw.foo.org/room1/hvac 点的信息。

```
〈query〉
  〈key id="http://gw.foo.org/room1/hvac" /〉
〈/query〉
```

7.3.3 查询 C

描述：获取 http://gw.foo.org/room1/hvac/temperature 及 http://gw.foo.org/room1/hvac/humidity 的当前值。

```
〈query〉
  〈key id="http://gw.foo.org/room1/hvac/temperature" attrName="time"
  select="maximum" /〉
  〈key id="http://gw.foo.org/room1/hvac/humidity" attrName="time"
select="maximum" /〉
```

8 数据结构

8.1 总则

本章针对组件之间，以及组件与注册器之间的通信协议，定义了相应的数据结构。所有的消息遵从XML格式(见W3C可扩展置标语言)。本标准中一个类对应一个XML元素，下面章条中类的定义给出了XML构架(见W3C XML概要　第1部分：结构，W3C XML概要　第2部分：数据类型)，在这种数据格式下XML的命名空间(见W3C XML的命名空间1.1版本)为xmlns＝http://gutp.jp/fiap/2009/11/。

8.2 对象类和XML元素的命名规则

命名有大小写之分，本标准规定，类的名称以大写字母开始，XML元素的名称以小写字母开始。从第二个字母开始，类的名称和XML的名称应该是完全一样的。

8.3 组件之间通信协议的数据结构

8.3.1 介绍

本条定义了组件之间通信的数据结构，此数据结构将在以下协议(5.2中定义)中以请求及响应交互方式传输：

——FETCH协议；
——WRITE协议；
——TRAP协议。

8.3.2 类列表

8.3.2.1 总则

针对UGCCNet中组件之间的通信协议，定义了如下10种类。

——Tranport；
——Header；
——Body；
——PointSet；
——Point；
——Value；
——Query；
——Key；
——OK；
——Error。

8.3.2.2 Transport类

8.3.2.2.1 类描述

Transport类用于在一个消息中同时传输数据平面信息和控制平面信息递。

8.3.2.2.2 类成员

Transport类有一个Header对象和Body对象。它们不包含任何数据时可能被忽略。

8.3.2.2.3 类属性

无。

8.3.2.2.4 类示例

```
〈transport〉
  〈header〉……〈/header〉
  〈body〉……〈/body〉
〈/transport〉
```

8.3.2.3 Header 类

8.3.2.3.1 类描述

Header 类用于承载控制平面信息，包括查询表达式、失败消息等。

8.3.2.3.2 类成员

Header 类有 Query 对象、OK 对象和 Error 对象。

8.3.2.3.3 类属性

无。

8.3.2.3.4 类示例 1

```
〈header〉
  〈query…〉…〈/query〉
〈/header〉
```

8.3.2.3.5 类示例 2

```
〈header〉
  〈OK/〉
〈/header〉
```

8.3.2.4 Body 类

8.3.2.4.1 类描述

Body 类用于承载点对象或点集合对象的相关数据信息。

注：与点对象或数据值密切相关的控制信息也应被当做数据处理并以点对象或数据值来表示。

8.3.2.4.2 类成员

Body 类有 PointSet 对象和 Point 对象。

8.3.2.4.3 类属性

无。

8.3.2.4.4 类示例 1

```
〈body〉
```

```
  〈pointSet id="http://gw.foo.org/tv1"〉...〈/pointSet〉
  〈pointSet id="http://gw.foo.org/refrigerator1"〉...〈/pointSet〉
  〈pointSet id="http://gw.foo.org/pot1"〉...〈/pointSet〉
〈/body〉
```

8.3.2.4.5 类示例 2

```
〈body〉
  〈point id="http://gw.foo.org/tv1/power"〉...〈/point〉
  〈point id="http://gw.foo.org/tv1/switch"〉...〈/point〉
  〈point id="http://gw.foo.org/pot1/power"〉...〈/point〉
〈/body〉
```

8.3.2.4.6 类示例 3

```
〈body〉
  〈pointSet id="http://gw.foo.org/room1/light"〉...〈/pointSet〉
  〈pointSet id="http://gw.foo.org/room1/hvac"〉...〈/pointSet〉
  〈point id="http://gw.foo.org/tv1/power"〉...〈/point〉
  〈point id="http://gw.foo.org/pot1/power"〉...〈/point〉
〈/body〉
```

8.3.2.5 PointSet 类

8.3.2.5.1 类描述

PointSet 类聚合相关的 Point 对象和 PointSet 对象。

8.3.2.5.2 类成员

PointSet 类包括 PointSet 对象和 Point 对象。

8.3.2.5.3 类属性

类属性为 id，是基于 URI 的标识符，用来唯一标识 PointSet 对象。

8.3.2.5.4 类示例 1

```
〈pointSetid="http://gw.foo.org/room1"〉
  〈pointSetid="http://gw.foo.org/room1/light"〉...〈/pointSet〉
  〈pointSetid="http://gw.foo.org/room1/hvac"〉...〈/pointSet〉
〈/pointSet〉
```

8.3.2.5.5 类示例 2

```
〈pointSetid="http://gw.foo.org/tv1"〉
  〈pointid="http://gw.foo.org/tv1/power"〉...〈/point〉
  〈pointid="http://gw.foo.org/tv1/switch"〉...〈/point〉
  〈pointid="http://gw.foo.org/tv1/channel"〉...〈/point〉
〈/pointSet〉
```

8.3.2.5.6 类示例 3

```
〈pointSetid="http://gw.foo.org/room1〉
  〈pointSetid="http://gw.foo.org/room1/light"〉...〈/pointSet〉
  〈pointSetid="http://gw.foo.org/room1/hvac"〉...〈/pointSet〉
  〈pointid="http://gw.foo.org/room1/temperature"〉...〈/point〉
  〈pointid="http://gw.foo.org/room1/humidity"〉...〈/point〉
〈/pointSet〉
```

8.3.2.6 Point 类

8.3.2.6.1 类描述

Point 类用以描述点对象。

8.3.2.6.2 类成员

Point 类成员包括 Value 对象。

8.3.2.6.3 类属性

Point 类的属性为 id,用来唯一标识 Point 对象。

8.3.2.6.4 类示例

```
〈pointid="http://gw.foo.org/room1/temperature"〉
  〈valuetime="2009-09-01T00:00:00.0000000+08:00"〉25.5〈/value〉
  〈valuetime="2009-09-01T00:01:00.0000000+08:00"〉25.6〈/value〉
  〈valuetime="2009-09-01T00:02:00.0000000+08:00"〉25.6〈/value〉
  〈value time="2009-09-01T00:03:00.0000000+08:00"〉25.7〈/value〉
〈/point〉
```

8.3.2.7 Value 类

8.3.2.7.1 类描述

Value 类包括了具体的数据值,来自传感器的输入数据或是发往执行器的指令数据都可以封装在 Value 对象中。

8.3.2.7.2 类成员

无。

8.3.2.7.3 类属性

time:按照 ISO 8601 定义的格式,至少指定秒和时区,记录输入数据的产生时间或指令数据的执行时间。

8.3.2.7.4 类示例

```
〈value time="2009-10-19T00:00:00+08:00"〉true〈/value〉
〈value time="2009-10-19T00:00:00+08:00"〉false〈/value〉
```

```
〈value time="2009-10-19T00:00:00+08:00"〉10〈/value〉
〈value time="2009-10-19T00:00:00+08:00"〉0〈/value〉
〈value time="2009-10-19T00:00:00+08:00"〉3.4〈/value〉
〈value time="2009-10-19T00:00:00+08:00"〉0.5323〈/value〉
〈value time="2009-10-19T00:00:00+08:00"〉HIGH〈/value〉
〈value time="2009-10-19T00:00:00+08:00"〉MID〈/value〉
〈value time="2009-10-19T00:00:00+08:00"〉LOW〈/value〉
```

8.3.2.8 Query 类

8.3.2.8.1 类描述

Query 类针对类型为“storage”以及“stream”的 query 方法管理相应的查询表达式。

8.3.2.8.2 类成员

Query 类的类成员为 Key。

8.3.2.8.3 类属性

Query 对象的属性包括：

——id：用于标识该请求(UUID)；

——type：标识 query 方法的类型，取值为“storage”或“stream”；

——cursor：用于接收连续数据集时的指针(当类型为“storage”时有效)；在一次远程过程调用过程中，接收方能够接收的值对象的最大长度；

——ttl：设置该请求在信息提供方处的生存时间(当类型为“stream”时有效)；

——callbackData：在 TRAP 协议中，数据信息回送方的 URI(当类型为“stream”时有效)；

——callbackControl：在 TRAP 协议中，控制信息回送方的 URI(当类型为“stream”时有效)。

8.3.2.8.4 类示例

```
〈query id ="6229c37f-970d-9292-83e4-7c0e54733f8a"
        type="storage"
        acceptableSize="20"
        cursor="dab751ed-0133-4ce4-8b7d-ba5c54ce4fb5"〉
  〈key〉 ... 〈/key〉
  〈key〉 ... 〈/key〉
〈/query〉

〈query id ="9eed9de4-1c48-4b08-a41d-dac067fc1c0d"
        type="stream"
        ttl="60"
        callbackData="http://foo.org/axis/services/GUTAPI"
        callbackControl="http://foo.org/axis/services/GUTAPI" 〉
  〈key〉 ... 〈/key〉
  〈key〉 ... 〈/key〉
〈/query〉
```

8.3.2.9 Key 类

8.3.2.9.1 类描述

Key 类用于描述查询表达式。Key 对象与 7.3 中的 key 描述相一致。

8.3.2.9.2 类成员

Key 类的成员是 Key 对象。

8.3.2.9.3 类属性

Key 对象的类属性包括：

——id：对应点或点集合的目标标识符；

——attrName：定义的如下属性的名称；

——eq：若 Key 对象的属性值等于某给定值，则为真，否则为假；

——neq：若 Key 对象的属性值不等于某给定值，则为真，否则为假；

——lt：若 Key 对象的属性值小于某给定值，则为真，否则为假；

——gt：若 Key 对象的属性值大于某给定值，则为真，否则为假；

——lteq：若 Key 对象的属性值小于或等于某给定值，则为真，否则为假；

——gteq：若 Key 对象的属性值大于或等于某给定值，则为真，否则为假；

——select：具有{maximum，minimum}两个属性值；

——trap：若检测到了发生某事件，则取值为{changed}(仅在请求类型为“stream”时有效)。

注：eq、neq、lt、gt、lteq、gteq 的时间格式应按照 ISO 8601 规定的格式，且至少指定到秒和时区。

8.3.2.9.4 类示例

```
〈key id="http://gw.foo.org/room1/temperature"
 attrName="time" select="maximum /〉

〈key id="http://gw.foo.org/room1/temperature"
attrName="time"
lteq="2009-10-01T00:00:00+08:00"
gteq="2009-09-01T00:00:00+08:00" /〉

〈key id="http://gw.foo.org/room1/temperature"
attrName="time" trap="changed" /〉

〈key id="http://gw.foo.org/room1/temperature"
attrName="value" trap="changed" /〉
```

8.3.2.10 OK 类

8.3.2.10.1 类描述

OK 类表示请求已被成功的接受。

8.3.2.10.2 类成员

无。

8.3.2.10.3 类属性

无。

8.3.2.10.4 类示例

〈OK/〉

8.3.2.11 ERROR 类

8.3.2.11.1 类描述

Error 类承载错误信息。

8.3.2.11.2 类成员

无。

8.3.2.11.3 类属性

Type:错误类型。

组件之间通信的类型属性值参见附录 F。

8.3.2.11.4 类示例

〈error type="INVALID_REQUEST"〉Malformed IEEE1888/XML Error-query element should be specified in FETCH request〈/error〉

〈error type="POINT_NOT_FOUND"〉Point id="..." is not managed in this storage.〈/error〉

8.4 组件与注册器之间通信协议的数据结构

8.4.1 介绍

本条定义了组件和注册器之间通信的数据结构,此数据结构将在以下协议(5.3 中定义)中以请求及响应交互方式传输:

——REGISTRATION 协议;

——LOOKUP 协议。

8.4.2 类列表

8.4.2.1 总则

针对 UGCCNet 中组件与注册器之间的通信协议,定义如下 9 种类:

——Transport;

——Header;

——Body;

——Component;

——Point;

——Lookup;

——Key;

——OK;

——Error。

8.4.2.2 Transport 类

8.4.2.2.1 类描述

Transport 类用于在一个消息中同时传输数据平面信息和控制平面信息递。

8.4.2.2.2 类成员

Transport 类有一个 Header 对象和 Body 对象，它们不包含任何数据时可以被忽略。

8.4.2.2.3 类属性

无。

8.4.2.2.4 类示例 1

```
〈transport〉
  〈header〉 ... 〈/header〉
  〈body〉 ... 〈/body〉
〈/transport〉
```

8.4.2.2.5 类示例 2

当向注册器注册点对象的属性时，需要创建点对象的属性文件，并在向注册器提交时指明命名空间，如：

```
〈transport xmlns:s="..."〉
  〈body〉 ... 〈/body〉
〈/transport〉
```

8.4.2.3 Header 消息头类

8.4.2.3.1 类描述

Header 类用于承载控制平面信息，包括查找表达式、失败消息等。

8.4.2.3.2 类成员

Header 类有 Lookup 对象、OK 对象和 Error 对象。

8.4.2.3.3 类属性

无。

8.4.2.3.4 类示例 1

```
〈header〉
  〈lookup ... 〉 ... 〈/lookup〉
〈/header〉
```

8.4.2.3.5 类示例 2

```
〈header〉
```

〈OK/〉
〈/header〉

8.4.2.4 Body 类

8.4.2.4.1 类描述

Body 类用于承载解析得到的组件(网关、存储器、应用单元)或点对象的属性信息。

8.4.2.4.2 类成员

Body 类有 Component 对象和 Point 对象(例如,与网关相关联的点对象,由存储器管理其点标识的点对象等)。

8.4.2.4.3 类属性

无。

8.4.2.4.4 类示例 1

〈body〉
〈component...〉...〈/component〉
〈/body〉

8.4.2.4.5 类示例 2

〈body〉
〈point id="..."/〉
〈point id="..."/〉
〈point id="..."/〉
〈/body〉

8.4.2.5 Component 类

8.4.2.5.1 类描述

Component 类是对网关、存储器和应用单元的抽象。

8.4.2.5.2 类成员

类成员包括 Key 对象。

8.4.2.5.3 类属性

Component 类的属性包括:
——name:指定组件的名称(例如,网关、存储器或应用单元);
——uri:指定某组件的接入 URI;
——priority:冗余数据集的接入优先级(可选属性);
——support:组件支持的协议类型(FETCH、WRITE、TRAP);
——expires:注册的有效期,以秒标识(可选属性)。

注 1:当存在多于一个的候选组件可以接入时(例如,存在组件的备份),需提供 priority 属性,宜选择 Priority 的值大的组件优先接入。

注 2：通过 expires 属性，注册器可以掌握某注册信息的有效期，并且每过 1 s 就将该属性值减去 1。当该属性值为 0 时，注册器将拒绝该注册消息。

注 3：要删除某组件的注册，注册发起方应调用注册器的 registration 方法将 expires 设置为 0。

注 4：如果注册发起方并未提供该属性，则由注册器自行决定对应注册条目的有效期。

8.4.2.5.4 类示例 1

一个典型的网关类示例如下：

〈componentname =" myGW" uri =" http:// gw. foo. org/axis/services/FIAPBACnetWSGW" support="FETCH|TRAP"〉

〈key id="http://gw.foo.org /EngBldg2/10F/102A2/DB1"stream="in" limit="1"/〉

〈keyid="http://gw.foo.org /EngBldg2/10F/102A2/DB2"stream="in" limit="1" /〉

〈key id="http://gw.foo.org /EngBldg2/10F/102A2/RH1"stream="in" limit="1" /〉

〈key id="http://gw.foo.org /EngBldg2/10F/102A2/RH2"stream="in" limit="1" /〉

…

〈/component〉

其中 limit= “1”指的是 myGW 只有一个数据的缓存。

8.4.2.5.5 类示例 2

一个典型的存储器类描述如下：

〈component name="myStorage"

uri="http://fiap－storage.gutp.ic.i.u－tokyo.ac.jp/axis/services/FIAPStorage"

support="FETCH|WRITE"〉

〈key id="http://gw.foo.org /EngBldg2/10F/102A2/DB1"/〉

〈key id="http://gw.foo.org /EngBldg2/10F/102A2/DB2"/〉

〈key id="http://gw.foo.org /EngBldg2/10F/102A2/RH1"/〉

〈key id="http://gw.foo.org /EngBldg2/10F/102A2/RH2"/〉

…

〈/component〉

8.4.2.6 Point 类

8.4.2.6.1 类描述

Point 类描述了点对象，但不包括点对象的数据值。

8.4.2.6.2 类成员

无。

注：不同于组件之间通信协议的数据结构，这里的点对象不包含任何对象成员。

8.4.2.6.3 类属性

id：点对象的标识。

注：为实现针对点属性设计的灵活性，其他的属性(例如，类型、可读写性、物理位置等)支持自定义。

8.4.2.6.4 类示例

〈point id="X" s:type="BINARY_INPUT" s:writable="false"

s:location="Building2F221MeetingRoom1"/〉

〈point id="Y" s:type="BINARY_INPUT" s:writable="false"
s:location="Building2F221MeetingRoom1"/〉
〈point id="Z" s:type="ANALOG_INPUT" s:writable="false"
s:location="Building2F221MeetingRoom1"/〉
〈point id="W" s:type="MULTI_STATE_INPUT" s:writable="false"
s:location="Building2 F221MeetingRoom1"/〉

8.4.2.7 Lookup 类

8.4.2.7.1 类描述

Lookup 类管理查找表达式。

8.4.2.7.2 类成员

Lookup 类成员为 Key 对象和 Point 对象。

8.4.2.7.3 类属性

Lookup 类的属性包括：
——id：用于标识该查找；
——type：标识 lookup 方法的类型，取值为“component”或“point”。

8.4.2.7.4 类示例 1：查找组件

〈lookup id="6e5a0e85－b4a0－485f－be54－a758115317e1" type="component"〉
〈key id="http://gw.foo.org/EngBldg2/10F/102A2/DB1".../〉
〈/lookup〉

8.4.2.7.5 类示例 2：查找点对象

〈lookup id="3f2504e0－4f89－11d3－9a0c－0305e82c3301" type="point"〉
〈point s:type="BINARY_INPUT" s:location="Building2F221MeetingRoom1" .../〉
〈/lookup〉
其中 S 是点配置文件的目标命名空间的命名空间前缀，例如，XMLNS:S
=“http://application－dependent.org/AttributeSet/”。

8.4.2.8 Key 关键类

8.4.2.8.1 类描述

Key 类用于描述查找表达式，以及点对象的属性文件。

8.4.2.8.2 类成员

无。

8.4.2.8.3 类属性

Key 对象的类属性包括：
——id：用于标识点对象的标识符，如用“pointID”表示；
——attrName：对象搜索属性的名称；

——stream:若组件接收数据,则取值为“in”;若组件发送数据,则取值为“out”;
——limit:数据缓存区的大小;
——eq:若 Key 对象的属性值等于某给定值,则为真,否则为假;
——neq:若 Key 对象的属性值不等于某给定值,则为真,否则为假;
——lt:若 Key 对象的属性值小于某给定值,则为真,否则为假;
——gt:若 Key 对象的属性值大于某给定值,则为真,否则为假;
——lteq:若 Key 对象的属性值小于或等于某给定值,则为真,否则为假;
——gteq:若 Key 对象的属性值大于或等于某给定值,则为真,否则为假。

注:eq、neq、lt、gt、lteq、gteq 的时间格式应按照 ISO 8601 规定的格式,且至少指定到秒和时区。

8.4.2.8.4 类示例

如果一个组件保存标识符为“X”的点对象从 2010 年 1 月 1 日之后的数据,则对应的 Key 对象为:
〈key id="X" attrName="time" gteq="2010-01-01T00:00:00+08:00"/〉
如果一个组件从传感器 X 中获取数据,则对应的 key 为:
〈key id="X" attrName="time" stream="in"/〉
如果一个组件向执行器 X 发送指令,则对应的 key 是:
〈key id="X" attrName="time" stream="out"/〉
如果一个组件可以提供输入和输出数据流,则对应的 key 是:
〈key id="X" attrName="time" stream="in|out"/〉

8.4.2.9 OK 类

8.4.2.9.1 类描述

OK 类表示请求已被成功的接受。

8.4.2.9.2 类成员

无。

8.4.2.9.3 类属性

无。

8.4.2.9.4 类示例

〈OK/〉

8.4.2.10 Error 类

8.4.2.10.1 类描述

Error 类承载错误信息。

8.4.2.10.2 类成员

无。

8.4.2.10.3 类属性

type:错误类型

Error 类的属性为“type”，用于描述错误类别等具体的错误信息，组件和注册器之间通信的 type 属性值参见附录 G。

8.4.2.10.4 类示例

〈error type="INVALID_REQUEST"〉Malformed IEEE1888/XML Error-lookup element should be specified.〈/error〉
〈error type="SERVER_ERROR"〉The server encountered an out of memory error.〈/error〉

9 协议绑定

泛在绿色社区控制网络协议支持几乎所有的现有通信协议实现数据传输，如 SMTP、SIP、FTP 和 HTTP 等，然而考虑到应用和实现的普遍性，本标准推荐采用 SOAP、HTTP 和 SIP 作为实现 UGCCNet 通信的底层绑定协议，尤其是应用 SIP 协议实现网络设施的操作和管理[17]。

10 安全考虑

本标准定义的协议是开放的，支持来自多领域的操作及其他领域系统组件的接入访问。在这种情况下，系统的安全考虑如下：

——避免向公众意外泄露数据；
——避免未经授权的访问资源；
——远程通信主机的可用性和保密性；
——数据的完整性和保密性；
——避免意外的访问或操作冲突。

我们将使用 VPN、SSL、SSH 和其他相关技术来保证远程通信主机的机密性，应用 HTTPS 或 SIP 及安全扩展协议将有助于保证数据的完整性和保密性。

访问控制和访问冲突管理应是另一个重要但不同类型的安全问题，应独立讨论。通常，访问控制用于仅允许特定用户访问可读和可写资源，这必将有助于避免来自公众用户（有时是匿名）的未授权访问或数据意外泄露。为了管理这一点，系统需要引入用户的概念来识别谁正在访问资源。我们采用基于 URI 的标识用于用户认证，正如用 URI 来标识点 Point ID 一样，用户和组件的认证（可能通过利用现有的认证平台）也需要考虑。

17) 基于 SIP 协议的 UGCCNet 标准是 IEEE P1888 工作组的后续工作。

附 录 A
（资料性附录）
UGCCNet 通信的典型序列

UGCCNet 通信的典型序列见图 A.1。

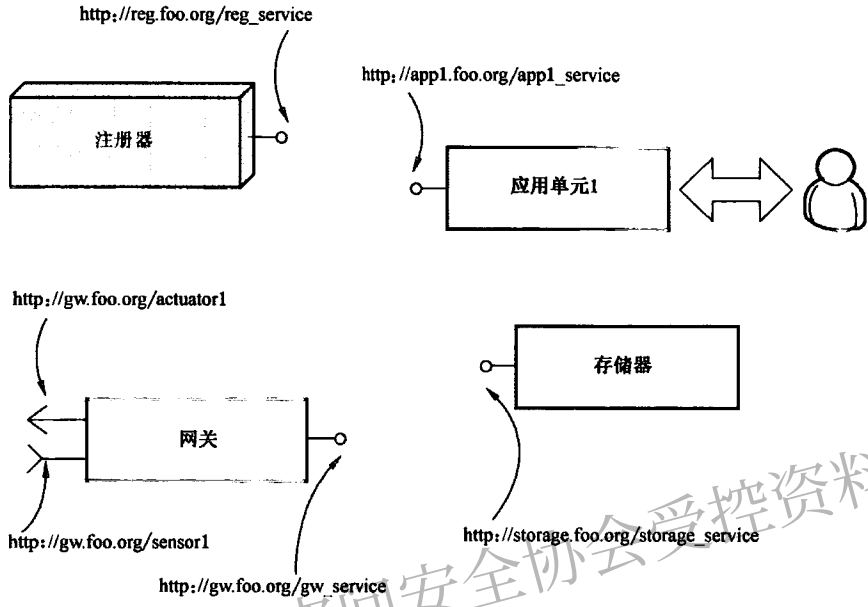


图 A.1 配置详情

过程 A:组件的注册：

http://reg.foo.org/reg_servic,e 〈-(request) - registration("Description of GW")

http://reg.foo.org/reg_service -(response)-〉"〈OK /〉"

http://reg.foo.org/reg_service 〈-(request) - registration("Description of Storage")

http://reg.foo.org/reg_service -(response)-〉"〈OK /〉"

http://reg.foo.org/reg_service 〈-(request) - registration("Description of APP")

http://reg.foo.org/reg_service -(response)-〉"〈OK /〉"

过程 B:通过语义查询寻找 Points 点：

http://reg. foo. org/reg _ service 〈-(request) - lookup ("Search query for Points through specified Point properties")

http://reg.foo.org/reg_service -(response)-〉lookup〈the lookup-match Point(s) profile〉

过程 C:搜索存储有 point ID=http://gw.foo.org/sensor1 信息的存储组件：

http://reg.foo.org/reg_service 〈-(request) - lookup("Search query for Storage that manages point ID=\"http://gw.foo.org/sensor1\.")

http://reg.foo.org/reg_service -(response)-〉It is http://storage.foo.org/storage_service

过程 D:从网关到存储器传输数据：

http://storage.foo.org/storage_service 〈-(request)- data("〈point id=\"http://gw.foo.org/sensor1\" 〉〈value time=\"2009-09-01T00:00:00+08:00\"〉25.5〈/value〉〈/point〉")

http://storage.foo.org/storage_service --(response)--〉"〈OK /〉"

过程 E:应用单元 APP 从存储器检索数据序列:

http://storage.foo.org/storage_service 〈--(request) -query("<query〉Values from time=\"2009-09-01T00:00:00+08:00\" until time=\"2009-10-01T00:00:00+08:00\" for point ID=\"http://gw.foo.org/sensor1\"〈/query〉")

http://storage.foo.org/storage_service --(response)--〉"〈query cursor=\"001\"〉"+ +"〈point id=\"http://gw.foo.org/ sensor1\"〉〈value〉...〈/value〉...〈/point〉"

http://storage.foo.org/storage_service 〈--(request) - query("〈query cursor=\"001\"〉Values from time=\"2009-09-01T00:00:00+08:00\" until time=\"2009-10-01T00:00:00+08:00\" for point ID=\"http://gw.foo.org/sensor1\"〈/query〉")

http://storage.foo.org/storage_service --(response)--〉"〈query cursor=\"001\"〉"+ +"〈point id=\"http://gw.foo.org/ sensor1\"〉〈value〉...〈/value〉...〈/point〉"

http://storage.foo.org/storage_service 〈--(request) - query("〈query cursor=\"001\"〉Values from time=\"2009-09-01T00:00:00+08:00\" until time=\"2009-10-01T00:00:00+08:00\" for point ID=\"http://gw.foo.org/sensor1\"〈/query〉")

http://storage.foo.org/storage_service --(response)--〉"〈query /〉"+ +"〈point id=\"http://gw.foo.org/ sensor1\"〉〈value〉...〈/value〉...〈/point〉"

过程 F:搜索管理点 point ID=http://gw.foo.org/sensor1 的网关组件:

http://reg.foo.org/reg_service 〈--(request) - lookup("Search query for GW that manages point ID=\"http://gw.foo.org/sensor1\.")

http://reg.foo.org/reg_service --(response)--〉It is "http://gw.foo.org/gw_service"

过程 G:应用单元 APP 从网关获取数据:

http://gw.foo.org/gw_service 〈--(request) - query("〈query〉Values select time gives maximum for point ID=\"http://gw.foo.org/sensor1\"〈/query〉")

http://gw.foo.org/gw_service --(response)--〉"〈query /〉"+"〈point id=\"http://gw.foo.org/sensor1\"〉〈value〉...〈/value〉...〈/point〉"

过程 H:网关向应用单元 APP 提交数据通知:

a) 应用单元 APP 周期性的向网关设定事件查询条件

http://gw.foo.org/gw_service 〈--(request) - query("〈query〉post data to \"http://app.foo.org/app_service\" if point id=\"http://gw.foo.org/sensor1\" has updated.〈 /query〉")

http://gw.foo.org/gw_service --(response)--〉"〈OK /〉"

b) 网关向 APP 提交更新的数据

http://app.foo.org/app_service 〈-- (request) - data("〈query〉" + "〈point id=\"http://gw.foo.org/sensor1\"〉〈value time="2009-09-02T01:00:00+08:00"〉30.2〈/value〉〈/point〉")

http://app.foo.org/app_service --(response)--〉"〈OK/〉"

过程 I:应用单元 APP 向网关写入数据:

http://gw.foo.org/gw_service 〈-- (request) - data("〈point id=\"http://gw.foo.org/actuator1\"〉〈value〉true〈/value〉〈/point〉")

http://gw.foo.org/gw_service - (response)--〉"〈OK/〉"

附 录 B
（资料性附录）
典型的设施网络系统部署

设施网络系统的典型部署图见图 B.1。{A, B, ..., G}代表组件，{a, b, ..., h}代表点。

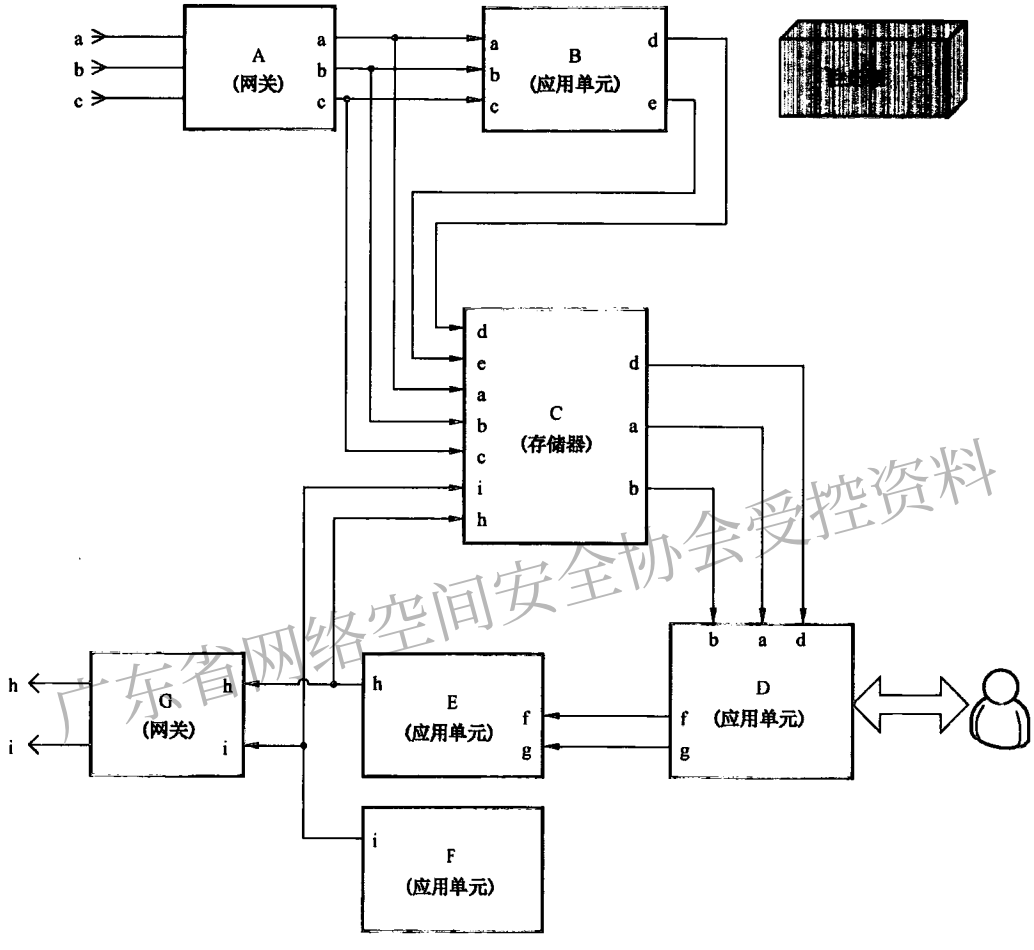


图 B.1 典型的设施网络系统部署图

每一个组件的功能角色如下：

A ——管理{a, b, c}物理传感器的网关；

B ——应用单元(数据处理组件)，读取{a, b, c} 点数据，生成{d, e}点数据；

C ——作为存储{a, b, c, d, e, h, i}点数据的存储器；

D ——应用单元(提供用户接口)，读取{a, b, d}点数据，生成{f, g}点数据；

E ——应用单元(数据交换组件)，读取{f, g}点数据，生成{h}点数据；

F ——应用单元(计时器)，生成{i}点数据；

G ——管理{h, i}物理执行器的网关。

组件的角色信息由注册器进行管理，组件通过查阅注册器找到相应的合作对象，并发送或接收数据值。

附 录 C
（资料性附录）
query 方法和 data 方法应用指南

C.1 FETCH 协议（请求类型为"storage"）。

C.1.1 首次调用 query 方法

```
〈transport〉
  〈header〉
    〈query id="6229c37f-970d-9292-83e4-7c0e54733f8a"
            type="storage"
            acceptableSize="20"〉
     ... SEARCH_KEY ...
    〈/query〉
  〈/header〉
〈/transport〉
```

C.1.2 应答

```
〈transport〉
  〈header〉
    〈query id="6229c37f-970d-9292-83e4-7c0e54733f8a"
            type="storage"
            acceptableSize="20"
             cursor="dab751ed-0133-4ce4-8b7d-ba5c54ce4fb5"〉〈! — cursor is given, so there are
succeeding dataset. —〉
  ... SEARCH_KEY ...
    〈/query〉
    〈OK /〉
  〈/header〉
  〈body〉
    〈point id="..."〉
      〈value time="..."〉...〈/value〉
      〈value time="..."〉...〈/value〉
      〈value time="..."〉...〈/value〉
      ...
    〈/point〉
  〈/body〉
〈/transport〉
```

C.1.3 第二次调用 query 方法

```
〈transport〉
  〈header〉
```

```
    〈query id="6229c37f-970d-9292-83e4-7c0e54733f8a"
           type="storage"
           acceptableSize="20"〉
           cursor="dab751ed-0133-4ce4-8b7d-ba5c54ce4fb5"〉 〈! — query with the given cursor i-
dentifier —〉
      ... SEARCH_KEY ...
    〈/query〉
  〈/header〉
〈/transport〉
```

C.1.4 应答

```
〈transport〉
〈header〉
    〈query id="61af19c3-2da2-4344-aa43-76e640521ce5"
           type="storage" acceptableSize="20" 〉〈! — cursor is not given, so all the dataset has
been retrieved. —〉
      ... SEARCH_KEY ...
    〈/query〉
    〈OK /〉
  〈/header〉
  〈body〉
    〈point id="..."〉
      〈value time="..."〉...〈/value〉
      〈value time="..."〉...〈/value〉
      〈value time="..."〉...〈/value〉
      ...
    〈/point〉
  〈/body〉
〈/transport〉
```

C.2 WRITE 协议

C.2.1 调用 data 方法

```
〈transport〉
  〈body〉
    〈point id="..."〉
    〈value time="..."〉...〈/value〉
    〈/point〉
  〈/body〉
〈/transport〉
```

C.2.2 应答

```
〈transport〉
  〈header〉
```

```
    〈OK/〉
  〈/header〉
〈/transport〉
```

C.3 TRAP 协议（请求类型为"stream"）

C.3.1 调用 query 方法

```
〈transport〉
  〈header〉
    〈query id="9eed9de4-1c48-4b08-a41d-dac067fc1c0d"
            type="stream"
            ttl="60"
            callbackData="http://foo.org/axis/services/GUTAPI"
            callbackControl="http://foo.org/axis/services/GUTAPI" 〉
      ... SEARCH_KEY ...
    〈/query〉
  〈/header〉
〈/transport〉
```

C.3.2 应答

```
〈transport〉
  〈header〉
    〈query id="9eed9de4-1c48-4b08-a41d-dac067fc1c0d"
            type="stream" ttl="60"
            callbackData="http://foo.org/axis/services/GUTAPI"
            callbackControl="http://foo.org/axis/services/GUTAPI" 〉
            ... SEARCH_KEY ...
〈/query〉
    〈OK /〉
  〈/header〉
〈/transport〉
```

C.3.3 调用 callback（Data）的 data 方法

```
〈transport〉
  〈header〉
    〈query id="9eed9de4-1c48-4b08-a41d-dac067fc1c0d"
            type="stream"
            ttl="60"
            callbackData="http://foo.org/axis/services/GUTAPI"
            callbackControl="http://foo.org/axis/services/GUTAPI" 〉
            ... SEARCH_KEY ...
    〈/query〉
  〈/header〉
```

```
  〈body〉
    〈point id="..."〉
    〈value time="..."〉...〈/value〉
    〈/point〉
  〈/body〉
〈/transport〉
```

C.3.4 callback (Data)回送应答消息给信息提供方

```
〈transport〉
  〈header〉
    〈OK/〉
  〈/header〉
〈/transport〉
```

附 录 D
（资料性附录）
组件访问接口的 Web 服务描述语言

对于组件与组件之间的通信，所有的组件都应使用如下的 Web 服务描述语言（WSDL）同时不做任何修改，以便执行针对 RPC 服务端和客户端的 SOAP 通信，这有助于在通信层面（SOAP/XML/HTTP）增加互操作性。

```
<? xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:s0="http://gutp.jp/fiap/2009/11/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:s="http://www.w3.org/2001/XMLSchema"
                  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:tns="http://soap.fiap.org/"
                  targetNamespace="http://soap.fiap.org/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://gutp.jp/fiap/2009/11/">

<s:simpleType name="uuid">
<s:restriction base="s:string">
<s:pattern
value="[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}"/>
</s:restriction>
</s:simpleType>

<s:simpleType name="queryType">
<s:restriction base="s:string">
<s:enumeration value="storage"/>
<s:enumeration value="stream"/>
</s:restriction>
</s:simpleType>

<s:simpleType name="attrNameType">
<s:restriction base="s:string">
<s:enumeration value="time"/>
<s:enumeration value="value"/>
</s:restriction>
</s:simpleType>

<s:simpleType name="selectType">
<s:restriction base="s:string">
<s:enumeration value="minimum"/>
```

```
〈s:enumeration value="maximum"/〉
〈/s:restriction〉
〈/s:simpleType〉

〈s:simpleType name="trapType"〉
〈s:restriction base="s:string"〉
〈s:enumeration value="changed"/〉
〈/s:restriction〉
〈/s:simpleType〉

〈s:complexType name="key"〉
〈s:sequence〉
〈s:element minOccurs="0" maxOccurs="unbounded" name="key" type="s0:key" /〉
〈/s:sequence〉
〈s:attribute name="id" type="s:anyURI" use="required" /〉
〈s:attribute name="attrName" type="s0:attrNameType" use="required" /〉
〈s:attribute name="eq" type="s:string" use="optional" /〉
〈s:attribute name="neq" type="s:string" use="optional" /〉
〈s:attribute name="lt" type="s:string" use="optional" /〉
〈s:attribute name="gt" type="s:string" use="optional" /〉
〈s:attribute name="lteq" type="s:string" use="optional" /〉
〈s:attribute name="gteq" type="s:string" use="optional" /〉
〈s:attribute name="select" type="s0:selectType" use="optional" /〉
〈s:attribute name="trap" type="s0:trapType" use="optional" /〉
〈/s:complexType〉

〈s:complexType name="query"〉
〈s:sequence〉
〈s:element minOccurs="0" maxOccurs="unbounded" name="key" type="s0:key" /〉
〈/s:sequence〉
〈s:attribute name="id" type="s0:uuid" use="required" /〉
〈s:attribute name="type" type="s0:queryType" use="required" /〉
〈s:attribute name="cursor" type="s0:uuid" use="optional" /〉
〈s:attribute name="ttl" type="s:nonNegativeInteger" use="optional" /〉
〈s:attribute name="acceptableSize" type="s:positiveInteger" use="optional" /〉
〈s:attribute name="callbackData" type="s:anyURI" use="optional" /〉
〈s:attribute name="callbackControl" type="s:anyURI" use="optional" /〉
〈/s:complexType〉

〈s:complexType name="error"〉
〈s:simpleContent〉
〈s:extension base="s:string"〉
〈s:attribute name="type" type="s:string" use="required" /〉
```

```
</s:extension>
</s:simpleContent>
</s:complexType>

<s:complexType name="OK">
</s:complexType>

<s:complexType name="header">
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="OK" type="s0:OK" />
<s:element minOccurs="0" maxOccurs="1" name="error" type="s0:error" />
<s:element minOccurs="0" maxOccurs="1" name="query" type="s0:query" />
</s:sequence>
</s:complexType>

<s:complexType name="value">
<s:simpleContent>
<s:extension base="s:string">
<s:attribute name="time" type="s:dateTime" use="optional" />
</s:extension>
</s:simpleContent>
</s:complexType>

<s:complexType name="point">
<s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="value" type="s0:value" />
</s:sequence>
<s:attribute name="id" type="s:anyURI" use="required" />
</s:complexType>

<s:complexType name="pointSet">
<s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="pointSet" type="s0:pointSet" />
<s:element minOccurs="0" maxOccurs="unbounded" name="point" type="s0:point" />
</s:sequence>
<s:attribute name="id" type="s:anyURI" use="required" />
</s:complexType>

<s:complexType name="body">
<s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="pointSet" type="s0:pointSet"/>
<s:element minOccurs="0" maxOccurs="unbounded" name="point" type="s0:point"/>
</s:sequence>
```

```
</s:complexType>

<s:complexType name="transport">
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="header" type="s0:header" />
<s:element minOccurs="0" maxOccurs="1" name="body" type="s0:body" />
</s:sequence>
</s:complexType>

<s:element name="transport" type="s0:transport" />

</s:schema>

<s:schema elementFormDefault="qualified" targetNamespace="http://soap.fiap.org/">
<s:import namespace="http://gutp.jp/fiap/2009/11/" />

<s:element name="queryRQ">
<s:complexType>
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" ref="s0:transport" />
</s:sequence>
</s:complexType>
</s:element>

<s:element name="queryRS">
<s:complexType>
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" ref="s0:transport" />
</s:sequence>
</s:complexType>
</s:element>

<s:element name="dataRQ">
<s:complexType>
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" ref="s0:transport" />
</s:sequence>
</s:complexType>
</s:element>

<s:element name="dataRS">
<s:complexType>
<s:sequence>
```

```
〈s:element minOccurs="1" maxOccurs="1" ref="s0:transport" /〉
〈/s:sequence〉
〈/s:complexType〉
〈/s:element〉

〈/s:schema〉
〈/wsdl:types〉
〈wsdl:message name="querySoapIn"〉
〈wsdl:part name="parameters" element="tns:queryRQ" /〉
〈/wsdl:message〉
〈wsdl:message name="querySoapOut"〉
〈wsdl:part name="parameters" element="tns:queryRS" /〉
〈/wsdl:message〉
〈wsdl:message name="dataSoapIn"〉
〈wsdl:part name="parameters" element="tns:dataRQ" /〉
〈/wsdl:message〉
〈wsdl:message name="dataSoapOut"〉
〈wsdl:part name="parameters" element="tns:dataRS" /〉
〈/wsdl:message〉
〈wsdl:portType name="FIAPServiceSoap"〉
〈wsdl:operation name="query"〉
〈wsdl:input message="tns:querySoapIn" /〉
〈wsdl:output message="tns:querySoapOut" /〉
〈/wsdl:operation〉
〈wsdl:operation name="data"〉
〈wsdl:input message="tns:dataSoapIn" /〉
〈wsdl:output message="tns:dataSoapOut" /〉
〈/wsdl:operation〉
〈/wsdl:portType〉
〈wsdl:binding name="FIAPServiceSoap" type="tns:FIAPServiceSoap"〉
〈soap:binding transport="http://schemas.xmlsoap.org/soap/http" /〉
〈wsdl:operation name="query"〉
〈soap:operation soapAction="http://soap.fiap.org/query" style="document" /〉
〈wsdl:input〉
〈soap:body use="literal" /〉
〈/wsdl:input〉
〈wsdl:output〉
〈soap:body use="literal" /〉
〈/wsdl:output〉
〈/wsdl:operation〉
〈wsdl:operation name="data"〉
〈soap:operation soapAction="http://soap.fiap.org/data" style="document" /〉
〈wsdl:input〉
```

```
〈soap:body use="literal" /〉
〈/wsdl:input〉
〈wsdl:output〉
〈soap:body use="literal" /〉
〈/wsdl:output〉
〈/wsdl:operation〉
〈/wsdl:binding〉
〈wsdl:service name="FIAPWS"〉
〈wsdl:port name="FIAPServiceSoap" binding="tns:FIAPServiceSoap"〉
〈soap:address location="http://localhost/axis2/services/FIAPWS" /〉
〈/wsdl:port〉
〈/wsdl:service〉
〈/wsdl:definitions〉
```

附 录 E
（资料性附录）
注册器访问接口的 Web 服务描述语言

对于组件和注册器之间的通信，所有的 UGCCNet 组件和注册器都应使用如下的 Web 服务描述语言（WSDL）同时不做任何修改，以便执行针对 RPC 服务端和客户端的 SOAP 通信，这有助于在通信层面（SOAP/XML/HTTP）增加互操作性。

```
〈? xml version="1.0" encoding="UTF-8"?〉
〈wsdl:definitions
targetNamespace="http://soap.fiap.org/"
xmlns:s0="http://gutp.jp/fiap-mgmt/2009/11/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://soap.fiap.org/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"〉

〈wsdl:types〉
〈s:schema
elementFormDefault="qualified"
targetNamespace="http://gutp.jp/fiap-mgmt/2009/11/"〉

〈s:simpleType name="uuid"〉
〈s:restriction base="s:string"〉
〈s:pattern
value="[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}"/〉
〈/s:restriction〉
〈/s:simpleType〉

〈s:simpleType name="streamType"〉
〈s:restriction base="s:string"〉
〈s:enumeration value="in"/〉
〈s:enumeration value="out"/〉
〈s:enumeration value="in|out"/〉
〈/s:restriction〉
〈/s:simpleType〉

〈s:simpleType name="attrNameType"〉
〈s:restriction base="s:string"〉
〈s:enumeration value="time"/〉
〈s:enumeration value="value"/〉
〈/s:restriction〉
```

```
</s:simpleType>

<s:simpleType name="lookupType">
<s:restriction base="s:string">
<s:enumeration value="component"/>
<s:enumeration value="point"/>
</s:restriction>
</s:simpleType>

<s:complexType name="key">
<s:attribute name="id" type="s:anyURI" use="required"/>
<s:attribute name="attrName" type="s0:attrNameType" use="required"/>
<s:attribute name="stream" type="s0:streamType" use="optional"/>
<s:attribute name="limit" type="s:nonNegativeInteger" use="optional"/>
<s:attribute name="eq" type="s:string" use="optional"/>
<s:attribute name="neq" type="s:string" use="optional"/>
<s:attribute name="lt" type="s:string" use="optional"/>
<s:attribute name="gt" type="s:string" use="optional"/>
<s:attribute name="lteq" type="s:string" use="optional"/>
<s:attribute name="gteq" type="s:string" use="optional"/>
</s:complexType>

<s:complexType name="lookup">
<s:sequence>
<s:element maxOccurs="unbounded" minOccurs="0" name="key" type="s0:key"/>
<s:element maxOccurs="unbounded" minOccurs="0" name="point" type="s0:point"/>
</s:sequence>
<s:attribute name="id" type="s0:uuid" use="required"/>
<s:attribute name="type" type="s0:lookupType" use="required"/>
</s:complexType>

<s:complexType name="component">
<s:sequence>
<s:element maxOccurs="unbounded" minOccurs="0" name="key" type="s0:key"/>
</s:sequence>
<s:attribute name="name" type="s:string" use="required"/>
<s:attribute name="uri" type="s:anyURI" use="required"/>
<s:attribute name="priority" type="s:nonNegativeInteger" use="optional"/>
<s:attribute name="support" type="s:string" use="required"/>
<s:attribute name="expires" type="s:nonNegativeInteger" use="optional"/>
</s:complexType>

<s:complexType name="error">
```

```
〈s:simpleContent〉
〈s:extension base="s:string"〉
〈s:attribute name="type" type="s:string" use="required"/〉
〈/s:extension〉
〈/s:simpleContent〉
〈/s:complexType〉

〈s:complexType name="OK"〉
〈/s:complexType〉

〈s:complexType name="header"〉
〈s:sequence〉
〈s:element maxOccurs="1" minOccurs="0" name="OK" type="s0:OK"/〉
〈s:element maxOccurs="1" minOccurs="0" name="error" type="s0:error"/〉
〈s:element maxOccurs="1" minOccurs="0" name="lookup" type="s0:lookup"/〉
〈/s:sequence〉
〈/s:complexType〉

〈s:complexType name="point"〉
〈s:attribute name="id" type="s:anyURI" use="optional"/〉
〈s:anyAttribute namespace="##other" processContents="lax"/〉
〈/s:complexType〉

〈s:complexType name="body"〉
〈s:sequence〉
〈s:element maxOccurs="unbounded" minOccurs="0" name="component" type="s0:component"/〉
〈s:element maxOccurs="unbounded" minOccurs="0" name="point" type="s0:point"/〉
〈/s:sequence〉
〈/s:complexType〉

〈s:complexType name="transport"〉
〈s:sequence〉
〈s:element maxOccurs="1" minOccurs="0" name="header" type="s0:header"/〉
〈s:element maxOccurs="1" minOccurs="0" name="body" type="s0:body"/〉
〈/s:sequence〉
〈/s:complexType〉

〈s:element name="transport" type="s0:transport"/〉

〈/s:schema〉
〈s:schema elementFormDefault="qualified" targetNamespace="http://soap.fiap.org/"〉
〈s:import namespace="http://gutp.jp/fiap-mgmt/2009/11/"/〉
```

```
〈s:element name="registrationRQ"〉
〈s:complexType〉
〈s:sequence〉
〈s:element maxOccurs="1" minOccurs="1" ref="s0:transport"/〉
〈/s:sequence〉
〈/s:complexType〉
〈/s:element〉

〈s:element name="registrationRS"〉
〈s:complexType〉
〈s:sequence〉
〈s:element maxOccurs="1" minOccurs="1" ref="s0:transport"/〉
〈/s:sequence〉
〈/s:complexType〉
〈/s:element〉

〈s:element name="lookupRQ"〉
〈s:complexType〉
〈s:sequence〉
〈s:element maxOccurs="1" minOccurs="1" ref="s0:transport"/〉
〈/s:sequence〉
〈/s:complexType〉
〈/s:element〉

〈s:element name="lookupRS"〉
〈s:complexType〉
〈s:sequence〉
〈s:element maxOccurs="1" minOccurs="1" ref="s0:transport"/〉
〈/s:sequence〉
〈/s:complexType〉
〈/s:element〉

〈/s:schema〉
〈/wsdl:types〉
〈wsdl:message name="registrationSoapIn"〉
〈wsdl:part name="parameters" element="tns:registrationRQ"〉
〈/wsdl:part〉
〈/wsdl:message〉
〈wsdl:message name="registrationSoapOut"〉
〈wsdl:part name="parameters" element="tns:registrationRS"〉
〈/wsdl:part〉
〈/wsdl:message〉
〈wsdl:message name="lookupSoapIn"〉
```

```
〈wsdl:part name="parameters" element="tns:lookupRQ"〉
〈/wsdl:part〉
〈/wsdl:message〉
〈wsdl:message name="lookupSoapOut"〉
〈wsdl:part name="parameters" element="tns:lookupRS"〉
〈/wsdl:part〉
〈/wsdl:message〉
〈wsdl:portType name="FIAPMgmtServiceSoap"〉
〈wsdl:operation name="registration"〉
〈wsdl:input message="tns:registrationSoapIn"〉
〈/wsdl:input〉
〈wsdl:output message="tns:registrationSoapOut"〉
〈/wsdl:output〉
〈/wsdl:operation〉
〈wsdl:operation name="lookup"〉
〈wsdl:input message="tns:lookupSoapIn"〉
〈/wsdl:input〉
〈wsdl:output message="tns:lookupSoapOut"〉
〈/wsdl:output〉
〈/wsdl:operation〉
〈/wsdl:portType〉
〈wsdl:binding name="FIAPMgmtServiceSoap" type="tns:FIAPMgmtServiceSoap"〉
〈soap:binding transport="http://schemas.xmlsoap.org/soap/http"/〉
〈wsdl:operation name="registration"〉
〈soap:operation soapAction="http://soap.fiap.org/registration" style="document"/〉
〈wsdl:input〉
〈soap:body use="literal"/〉
〈/wsdl:input〉
〈wsdl:output〉
〈soap:body use="literal"/〉
〈/wsdl:output〉
〈/wsdl:operation〉
〈wsdl:operation name="lookup"〉
〈soap:operation soapAction="http://soap.fiap.org/lookup" style="document"/〉
〈wsdl:input〉
〈soap:body use="literal"/〉
〈/wsdl:input〉
〈wsdl:output〉
〈soap:body use="literal"/〉
〈/wsdl:output〉
〈/wsdl:operation〉
〈/wsdl:binding〉
〈wsdl:service name="FIAPMgmtWS"〉
```

```
〈wsdl:port name="FIAPMgmtServiceSoap" binding="tns:FIAPMgmtServiceSoap"〉
〈soap:address location="http://localhost/axis2/services/FIAPMgmtWS"/〉
〈/wsdl:port〉
〈/wsdl:service〉
〈/wsdl:definitions〉
```

附 录 F
（资料性附录）
组件与组件通信的错误类型

组件与组件之间通信的错误类型见表 F.1。

表 F.1 组件与组件之间通信的错误类型

错误类型	语义说明
INVALID_REQUEST	请求的消息结构错误
INVALID_CURSOR	FETCH 请求中指定的指针未在该服务器管理范围下
QUERY_NOT_SUPPORTED	服务器不支持 Key 中指定的属性操作(例如 eq，lt，gteq,...)
POINT_NOT_FOUND	Key 中指定的点标识不在该服务器的管理范围下
FORBIDDEN	访问被禁止，未被处理
TOO_MANY_KEYS	查询中指定的 Key 的数量超出了服务器配置的上限
TOO_MANY_CURSOR_SESSIONS	含有指针的会话数量超出了服务器配置的上限
TOO_MANY_POINTS	在消息体中指定的点的数量超出了服务器配置的上限
TOO_MANY_VALUES	在消息体里指定的数据值的数量超出了服务器配置的上限
VALUE_TIME_NOT_SPECIFIED	在数据值中，该客户端应指明时间属性
TIME_FORMAT	数据值的时间属性的格式 eq、neq、lt、gt、lteq、gteq 无效。应遵循 ISO 8601 的格式，显示秒和时区
TIME_PRECISION	在请求中指定的时间精度太详细，无法精确处理
CALLBACK_CONTROL_NOT_SUPPORTED	客户端在 TRAP 的请求查询中指定了 callbackControl 属性，但是服务端不支持 callbackControl 功能
TRAP_ON_POINTSET	客户端尝试在点集合上进行 TRAP 协议
TOO_MANY_TRAP_QUERIES	服务器已经超过了 TRAP 查询管理的上限
INVALID_KEY_IN_TRAP_REQUEST	TRAP 查询中指定的 Key 是无效的
SERVER_ERROR	服务器遇到问题：如内存资源问题，I/O 接口错误
UNKNOWN	由于某些原因(有时原因未知)，服务器无法处理该请求

附 录 G
（资料性附录）
组件与注册器通信的错误类型

组件与注册器通信的错误类型见表 G.1。

表 G.1 组件与注册器通信的错误类型

错误类型	语义说明
INVALID_REQUEST	请求的消息结构错误
POINT_NOT_FOUND	Key 中指定的点标识不在该服务器的管理范围下
TOO_MANY_KEYS	在查询 lookup 命令中指定的 Key 的数量超过了服务器配置的上限
TOO_MANY_POINTS	在消息中指定的点的数量超过了服务器配置的上限
SERVER_ERROR	服务器遇到问题：如内存资源问题，I/O 接口错误
UNKNOWN	由于某些原因（有时原因未知），服务器无法处理该请求

中华人民共和国
国家标准
信息技术 系统间远程通信和信息交换
社区节能控制网络协议
GB/T 36451—2018/ISO/IEC/IEEE 18880:2015
*
中国标准出版社出版发行
北京市朝阳区和平里西街甲2号(100029)
北京市西城区三里河北街16号(100045)
网址 www.spc.net.cn
总编室:(010)68533533 发行中心:(010)51780238
读者服务部:(010)68523946
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 3.75 字数 106 千字
2018年6月第一版 2018年6月第一次印刷
*
书号:155066·1-60340 定价 51.00元

GB/T 36451-2018

打印日期:2018年7月11日 F002AA