



国际电气工程先进技术译丛

# OPC统一架构

OPC Unified Architecture

Wolfgang Mahnke

(德) Stefan-Helmut Leitner

Matthias Damm

著

马国华

译

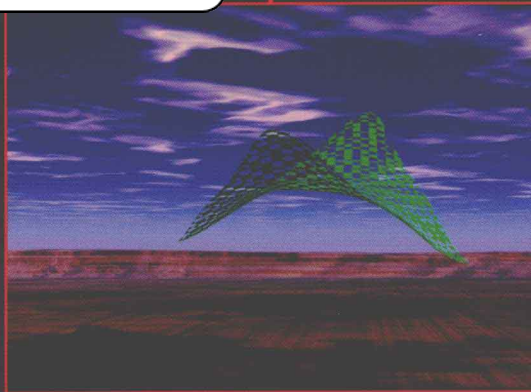
## 关于本电子书说明

本人由于一些便利条件，可以帮您提供各种中文电子图书资料，且质量均为清晰的PDF图片格式，方便阅读和携带。文学、法律、计算机、人文、经济、医学、工业、学术等方面的图书，都可以帮您找提供电子版本，500万图书馆资源收藏供你选择。

我的QQ是859109769 佳佳e图书（提供完整版）



机械工业出版社  
CHINA MACHINE PRESS



## 国际电气工程先进技术译丛 传播国际最新技术成果 搭建电气工程技术平台

- 《OPC统一架构》
- 《碳捕获与封存》
- 《可再生能源系统——100%可再生能源解决方案的选择与模型》
- 《机电系统中的传感器与驱动器——设计与应用》
- 《LED照明应用技术》
- 《零排放动力循环》
- 《柔性交流输电系统在电网中的建模与仿真》
- 《风电并网：联网与系统运行》
- 《可再生能源的转换、传输和储存》
- 《海底电力电缆——设计、安装、修复和环境影响》
- 《光伏技术与工程手册》
- 《风力发电的模拟与控制》
- 《风电场并网稳定性技术》
- 《智能电网中的电力电子技术》
- 《电接触理论、应用与技术》
- 《电磁屏蔽原理与应用》
- 《高效可再生分布式发电系统》
- 《电网保护》
- 《分布式发电——感应和永磁发电机》
- 《电力系统谐波》
- 《风能与太阳能发电系统——设计、分析与运行》（原书第2版）
- 《瞬时功率理论及其在电力调节中的应用》
- 《风力机控制系统原理、建模及增益调度设计》
- 《高压输配电设备实用手册》
- 《电力变流器电路》
- 《电力系统中的电磁兼容》
- 《超高压交流输电工程》（原书第3版）
- 《高压直流输电与柔性交流输电控制装置——静止换流器在电力系统中的应用》
- 《电磁兼容原理与应用》（原书第2版）
- 《电力电子技术手册》
- 《基于晶闸管的柔性交流输电控制装置》
- 《电力电容器》
- 《电力系统谐波——基本原理、分析方法和滤波器设计附习题解答》
- 《配电可靠性与电能质量》
- 《现代电动汽车、混合动力电动汽车和燃料电池车——基本原理、理论和设计》（原书第2版）

上架指导：工业技术 / 电气工程 / 自动

● ISBN 978-7-111-35825-1

● 封面设计：马精明

定价：88.00元

地址：北京市百万庄大街22号 邮政编码：100037  
电话服务 网络服务  
社服务中心：(010)88361066 门户网：<http://www.cmpbook.com>  
销售一部：(010)68326294 教材网：<http://www.cmpedu.com>  
销售二部：(010)88379649 封面无防伪标均为盗版  
读者购书热线：(010)88379203

ISBN 978-7-111-35825-1



9 787111 358251

国际电气工程先进技术译丛

# OPC 统一架构

Wolfgang Mahnke

(德) Stefan-Helmut Leitner 著

Matthias Damm

马国华 译



机械工业出版社

## 图书在版编目 (CIP) 数据

OPC 统一架构/(德)马科 (Mahnke, W.) 等著; 马国华译. —北京: 机械工业出版社, 2011. 10

(国际电气工程先进技术译丛)

书名原文: OPC Unified Architecture

ISBN 978-7-111-35825-1

I. ①O… II. ①马… ②马… III. ①过程控制—自动控制系统 IV. ①TP273

中国版本图书馆 CIP 数据核字 (2011) 第 185245 号

机械工业出版社 (北京市百万庄大街 22 号 邮政编码 100037)

策划编辑: 张俊红 责任编辑: 张俊红

版式设计: 霍永明 责任校对: 樊钟英

封面设计: 马精明 责任印制: 乔 宇

北京瑞德印刷有限公司印刷 (三河市胜利装订厂装订)

2012 年 1 月第 1 版第 1 次印刷

169mm×239mm · 19 印张 · 379 千字

0001—3000 册

标准书号: ISBN 978-7-111-35825-1

定价: 88.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

电话服务

网络服务

社服务中心: (010)88361066

门户网: <http://www.cmpbook.com>

销售一部: (010)68326294

教材网: <http://www.cmpedu.com>

销售二部: (010)88379649

读者购书热线: (010)88379203

封面无防伪标均为盗版

OPC 基金会定义了在线自动化系统之间的数据交换标准，成功地应用在工控自动化行业中。新的 OPC 统一架构（OPC UA）统一了现行标准，采用面向服务的架构（SOA）。原来的 OPC 应用程序只能运行在基于 Windows 的 PC 系统上，而新标准的主要优点在于平台的无关性。

本书首先简单介绍了传统 OPC，并简短概述了 OPC UA；接下来侧重介绍信息建模，也就是使用 OPC UA 数据如何被表现，并介绍了如何操作已经建立好的 OPC UA 信息模型；然后对 OPC UA 的安全方面进行了讨论，介绍了 OPC UA 的应用程序架构和系统架构，并介绍了如何从传统的 OPC 概念映射到 OPC UA 概念，如何从传统 OPC 应用程序转移到 OPC UA 的方法；最后本书对 OPC UA 协议子集进行介绍，并对 OPC UA 与传统 OPC 的性能进行了对比。对于那些需要进一步查询 OPC UA 详情的读者，本书的附录给出了参考。

本书对 OPC UA 整体进行了深入浅出、易于理解的介绍，并对一些标准中不直接提到的概念进行了解释。本书既可以作为工控自动化专业研究生和老师的参考，也为从事自动化工程应用的工程技术人员以及研发人员提供了宝贵的参考资料。

Translation from the English language edition;

OPC Unified Architecture; edited by Wolfgang Mahnke, Stefan-Helmut Leitner, Matthias Damm; Copyright © 2009 by Springer LLC.

All Rights Reserved.

本书简体中文版由 Springer 出版社授权机械工业出版社独家出版。

版权所有，侵权必究。

本书版权登记号：图字 01—2010—4073 号。

## 译 者 序

OPC UA (Unified Architecture, 统一架构) 作为 OPC 基金会大力推介的最新标准, 其相关文档和资料基本上都是英文的, 不利于国内工业控制软件的发展。本书的英文版是由业内的权威专家完成, 并得到了较好的评价。其内容较为易懂, 适合初次接触 OPC UA 的用户阅读。为了推动 OPC UA 在国内的普及, 特将其翻译为中文出版。

本书的中文版是 OPC UA 规范的第一本中文书籍, 由于 OPC UA 规范是一个全新的规范, 有一些术语目前尚无通用的中文词汇。另外, 一些概念在新的规范中已经有了新的含义。还有些词汇在不同的行业和企业内部有不同的中文译法。本书的译者在查阅了大量相关资料后, 最终诚惶诚恐地选择了相对合适的译法, 与部分行业 and 企业的惯用方法可能有所不同, 请读者注意这些差别。

本书的中文专业词汇在第一次出现时, 都在括号内给出了英文原文, 而一些在软件开发和设计中作为关键字出现的英文 (如类或者服务的名称), 不方便翻译成中文的, 首次出现时都在英文后面用括号给出了中文释义, 具体列表见附录 D。

有一些专业词汇对于理解本书的内容和 OPC UA 规范都非常重要:

Node, 中文一般翻译为节点或结点, 在本书中, 统一使用“节点”。

Reference, 中文一般翻译为参考或引用, 在本书中, 统一使用“引用”。

Attribute 和 Property, 中文都翻译为属性, 在 OPC UA 中, 这两个词汇同时出现, 根据其含义的差别, 本书将 Attribute 翻译为“属性”, 将 Property 翻译为“特性”。在译者认为容易混淆之处, 会将英文附在括号中, 以帮助读者甄别。

Profile, 根据用途不同, 中文有翻译为“配置”、“轮廓”、“概要”等, 在本书中, 根据其表达的意思, 翻译为“协议子集”。

其他一些术语的翻译见附录 E。

由于本书新术语较多, 限于译者的水平, 不妥之处在所难免, 还望读者见谅。

本书翻译的过程中, 得到了 OPC 基金会中国促进会的大力支持。另外, 来自北京三维力控科技有限公司的殷远超、何迪江、陆云鹏、季然和童伟等人也为本书的翻译工作做出了重大贡献, 在此一并表示感谢。

译 者

# 原 书 序 一

OPC 基金会非常荣幸地推荐这本难得的好书，并感谢作者们付出的辛勤劳动。这本书提供了关于理解 OPC 基金会的技术和标准的坚实的框架，从最初到现在，以及最重要的 OPC 互操作性标准——这就是 OPC UA。

我很高兴认识这本书的作者已经很多年了，很显然他们是架构和开发方面的领导者，这使得 OPC 取得了如此瞩目的成功。这本书的读者们很幸运地能从这些实际上参与了开发 OPC 基金会 UA 规范和技术的专家们那里学到知识。

OPC UA 是由 30 多家公司花了约 5 年多时间共同开发的。完整的引用实现和技术的开发为的是使规范生效，以及证明技术的可行性。OPC UA 的目的是为工厂车间和企业之间的数据和信息传递提供一个与平台无关的互操作性标准。很明显，在开发的过程中，OPC UA 定位准确，延伸到了工业自动化之外的范围。OPC 已经涉及楼宇自动化、安全、家庭自动化、发电、包装以及石化领域。由于 OPC UA 的高度可扩展的架构，对智能嵌入式设备的部署也是很好的定位。

OPC UA 也是和其他标准制定机构通力合作的成果。OPC UA 是在复用现有的技术的前提下创建的，它汇集了所有必要的细节，为的是获得最真实、安全、可靠的互操作性。

这本书为您提供了坚实的基础，您可以学到一切您想知道的在 OPC UA 基础上为多个厂商的互操作性开发世界级产品的所有知识。

OPC 基金会很自豪地看到作者们从着手编著到出版这一路以来所付出的劳动以及取得的成就。我鼓励大家反复阅读这本书，在各自开发的领域或使用 OPC 产品的时候将其作为一本长期参考资料。作为读者，您将会庆幸自己拥有了这本书！

我鼓励大家不断地参考和充分利用好这本书。

Tom Burke

OPC 基金会总裁兼执行董事

## 原 书 序 二

在 5 年多的制作过程中，对于那些想和其他公司进行软件交互的厂商来说，OPC UA 象征了革命性的一步。OPC 基金会的一群具有奉献精神志愿者们献出了无数宝贵时间才完成了这一巨作。通过结合上一代 OPC 接口的功能以及计算机科学的最新成果，例如面向对象编程（Object Oriented Programming, OOP）、面向服务的体系结构（Service Oriented Architecture, SOA）、语义 Web（The Semantic Web）和网络模型数据库（Network Model Databases），OPC UA 代表了生产/消费数据以及任何复杂元数据（Metadata）的通用框架。

在 20 世纪 90 年代中期，OPC 基金会发表了它的第一个 Microsoft COM 规范——数据存取（DA），它规定了服务器如何暴露一个简单的层次组织（标签，tag），并由符合的客户端读取、写入和订阅。很快基金会紧接着推出了不同数据类型的其他的较受欢迎的规范，尤其是报警和事件（Alarms&Events, A&E）以及历史数据规范。随着 XML Web Services 技术的出现，基金会创建了 XML-DA，一个为 DA 设计的平台无关的 Web 服务接口，具有和原先的 COM 版相似的功能。但不幸的是，Web Service 版的性能比 COM 版慢了几个数量级，所以 XML-DA 没能成为特定平台的 COM 版更新更好的替代方案。为了创建一个真正的在运行中不丢失任何特点（或性能）的基于现存 COM 规范的替代品，OPC UA 因而在这样的呼声中诞生了。下面是 OPC UA 的一些设计理念：

- 支持使用复杂的而非简单数据的更广范围的应用程序（MES、ERP 和资产管理等）。
- 允许使用一组单一的、通用的和数据无关的服务集，暴露数据存取（DA）、报警和事件以及历史数据。
- 允许地址空间中的节点以层次和非层次网络连接。
- 通过使它们更抽象，不依赖于现有的通信技术，使规范能长期被使用。
- 通过使用已被广为接受的因特网标准（Web Services、XML、HTTP 和 TCP 等）来详细指明具体的数据序列化和协议映射。
- 允许暴露（以和数据本身相同的方式）富元数据（Rich Metadata），从而使普通用户能够解释数据而不用具备先验知识。

OPC UA 实现了所有的这些目标甚至更多。现在我们有可执行的规范、



## VI OPC 统一架构

通信堆栈和在多个编程语言里的 SDK，以及第三方的高级开发工具包，我们真诚地邀请全世界的开发者共同使用 OPC UA 来创造高水平的互操作的软件应用程序。

由于高端的设计目标以及 OPC UA 的巨大范畴，目前该规范由 13 部分组成，简炼、精确而逐步深入。对于开发者和架构师来说，规范本身不是接近 OPC UA 最容易的方法。以 SQL 为例，有成千上万的软件开发人员能非常熟练地开发 SQL 应用程序，但是其中大多数人从来没有读过 SQL 规范。他们通过读书、学习开发商文档、上课等方式来学习了解 SQL。这本书是由 OPC UA 规范和代码的主要作者和贡献者们所写的，代表了您在编程和设计任务中学习和使用 OPC UA 最行之有效的方法。

Jim Luth

OPC 基金会技术董事

# 原 书 前 言

## 本书的目的

OPC 基金会提供用于工业自动化的数据交换的规范。以 COM/DCOM 为基础的规范已有很长的历史，最显著的 OPC 数据存取 (DA)、OPC 报警和事件 (A&E) 以及 OPC 历史数据存取 (HDA)，都已被业界广泛接受，并被几乎所有以工业自动化为目标的系统所实现。

现在 OPC 基金会发布了名为 OPC 统一架构 (OPC UA) 的最新一代 OPC 规范。通过 OPC UA，OPC 基金会实现了从过期的 COM/DCOM 技术向面向服务的架构 (SOA) 的技术迁移，它通过 Web 服务或者优化的基于 TCP 的平台无关的方式提供数据。OPC UA 将以前的规范统一成一个单一的地址空间，提供当前数据处理、报警和事件、历史数据以及事件历史等功能。

OPC UA 的一个了不起的改进是地址空间模型，通过它，供应商可以用面向对象技术发布丰富和可扩展的信息模型。OPC UA 非常好地适用于智能设备、控制器、DCS 和 SCADA 系统乃至 MES 和 ERP 系统。它也很好地适用在它提供信息方面的能力：在低端，可以使用一种提供基本信息的类似经典 OPC 的模型；在高端，可以描述高度复杂的模型，提供大量的包括复杂类型层次结构的元数据。

很多领域对高级建模能力有着很高的兴趣，已经有些措施来标准化 OPC UA 的信息模型。例如现场设备集成 (FDI)，针对常见现场设备的描述和使用 MIMOSA (维护信息-ERP 等)、S95 (产品信息-MES) 和 PLCopen (工业控制) 的日常活动。

OPC UA 规范目前由 13 部分构成，因此有 13 个文件。其中一些详细阐明了一些基础技术，另外一些着重说明特定的信息模式，例如一个说明如何提供流程自动化的具体报警信息的模型。总之，规范共有 700 多页，尽可能地写得精确和完整。规范最主要的是阐述如何做，较少程度上解释了为什么要设计成这样。规范就是这样，一定要被写出来以及被很多开发者广泛应用并保证在不同应用程序之间的互通性。所以，规范对于一些刚接触 OPC UA 的新人来说有点难以理解，就好像其他的规范 (如 SQL 或者 UML) 一样。

我们想用本书来填补这个空白，提供一条通向 OPC UA 的简单易懂的路径。我们不会提供和规范一样同级别的细节，但是宁愿介绍并解释 OPC UA 的主要

概念。我们会给出指导方针来帮助您在不同概念之间做出最好的选择，我们还会把目标锁定在那些在规范里没有被直接提起，但是在平时 OPC UA 的应用中用到的相关议题。

### 谁应该阅读这本书？

如果您对 OPC UA 感兴趣——这也许是您正在阅读本书的原因——您应该阅读这本书。本书的作者是编写 OPC UA 规范的最主要的几位成员，他们会向您解释缩写字 OPC UA 背后的故事。

我们在写这本书时，脑海里就有了一个很广泛的读者群，包括以下人群：

- 判断是否运用 OPC UA 到他们的应用中去的人（决策者）；
- 应用 OPC UA 到他们的客户端或服务器应用程序中的人（软件架构师、工程师和软件开发人员）；
- 使用基于 OPC UA 的应用程序的人（管理员和工程师，如负责配置过程控制系统；非终端用户，如过程控制系统操作员等）。

本书会介绍 OPC UA 通信和信息建模概念，还会解释如何定义您的模型和存取数据。您将学会冗余、安全等是如何在 OPC UA 中定位的，以及它和传统 OPC 相比，如何更好地执行。但是您不会找到任何的编码示例。当您在实现您的 OPC UA 应用程序时，您可能会使用 SDK，您应该在 SDK 文档中查找样例代码。本书解释了藏在 OPC UA SDK 和 OPC UA Service 后面的机制是如何工作的，以及如何用 OPC UA 构建您的信息模型。同时，本书还解释了当您进入 OPC UA 服务器时将会碰到的一些信息。

您不用非得熟悉传统 OPC 才阅读本书，但您应该对面向对象概念有一个基本的了解，才能理解 OPC UA 的信息模型，为了更好地理解 OPC UA，一些对软件设计的基础知识也是必需的。

### 提纲

第 1 章在说明 OPC UA 动机之前简单介绍了传统 OPC 以及简短概述了 OPC UA。

接下来的三章侧重于信息建模，也就是使用 OPC UA 数据如何被表现。第 2 章介绍了建模概念。我们从基本的概念开始，以提供数据，然后介绍一些相对复杂的构造，如类型层次结构。第 3 章介绍了一个如何用 OPC UA 建立模型的真实案例，然后通过解释一些最好的实践来概括信息模型。第 4 章介绍了标准信息模型，从解释什么是信息模型开始，如何被具体制定，然后是 OPC UA 如何处理信息模型。接着，通过 OPC UA 规范的更专业的信息模型延伸来介绍

OPC UA 信息模型的基础。最后会看一下由其他组织机构提供的额外的信息建模标准的当前状态。

在接下来的两章里会侧重于如何操作已经建立好的 OPC UA 信息模型。在第 5 章里描绘了抽象的服务 (Service)，它用来访问和操控数据。在第 6 章里介绍了如何将服务对应到相应的技术，包括数据如何被序列化、信息如何被加固、使用了什么传输协议。

在第 7 章里，就 OPC UA 的安全方面进行了讨论。这包括了藏在 OPC UA 安全模式后面的理论思考以及对应用 OPC UA 的开发者和管理员的实际意义。

第 8 章解释了 OPC UA 的应用程序架构。在这里介绍了实现 OPC UA 时会用到的不同的组件。

第 9 章描述了 OPC UA 的系统架构。包括对您在自己的系统中如何部署和配置 OPC UA 应用程序，以及如何处理备份、服务器的聚集等。

在接下来的章节中，着眼于从现有的应用程序迁移到 OPC UA。在第 10 章里，解释了如何从传统 OPC 的概念映射到 OPC UA 概念。这一章对于那些有着传统 OPC 深厚知识的读者来说特别有用。第 11 章提供了如何从传统 OPC 应用程序转移到 OPC UA 的策略，以及由 OPC 基金会提供的哪些组件在这方面可以帮助您。

OPC UA 详细阐明了很多特点，但并不是每一个应用程序都会利用到所有的这些。OPC UA 提供了一些协议子集 (Profile) 来处理这个情况，协议子集说明了一个产品需要包括的子特性。应用程序以交换这些协议子集信息来知道它们可以从别的应用程序中得到哪些。协议子集的详情以及如何组织它们在第 12 章中阐述。

在如今使用传统 OPC 的情节中，性能是一个关键因素。在第 13 章里给出了 OPC UA 性能方面的考虑，其中包括了 OPC UA 和传统 OPC 之间性能的比较。

第 14 章是本书的结尾，在这一章里，对 OPC UA 做出了总结，讨论了 OPC UA 的复杂性，指出了在大多数情况下很简单，也解释了为什么有些部分必须具有复杂性。同时就 OPC UA 方面，期待在不久的将来发生什么而做出了展望。

此外，当需要寻找 OPC UA 的一些详情时，附录给您提供了便捷的引用。

# 作者介绍

## **Wolfgang Mahnke 博士**

Wolfgang Mahnke 供职于德国的 ABB 集团研究中心工业软件技术领域。近年来，他一直是几个有关 OPC UA 项目的项目负责人。这些项目都着眼于 OPC UA 规范，由 OPC UA 基金会提供的基础设施的执行以及 ABB 内部 OPC UA 的应用，例如，ABB 主要的 DCS 800xA。他是 OPC UA 标准地址空间模型和信息模型部分的编者，在过去的几年里指导了几次 OPC UA 规范的培训课程，并做了几次演示。

Wolfgang Mahnke 拥有斯图加特大学计算机科学文凭。在凯泽斯劳滕大学工作期间，他获得了数据库和信息系统的博士学位。

## **Stefan-Helmut Leitner**

Stefan-Helmut Leitner 供职于德国的 ABB 集团研究中心工业软件技术领域。他曾参与 ABB 公司内外有关 OPC UA 不同课题的研发，例如，ANSI-C 协议栈的开发、OPC UA 的证书管理，并举办了几次培训和演示。另外，他负责编辑了 OPC UA 规范的安全模型部分。

Stefan-Helmut Leitner 拥有曼海姆教育大学的信息技术文凭。

## **Matthias Damm**

Matthias Damm 是 ascolab 的执行董事和创办人，他主要负责 OPC 咨询和认证。在过去的 10 年里，他一直积极地参与 OPC 工作，特别是符合性测试和 OPC UA 领域。

在创办 ascolab 之前，Matthias Damm 在西门子工业服务和解决方案部门的 OPC 运营中心担任经理一职。

Matthias Damm 现为 ascolab 的 OPC 基金会认证测试实验室经理。几年来，他的团队负责 OPC 基金会符合性测试工具的开发和维护。

他是 OPC UA 规范的服务部分的编者，在过去的几年里，指导过 OPC UA 培训班及做过几次演示。他参与了由 ascolab 开发并捐给了 OPC 基金会的便携式 ANSI C UA 栈的设计。他还负责被很多早期 OPC UA 产品使用的 C++ UA 服务器 SDK 的设计和开发工作，由 Unified Automation GmbH 发行。

Matthias Damm 拥有德国施魏因富特应用科技大学的电子工程文凭。

# 原 书 致 谢

出书是一个耗费时间的任务，需要投入大量的下班时间和周末。首先，我们想感谢我们的家人和朋友们的耐心和支持。Stefan-Helmut 和 Wolfgang Mahnke 感谢 ABB 的 Martin Naedele 为本书所付出的宝贵时间。Matthias Damm 感谢他在 ascolab 的同事 Gerhard Gappmeier 和 Uwe Steinkrauß。

本书的标题——OPC 统一架构——是由 OPC 基金会的 OPC UA 工作组创立的。我们获得了知识，因而基金会把写书这一任务交给了这一工作小组。我们总是很享受团队里侧重于技术方面而非政治的积极气氛。很多人都参与了工作组的会议，有些从一开始到现在，有些则只是一段时间。我们感谢他们每一个人所做出的积极配合。我们特别要感谢 OPC 基金会的 Jim Luth——这个工作组的领导，还有那些参与了无数次讨论的人，例如 Randy Armstrong 和 Tom Burke (OPC 基金会)、Jeff Harding 和 Paul Hunkar (ABB)、Karl-Heinz Deiretsbacher (Siemens)、Lee Neit-zel (Emerson)、Ayana Craven (OSIsoft)、Erik Murhpy (Matrikon)、Christian Zugfil (ascolab)、Jörg Allmendinger (Allmendinger) 和 Betsy Hawkinson (Honeywell) 等。

几位评论家帮助我们提高了本书的质量，在此要感谢 Randy Armstrong (OPC 基金会)、Karl-Heinz Deiretsbacher (Siemens)、Jens Doppelhamer (ABB)、Gerhard Gappmeier (ascolab)、Jeff Harding (ABB)、Paul Hunkar (ABB)、Emanuel Kolb (ABB)、Heiko Kozirolek (ABB)、Claude Lafond (ABB)、Jim Luth (OPC 基金会)、Uwe Steinkrauß (ascolab) 和 Roland Weiss (ABB)，感谢他们的帮助。

也非常感谢出版商的大力支持，特别是 Dorothea Glaunsinger 和 Hermann Engesser。还有那些母语为英语的伙伴们，他们大大改善了我们的英语。

最后，还要感谢在 DevCons、工作室、培训课以及其他场合就 OPC UA 进行讨论的所有人。这些讨论在很大程度上帮助我们认识和理解了 OPC UA 关键的问题，也给了我们写这本书的动力。

给本书的读者们：如果您在书中发现了错误或不明晰的表述，抑或是提供改进的建议，希望联系我们，我们将会把更正及额外的信息发表在 [www.opcuabook.com](http://www.opcuabook.com) 网站上。

Wolfgang Mahnke ([wolfgang.mahnke@de.abb.com](mailto:wolfgang.mahnke@de.abb.com))

Stefan-Helmut Leitner ([stefan.leitner@de.abb.com](mailto:stefan.leitner@de.abb.com))

Matthias Damm ([matthias.damm@ascolab.com](mailto:matthias.damm@ascolab.com))

# 目 录

译者序

原书序一

原书序二

原书前言

作者介绍

原书致谢

第 1 章 概论 .....	1
1.1 OPC 基金会 .....	1
1.2 经典 OPC .....	2
1.2.1 OPC 数据访问 .....	3
1.2.2 OPC 报警和事件 .....	4
1.2.3 OPC 历史数据访问 .....	4
1.2.4 其他 OPC 接口标准 .....	5
1.2.5 OPC XML-DA .....	5
1.3 OPC UA 的动机 .....	6
1.4 OPC UA 概述 .....	8
1.5 OPC UA 规范 .....	9
1.6 OPC UA 软件层 .....	11
1.7 进化而非革命 .....	12
1.8 小结 .....	13
1.8.1 关键信息 .....	13
1.8.2 在哪里能找到更多的信息 .....	13
1.8.3 接下来是什么 .....	14
第 2 章 信息建模：概念 .....	15
2.1 为什么要对信息建模 .....	15
2.2 节点和引用 .....	18
2.3 引用类型 .....	20
2.4 对象、变量和方法 .....	24
2.5 对象和变量类型 .....	30
2.5.1 简单对象类型 .....	30
2.5.2 简单变量类型 .....	33
2.5.3 复杂对象类型 .....	36

2.5.4 实例声明 .....	38
2.5.5 复杂变量类型 .....	41
2.5.6 建模规则 .....	41
2.5.7 复杂类型的子类型化 .....	46
2.6 数据变量和特性 .....	50
2.7 对象、变量和方法的 ModelParent .....	51
2.8 数据类型 .....	53
2.8.1 数据类型节点类别 .....	53
2.8.2 内置和简单数据类型 .....	54
2.8.3 枚举数据类型 .....	55
2.8.4 结构化数据类型 .....	55
2.8.5 特定的内置数据类型 .....	58
2.8.6 数据类型综述 .....	61
2.9 视图 .....	62
2.10 事件 .....	65
2.11 历史访问 .....	67
2.11.1 数据历史 .....	67
2.11.2 事件历史 .....	68
2.11.3 地址空间历史 .....	68
2.12 地址空间模型和信息模型 .....	71
2.13 小结 .....	73
2.13.1 关键信息 .....	73
2.13.2 在哪里能找到更多的信息 .....	73
2.13.3 接下来是什么 .....	73
<b>第3章 信息建模：实例与最佳实践 .....</b>	<b>74</b>
3.1 概述 .....	74
3.2 实例 .....	74
3.2.1 应用方案 .....	74
3.2.2 初级方案——类似于传统的 OPC .....	75
3.2.3 高级方案——提供全部功能的 OPC UA .....	76
3.3 最佳实践 .....	82
3.3.1 对象结构、信息类型和视图 .....	82
3.3.2 类型定义（对象类型和变量类型） .....	84
3.3.3 提供复杂的数据结构 .....	87
3.3.4 提供用户自定义数据类型 .....	88
3.3.5 特性 .....	89
3.3.6 方法 .....	89
3.3.7 建模规则 .....	90



3.3.8 代理对象（引用上的特性） .....	90
3.4 小结 .....	91
3.4.1 关键信息 .....	91
3.4.2 在哪里能找到更多的信息 .....	91
3.4.3 接下来是什么 .....	91
<b>第4章 标准信息模型</b> .....	<b>92</b>
4.1 概述 .....	92
4.2 处理信息模型 .....	92
4.2.1 信息模型指出了什么 .....	92
4.2.2 信息模型如何指定 .....	94
4.2.3 如何支持多信息模型 .....	95
4.3 OPC UA 基本信息模型 .....	95
4.4 能力与诊断 .....	97
4.5 数据访问 .....	98
4.6 历史数据的访问和结果返回 .....	99
4.7 状态机 .....	100
4.8 程序 .....	101
4.9 报警和条件 .....	102
4.10 特定领域信息模型 .....	102
4.10.1 概述 .....	102
4.10.2 设备信息模型 .....	103
4.11 小结 .....	104
4.11.1 关键信息 .....	104
4.11.2 在哪里能找到更多的信息 .....	105
4.11.3 接下来是什么 .....	105
<b>第5章 服务</b> .....	<b>106</b>
5.1 概述 .....	106
5.2 服务的一般概念 .....	107
5.2.1 超时处理 .....	107
5.2.2 请求与应答的消息头 .....	107
5.2.3 错误处理 .....	108
5.2.4 可扩展参数 .....	109
5.2.5 通信上下文 .....	109
5.2.6 本章有关服务描述的约定 .....	110
5.3 查找服务器 .....	111
5.3.1 查找服务器（FindServers）服务 .....	111
5.3.2 获得终端（GetEndpoints）服务 .....	112
5.3.3 注册服务（RegisterServer）服务 .....	113

5.4 客户端与服务器的连接管理 .....	113
5.4.1 安全通道的建立 .....	113
5.4.2 创建一个应用程序会话 .....	114
5.4.3 关闭一个应用程序会话 .....	116
5.4.4 取消未完成的服务请求 .....	116
5.5 在地址空间中查找信息 .....	117
5.5.1 使用服务发现地址空间 .....	117
5.5.2 在地址空间中查找信息的用例 .....	122
5.6 读写数据及元数据 .....	130
5.6.1 读取数据 .....	131
5.6.2 写入数据 .....	132
5.7 订阅数据变化和事件 .....	133
5.7.1 数据变化及事件的传递 .....	134
5.7.2 创建和管理订阅 .....	137
5.7.3 创建和管理监视项 .....	139
5.7.4 监视数据变化 .....	143
5.7.5 监视事件 .....	144
5.7.6 监视数据聚合 .....	146
5.8 调用服务器的方法 .....	147
5.9 访问历史数据和历史事件 .....	148
5.9.1 历史读取 (HistoryRead) 服务 .....	148
5.9.2 历史更新 (HistoryUpdate) 服务 .....	153
5.10 在复杂地址空间里查找信息 .....	155
5.11 修改地址空间 .....	156
5.11.1 添加节点 .....	157
5.11.2 建立节点间的引用 .....	157
5.11.3 删除节点 .....	157
5.11.4 删除节点间的引用 .....	158
5.12 小结 .....	158
5.12.1 关键信息 .....	158
5.12.2 在哪里能找到更多的信息 .....	159
5.12.3 接下来是什么 .....	159
<b>第6章 技术映射 .....</b>	<b>160</b>
6.1 概述 .....	160
6.2 数据编码 .....	161
6.2.1 OPC UA 二进制 .....	161
6.2.2 XML .....	162
6.3 安全协议 .....	162

## **XI OPC 统一架构**

6.3.1 WS-SecureConversation .....	163
6.3.2 UA-SecureConversation .....	164
6.4 传输协议 .....	165
6.4.1 UA TCP .....	166
6.4.2 SOAP/HTTP .....	166
6.5 可用映射的实现 .....	167
6.6 小结 .....	168
6.6.1 关键信息 .....	168
6.6.2 在哪里能找到更多的信息 .....	169
6.6.3 接下来是什么 .....	169
<b>第7章 安全 .....</b>	<b>170</b>
7.1 为什么安全问题如此重要 .....	170
7.2 安全的组织视角 .....	171
7.3 安全的技术视角 .....	172
7.4 确定适当的安全级别 .....	172
7.4.1 安全评估 .....	172
7.4.2 OPC UA 的安全评估 .....	174
7.5 OPC UA 的安全模型 .....	175
7.5.1 安全架构 .....	175
7.5.2 保证通信通道安全 .....	177
7.5.3 身份认证和授权 .....	182
7.5.4 安全策略和配置文件 .....	185
7.6 证书 .....	186
7.6.1 什么是证书 .....	186
7.6.2 OPC UA 证书 .....	187
7.7 OPC UA 的公钥基础设施 .....	189
7.7.1 什么是 PKI .....	189
7.7.2 信任模型 .....	190
7.7.3 证书生命周期管理 .....	192
7.7.4 可用的公钥基础设施框架 .....	198
7.7.5 工业应用的 PKI .....	201
7.8 小结 .....	208
7.8.1 关键信息 .....	208
7.8.2 在哪里能找到更多的信息 .....	209
7.8.3 接下来是什么 .....	209
<b>第8章 应用程序架构 .....</b>	<b>210</b>
8.1 简介 .....	210
8.2 架构概述 .....	210

8.3 栈 .....	211
8.3.1 接口 .....	211
8.3.2 编码层 .....	211
8.3.3 安全层 .....	211
8.3.4 传输层 .....	211
8.3.5 平台层 .....	212
8.4 软件开发工具箱 .....	212
8.4.1 UA 专用功能 .....	212
8.4.2 通用功能 .....	213
8.4.3 接口 .....	213
8.5 应用程序 .....	213
8.5.1 客户端 .....	213
8.5.2 服务器 .....	213
8.6 OPC 基金会提供的成品 .....	214
8.6.1 栈 .....	214
8.6.2 SDK .....	214
8.6.3 应用程序 .....	215
8.7 小结 .....	215
8.7.1 关键信息 .....	215
8.7.2 在哪里能找到更多的信息 .....	215
8.7.3 接下来是什么 .....	216
<b>第9章 系统架构 .....</b>	<b>217</b>
9.1 系统环境 .....	217
9.2 基本架构模式 .....	218
9.2.1 客户端—服务器 .....	218
9.2.2 链接服务器 .....	218
9.2.3 服务器至服务器通信 .....	219
9.2.4 聚合服务器 .....	219
9.3 冗余 .....	220
9.3.1 客户端冗余 .....	220
9.3.2 服务器冗余 .....	220
9.4 发现 .....	223
9.4.1 为何发现 .....	223
9.4.2 发现实体 .....	224
9.4.3 发现过程 .....	224
9.5 审计 .....	226
9.5.1 概览 .....	226
9.5.2 审计日志 .....	227

9.5.3 审计事件 .....	227
9.5.4 服务审计 .....	228
9.5.5 使用案例 .....	228
9.6 小结 .....	230
9.6.1 关键信息 .....	230
9.6.2 在哪里能找到更多的信息 .....	230
9.6.3 接下来是什么 .....	230
<b>第 10 章 从 COM OPC 到 OPC UA 的映射 .....</b>	<b>231</b>
10.1 概述 .....	231
10.2 OPC 数据访问 2.05A 和 3.0 .....	231
10.2.1 地址空间 .....	232
10.2.2 访问信息 .....	232
10.2.3 OPC XML-DA1.01 .....	233
10.3 OPC 报警和事件 1.1 .....	234
10.3.1 地址空间 .....	234
10.3.2 访问信息 .....	235
10.4 OPC 历史数据访问 .....	237
10.4.1 地址空间 .....	237
10.4.2 访问信息 .....	237
10.5 小结 .....	239
10.5.1 关键信息 .....	239
10.5.2 在哪里能找到更多的信息 .....	239
10.5.3 接下来是什么 .....	239
<b>第 11 章 迁移 .....</b>	<b>240</b>
11.1 概述 .....	240
11.2 包装器: UA 客户端访问 COM 服务器 .....	240
11.3 代理: COM 客户端访问 UA 服务器 .....	242
11.4 原生开发 .....	243
11.5 小结 .....	243
11.5.1 关键信息 .....	243
11.5.2 在哪里能找到更多的信息 .....	244
11.5.3 接下来是什么 .....	244
<b>第 12 章 协议子集 .....</b>	<b>245</b>
12.1 动机 .....	245
12.2 协议子集、一致性单元和测试用例 .....	245
12.3 服务器程序协议子集 .....	246
12.4 客户端程序协议子集 .....	247
12.5 传输协议子集 .....	247

12.6 安全协议子集 .....	247
12.7 认证流程 .....	248
12.8 小结 .....	248
12.8.1 关键信息 .....	248
12.8.2 在哪里能找到更多的信息 .....	249
12.8.3 接下来是什么 .....	249
<b>第 13 章 性能指标</b> .....	250
13.1 综述 .....	250
13.2 性能指标数值 .....	250
13.3 小结 .....	253
13.3.1 关键信息 .....	253
13.3.2 在哪里能找到更多的信息 .....	253
13.3.3 接下来是什么 .....	253
<b>第 14 章 总结与展望</b> .....	254
14.1 OPC UA 概述 .....	254
14.2 OPC UA 是否复杂 .....	254
14.2.1 OPC UA 服务是否很难被处理 .....	255
14.2.2 信息建模是否痛苦 .....	257
14.2.3 传输协议与加密, 为什么这么多 .....	259
14.2.4 实现问题 .....	260
14.2.5 现有代码的迁移 .....	261
14.2.6 小结 .....	262
14.3 展望 .....	262
<b>附录</b> .....	264
附录 A 图形表示法 .....	264
附录 B 节点类别和属性 .....	268
附录 C 基本信息模型引用 .....	270
附录 D 保留英文形式的常用术语 .....	272
附录 E 翻译为中文的常用术语 .....	278
<b>参考文献</b> .....	279

# 第 1 章 概 论

## 1.1 OPC 基金会

自 20 世纪 90 年代初以来,工业自动化领域内,基于 PC 和软件的自动化系统,特别是基于 Windows PC 上的可视化和控制应用迅速增长。在过去几年里,标准化的自动化软件开发的一个主要努力方向就是,通过大量不同总线系统、协议、接口访问自动化设备上的数据。

类似的软件应用程序问题在程序访问打印机中也存在,在过去的 DOS 时代,每个程序需要对所有支持的打印机写驱动程序。Windows 通过将打印机的支持合并到操作系统中解决了打印机驱动的问题。这一打印机驱动接口为所有应用程序提供访问打印机的服务。而提供这些打印机驱动的是打印机制造商,而不是应用软件开发商。

由于人机界面(HMI)、监控和数据采集(SCADA)软件供应商有类似的问题,1995 年,一个由 Fisher-Rosemount、Rockwell Software(罗克韦尔软件)、Opto 22、Intellution 和 Intuitive Technology 公司共同发起的工作小组成立了。该工作小组的目标是基于 Windows 系统,定义一个设备驱动访问自动化数据的即插即用标准。

很快,1996 年 8 月,OPC DA(数据访问)规范发布了。OPC 基金会是维护这个标准的非盈利组织。几乎所有提供工业自动化系统的厂商都成为了 OPC 基金会的成员。OPC 基金会能够比其他组织快得多地定义、通过、实践有关标准。取得这样的成功的一个原因是:削减主要特性和限定 API 的定义使用微软 Windows 的 COM 和 DCOM 技术。集中关注重要特性,使用基于 Windows 技术,让已经提出的与用例相关的标准得以快速通过。

通过产品开发、多厂商论证、互操作性研讨会积累的经验,OPC DA 规范的第 2 版于 1998 年推出。大量产品基于这个版本实现了标准。目前,OPC DA 第 2 版依然是 OPC 产品最重要的接口。

今天,SCADA 和 HMI 系统、流程管理和分布式控制系统(DCS)、基于 PC 的控制系统、制造执行系统(MES)必须支持 OPC 接口。在制造业和流程工业不同的工业自动化系统之间进行数据交换方面,OPC 是一个普遍被接受的标准。

经过 12 年的发展,OPC 基金会已拥有超过 450 个成员,包括全世界所有相关的自动化系统供应商。图 1-1 显示了 OPC 基金会按成员类别和地区划分的成员统

计。成员类别是根据公司成员的销售额以及最终用户成员、无选举权成员（如大学或其他组织）区分的。OPC 基金会由其成员选出的董事会进行管理。然后，董事会任命基金会的工作人员与 OPC 的首席架构师。目前，OPC 成立了一个市场管理委员会、一个技术咨询委员会和大量工作组。

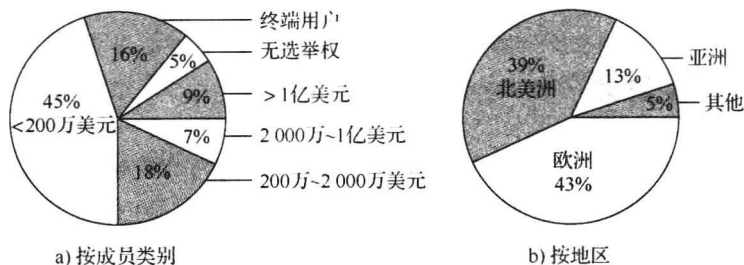


图 1-1 OPC 基金会成员统计

OPC 基金会在其产品目录中列出超过 1 500 个基于 OPC 的产品。该目录仅包含来自 OPC 成员的产品。整个 OPC 市场上，2 500 多家供应商提供了 15 000 多个使用 OPC 的产品。

由于这一伟大的成功，需要一个核查机制，以确认所有的 OPC 产品可以互操作，并确保一定的质量水平。基于这个原因，OPC 基金会各工作组除了开发新标准之外，主要工作重心就是 OPC 适应性程序。

OPC 适应性程序定义了两个认证级别。第一个级别结合了自我认证和互操作性研讨会。OPC 基金会提供所有相关 OPC 标准的一致性测试工具。用这些工具进行测试，加密后的结果会被发送到 OPC 基金会。这些测试工具覆盖了接口级别的功能测试。互操作性研讨会每年在欧洲、北美、日本举行，不同的厂商可以在那里测试产品之间的互操作性。通过自我认证的产品可以使用 Self-Tested logo，以表明其符合基本级别的 OPC 适应性。

第二个级别是独立的测试实验室产品认证。经认可的第三方测试实验室正在使用更广泛的测试覆盖验证 OPC 产品。除了通过一致性测试工具执行基本功能测试，测试实验室还将进行行为测试、负载和压力测试、互操作性测试、环境和可用性测试。对于通过第三方认证的产品，用 OPC 认证 logo 标明其高质量和 OPC 适应性。

终端用户只被鼓励购买通过 OPC 适应性测试的产品，以减少互操作性问题，并确保其 OPC 解决方案的可靠性和性能。

## 1.2 经典 OPC

近年来，OPC 基金会已定义了大量软件接口，用来标准化从过程层到管理层



的信息流。主要用例是工业自动化系统接口，如 HMI 和 SCADA 系统，从设备取得当前数据，并为管理应用提供实时历史数据和事件。

根据工业应用的不同需求，已经制定了三个主要 OPC 规范：数据访问（DA）、报警和事件（A&E）、历史数据访问（HDA）。DA 规范描述了访问过程数据的当前值。A&E 描述了基于事件的信息接口，包括过程报警确认。HDA 描述了访问历史数据的函数。所有接口提供通过地址空间浏览的方法，并提供可用数据的信息。

OPC 采用客户端/服务器（C/S）方式进行信息交换。OPC 服务器封装了过程信息来源（如设备），使信息可以通过它的接口访问。OPC 客户端连接到 OPC 服务器后，可以访问和使用它所提供的数据。使用和提供数据的应用可以是客户端，也可以是服务器。图 1-2 展示了一个客户端和服务器的典型用例。

经典 OPC 接口是基于微软的 COM 和 DCOM 技术的。

这种办法的好处是：无须定义一个网络协议或进程间通信机制，这样就减少了为不同的特定需求定义不同的 API 时的规范化工作。COM 和 DCOM 提供一个透明的机制，使客户端调用在同一进程、其他进程或在其他网络节点上运行的服务器中的 COM 对象的方法。这种技术对所有基于 PC 的 Windows 操作系统可用，降低了规范和产品的开发时间以及 OPC 的市场投放时间。这是让 OPC 取得成功的重要优势。

两个主要缺点是 OPC 的 Windows 平台依赖性和在使用 OPC 的远程通信时的 DCOM 问题。DCOM 难以配置，有很长的不可配置的超时时间，并且不能用于互联网通信。

### 1.2.1 OPC 数据访问

OPC 数据访问（DA）接口可以读、写、监测包含当前过程数据的变量。主要用例是将 PLC、DCS 和其他控制设备的实时数据迁移到 HMI 和其他显示客户端。OPC DA 是最重要的 OPC 接口。现在 99% 的采用 OPC 技术的产品都已实现。其他 OPC 接口大多作为 DA 的附加产品实现。

OPC DA 客户端明确地选择它们需要从服务器中读、写或监测的变量（OPC 项）。OPC 客户端通过创建一个 OPCServer 对象来建立一个到服务器的连接。该服务器对象提供方法通过浏览地址空间分层寻找项目和它们的属性，如数据类型和访问权限。

为了访问数据，客户端根据相同的设置（如更新时间）将 OPC 项目分组到一

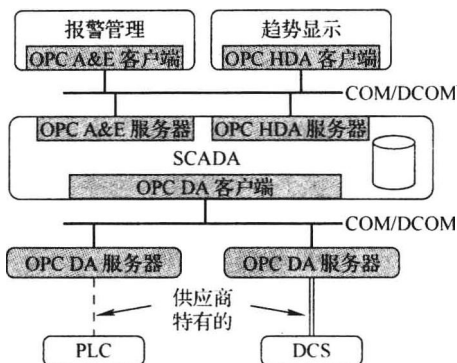


图 1-2 客户端和服务器的典型用例

个 OPCGroup 对象。图 1-3 显示了服务器中 OPC 客户端创建的不同对象。

当被添加到一个组后，项目就可以被客户端读取或写入。然而，对于客户端周期性读取数据，更好的方法是监测服务器中值的变化。客户端定义了包含感兴趣项目的组的更新速率。服务器按此更新速率循环检测值变化。每个周期后，服务器仅发送已变化的值给客户端。

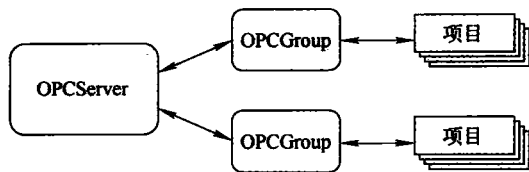


图 1-3 OPC 客户端创建对象以访问数据

OPC 提供的实时数据，可能不是一直都能访问，例如，设备通信暂时中断。经典 OPC 技术为已送达的数据提供了时间戳和质量戳处理这个问题。质量戳用来区分数据是准确（好）、不可用（坏）或未知（不确定）。

### 1.2.2 OPC 报警和事件

OPC A&E 接口可以接收事件通知和报警通知。事件是单条通知告诉客户端一个事件的发生。报警通知客户端过程状态的变化。这样的状态可以是水箱的水位。在这个例子中，超过最高水位或低于最低水位就可以发生状态改变。许多报警要求必须被确认。这种确认也是通过 OPC A&E 接口完成的。

OPC A&E 为传输来自不同事件源的过程报警和事件提供了灵活的接口。

要接收通知，OPC A&E 客户端连接到服务器，订阅通知，然后接收在服务器触发的所有通知。为了限制通知的数量，OPC 客户端可以指定某种过滤器准则。

OPC 客户端连接第一步是在 A&E 服务器创建一个 OPCEventServer（OPC 事件服务器）对象，第二步生成 OPCEventSubscription（OPC 事件订阅）对象来接收事件消息。这些事件消息的过滤器，可为每个订阅单独配置。图 1-4 显示了 OPC 客户端在服务器创建的不同对象。

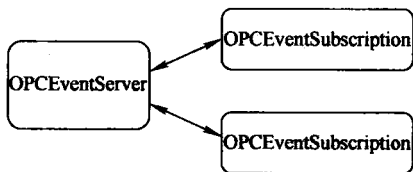


图 1-4 OPC 客户端创建对象以接收事件

与 OPC DA 相反，没有类似读数据的对指定信息的显式请求，然而所有的过程事件都提供，并且客户端可以通过设置某种过滤器准则（如事件类型、优先级或事件源）来限制事件数量。

### 1.2.3 OPC 历史数据访问

OPC DA 可以访问实时的不断变化的数据，OPC 历史数据访问（HDA）则提供了对已存储的数据的访问。从简单的串行数据记录系统，到复杂的 SCADA，历史记录能够以统一的方式被检索。

OPC 客户端通过在 HDA 服务器中创建一个 OPCHDAServer（OPCHDA 服务器）

对象进行连接。此对象提供了读取和更新历史数据的所有的接口和方法。另一个 OPC HDA Browser (OPCHDA 浏览器) 对象用来浏览 HDA 服务器的地址空间。

其主要功能是对历史数据以三种不同的方式读取。第一种机制从记录中读取原始数据, 在客户端定义一个或多个变量和它要读取原始数据的时域。服务器返回记录中指定的时间范围内的所有值, 直至达到客户端定义的最大数量。第二种机制读取一个或多个变量在指定的时间戳的值。第三种读机制为历史数据库中一个或多个变量在指定的时间域的数据计算聚合值。值始终包括相关的质量戳和时间戳。

除了读方法, OPC HDA 还定义了对历史数据库中的数据进行插入、替换和删除的方法。

#### 1.2.4 其他 OPC 接口标准

OPC 指定了几个额外的标准, 有的作为基础规范, 有的是为了满足专门的需求。基础规范是 OPC Overview 与定义所有基于 COM 的 OPC 规范的接口和行为的 OPC 通用规范。图 1-5 给出了一个所有经典 OPC 规范的概览。

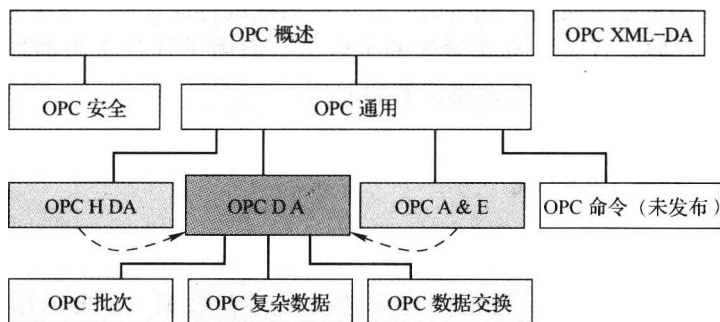


图 1-5 经典 OPC 接口标准

OPC 安全规定如何控制客户端访问服务器, 以保护敏感信息, 防止对工艺参数未经授权的修改。

OPC 复杂数据、OPC 批次和 OPC 数据交换 (DX) 是 OPC DA 的扩展。OPC 复杂数据定义了如何描述和传输复杂结构类型的值。OPC 批次通过为服务器中的客户端定义客户端行为和配置接口, 指定了 DA 服务器的数据交换。OPC 批次扩展了 DA 的批量操作特定需求。它提供了交换设备容量和当前操作情况的接口, 设备容量符合 S88.01 Physical Model (物理模型) [ISA88]。

OPC 命令定义了通过 OPC 来调用方法或执行程序的机制。这个规范从未发布, 因为它在 OPC UA 开始后才完成。但其内容和功能完全纳入到了 UA。

#### 1.2.5 OPC XML-DA

OPC XML-DA 是第一个与平台无关的 OPC 规范, 以 HTTP /SOAP 和 Web 服务技术取代 COM / DCOM。因此, 一个供应商和平台中立的通信基础设施被引进, 而

广泛接受的 OPC DA 功能得以保留。

由于典型的 Web 服务是无状态的，功能被减少到交换 OPC DA 信息的方法的一个最小子集，而不需要创建和修改通信上下文的方法。只有 8 个覆盖 OPC DA 的关键特性的方法是需要的。

这 8 项服务如下：

- GetStatus 以验证服务器的状态
- Read 读取一个或多个项目的值
- Write 写入一个或多个项目的值
- Browse 和 GetProperties 获取有关可用项目的信息
- Subscribe 创建一个项目列表的订阅
- SubscriptionPolledRefresh 用于订阅值的变化
- SubscriptionCancel 删除订阅

OPC XML-DA 是为通过互联网访问和企业集成进行设计的。由于它的平台独立性，主要实现在嵌入式系统和非微软的平台上。但由于其高资源消耗和有限的性能，它在这类型的应用上没有取得预期的成功。

### 1.3 OPC UA 的动机

第一个并且仍然是最成功的经典 OPC 标准——OPC DA，目的是作为通信驱动接口，允许标准化地读取和写入自动化设备上的当前数据。主要的用例是 HMI 和 SCADA 系统，使用一个确定的由硬件供应商提供的软件接口，访问不同供应商的不同类型自动化硬件和设备上的数据。接下来的标准 OPC A&E 和 OPC HDA 的目的同样是访问 SCADA 系统提供信息。

随着 OPC 在数以千计的产品中顺利采用，今天的 OPC 已经作为自动化系统之间的标准接口而应用在自动化金字塔的不同层次上，它甚至用在很多不是预先设计的领域，还有许多其他领域制造商希望利用这样一个 OPC 标准，但却不能使用，因为 OPC 的 COM 依赖性和它使用 DCOM 的远程访问的局限性。

OPC XML-DA 是 OPC 基金会为保留 OPC 过去成功特性的第一个标准，它使用一个供应商和平台中立的通信基础设施。但有几个原因导致只是创建过去成功的 OPC 规范对应的 Web 服务版本并不能满足新一代 OPC 的需求。其中一个原因是，相对原始 COM 版本，XML Web 服务性能欠佳。此外，通过使用不同的 XML Web Service 栈会导致互操作性的问题。

除了对平台的独立性问题，OPC 的会员公司还提出了公开复杂的数据和复杂的系统的需求，以消除经典 OPC 的局限性。

OPC UA 的诞生是希望为所有现有的基于 COM 的规范，建立一个没有损失任何功能和性能的真正替代品。此外，它必须满足能够描述复杂系统的丰富的和可扩展的建模能力，以及平台独立的系统接口的所有需求。OPC 的广泛应用要求提供从嵌入式系统跨越 SCADA 和 DCS 到 MES 和 ERP 系统的延展性。OPC UA 最重要的需求见表 1-1。

表 1-1 OPC UA 的需求

分布式系统通信	数据建模
<ul style="list-style-type: none"><li>• 可靠性（健壮性、容错性和冗余性）</li><li>• 平台独立性</li><li>• 可伸缩性</li><li>• 高性能</li><li>• 因特网和防火墙</li><li>• 安全和访问控制</li><li>• 互操作性</li></ul>	<ul style="list-style-type: none"><li>• 所有 OPC 数据的常见模型</li><li>• 面向对象</li><li>• 扩展类型系统</li><li>• 元信息</li><li>• 复杂数据和方法</li><li>• 从简单到复杂的可伸缩性模型</li><li>• 抽象基础模型</li><li>• 其他标准数据模型的基础</li></ul>

这些需求可以分成以下两类：为分布式系统之间交换信息建立沟通的需求，建模描述一个系统和现有信息的需求。

经典的 OPC 设计为设备驱动接口。今天，OPC 也作为系统接口使用，因此分布式系统之间通信的可靠性是非常重要的。由于网络通信是不可靠的，健壮性和容错性是重要的需求，包括为了高可用性的冗余功能。平台独立性和可伸缩性是必要的，以便 OPC 接口能够集成在许多运行不同系统的平台上。为了替代专用的通信，在企业内部网环境提供高性能始终是一个重要的需求。并且在互联网通信中，必须有穿过提供安全和访问控制的防火墙的能力，这是另一重要条件。首要的是，不同供应商的系统之间的互操作性仍然是最重要的需求。

经典 OPC 数据的建模是非常有限的，需要通过为所有 OPC 数据提供一个通用的、面向对象的模型来增强。这一模型必须包括一个可扩展的类型系统，以便能够提供元信息，并描述复杂的系统。具备由服务器提供描述方法和客户端调用的强大功能，使得 OPC 更灵活的和可扩展。复杂数据（Complex data）用于描述和一致性传输复杂的数据结构。提高建模能力是一个重要的需求，但同样重要的是用简单的概念来支持简单的模型。为此，必须有一个简单而抽象，但可扩展的基础模型，以便能够适应从简单到复杂的模型。

除了新一代 OPC 的功能需求，定义 OPC UA 的需求和用例的首批 40 名代表不仅由 OPC 的成员组成。其他对使用 OPC 作为信息传输机制感兴趣的标准化组织，如 IEC 和 ISA，也参与了早期的设计过程。在这个小组中，OPC 基金会定义了如何描述和传输数据，合作组织根据其信息模型来确定他们想要描述和传输什么样的

数据。

另一个重要的设计目标是能够轻松地迁移到 OPC UA，保障在成功的经典 OPC 标准上的投资，并且在大量安装了 OPC 的基础之上构建。

## 1.4 OPC UA 概述

为了达到既定目标，OPC UA 由图 1-6 所示的不同层次构成。

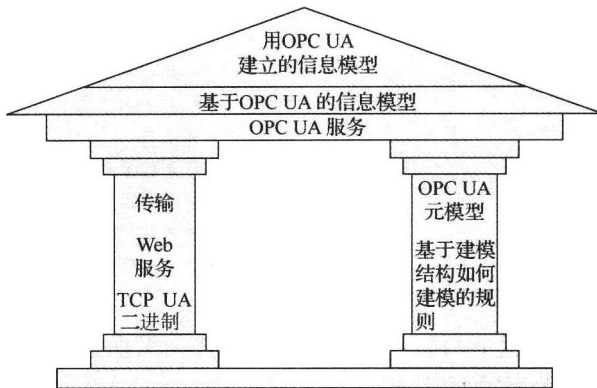


图 1-6 OPC UA 基础

OPC UA 的基础组件是传输机制和数据建模。

为了优化不同的用例，OPC UA 规定了不同的传输机制。其第一个版本为高性能企业内部网通信定义了一个优化的二进制 TCP，并为防火墙友好的互联网通信定义了一个映射，接受类似 Web 服务，XML、HTTP 等的互联网标准的通信。这两个传输都使用同一个用于 Web 服务的基础消息的安全模型。抽象的通信模型不局限于特定的协议映射，它允许将来添加新的协议。传输机制将在第 6 章进行更详细的介绍。

数据模型定义了提供 OPC UA 的信息模型的规则和基础构件，它也定义了地址空间的入口和建立类型层次的基本类型，这个基础可以被建立在抽象模型概念上的信息模型扩展。此外，它还定义了一些增强的概念，如在不同的信息模型中描述状态机。第 2 章中描述了信息建模的基础知识。第 3 章介绍了一个例子和最佳实践。

UA 服务是作为信息模型的提供者——服务器和信息模型的用户——客户端之间的接口。服务是以抽象的方式定义的，它使用传输机制，在客户端和服务器之间交换数据。

这种 OPC UA 的基本概念使得 OPC UA 的客户端可以访问最小的一块数据，而不需要了解复杂的系统公开出来的整个模型。OPC UA 的客户端也可以理解专有模

型,这种模型可以使用为特定领域和用例定义的增强功能。图 1-7 显示了 OPC、其他组织或供应商定义的信息模型的不同层次。

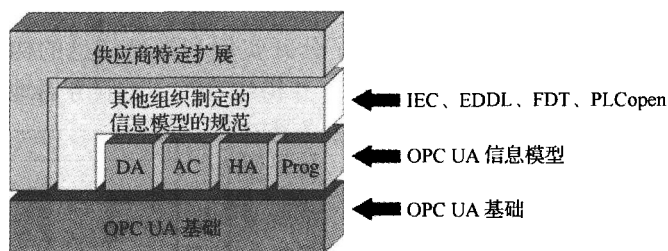


图 1-7 OPC UA 分层架构

为了覆盖传统 OPC 成功的功能, OPC UA 在基础规范之上定义了过程信息领域的信息模型。DA 定义了自动化数据方面的特定扩展,例如模拟和离散数据建模,以及服务质量公开。DA 的其他所有功能都已经被基础规范覆盖。报警和状态 (AC) 定义了一个处理报警管理和状态监视的高级模型。历史访问 (HA) 定义了访问历史数据和历史事件的机制。程序 (Prog) 制定了启动、操作、监视程序执行的机制。

其他组织可以在 UA 基础上或者在 OPC 信息模型上构造他们的模型,通过 OPC UA 公开特定的信息,已经开始进行 OPC UA 映射的标准是现场设备集成 (FDI),它合并了用来描述、配置和监视设备的电子设备描述语言 (EDDL) 和现场设备工具 (FDT)。另一个例子是标准化的 PLC 编程语言 PLCopen。

附加的供应商特定信息模型可以通过直接使用 UA 基础、使用 OPC 模型或者其他基于 OPC UA 的信息模型来定义。

## 1.5 OPC UA 规范

OPC UA 规范可划分为不同的部分,这也是 IEC 标准化的需要, OPC UA 将作为 IEC 62541 标准。图 1-8 显示了所有划分到定义 OPC UA 基础的核心规范和主要在 OPC UA 信息模型中指定的访问类型特定部分规范的一个概览。

前两个部分不是规范的。概念部分 [UA 第 1 部分] 给出了一个有关 OPC UA 的概述, UA 第 2 部分介

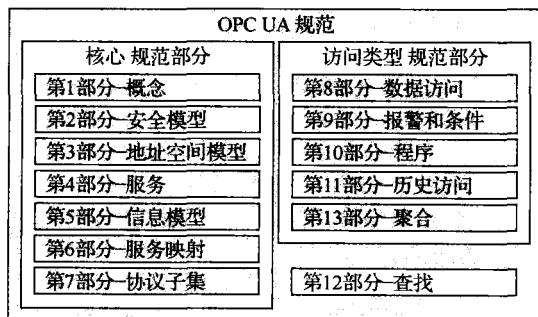


图 1-8 OPC UA 规范

绍了 OPC UA 的安全要求和安全模型。

对于理解如何建模和访问信息，最重要的是第 3 和第 4 部分，这两个规范是设计和开发 OPC UA 应用的关键文档。

地址空间模型 [UA 第 3 部分] 指定了公开实例和类型信息的构件，从而可以用 OPC UA 元模型来描述和公开信息模型，并构建 OPC UA 服务器地址空间。

定义在 [UA 第 4 部分] 的抽象 UA 服务描述了 UA 客户端和 UA 服务器之间可能的交互。客户端使用服务来查找和访问服务器提供的信息。该服务是抽象的，因为它们定义的是 UA 应用之间交换的信息，但不是在传输线路上的具体行为，也不是应用软件使用的 API 的具体行为。图 1-9 显示了 OPC UA 的分层通信架构。

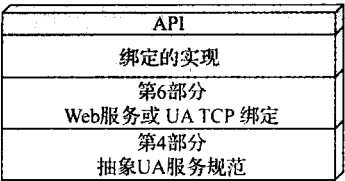


图 1-9 OPC UA 分层通信架构

[UA 第 5 部分] 中的基本信息模型为所有使用 OPC UA 的信息模型提供框架。它定义了以下内容：

- 地址空间的入口点被客户端用来浏览 OPC UA 服务器中的实例和类型
- 构建不同类型层次结构所用的基础类型
- 内置的、但可扩展的类型，如对象类型和数据类型
- 该 Server Object（服务器对象）提供容量和诊断信息

UA 服务到消息的映射，应用到消息的安全机制，消息的具体线路传输在 [UA 第 6 部分] 中定义。只有实现 UA 栈才需要完全理解此规范。由于 OPC 基金会提供了合适的 UA 栈，典型的 UA 应用程序架构师和程序员并不需要阅读本规范。

协议子集在 [UA 第 7 部分] 中定义了 OPC UA 功能的一个有用子集。该子集必须由 UA 应用程序完全实现，以确保子集的互操作性。该规范在两个级别定义了该子集。第一个级别是定义一个小功能集的一致性单元，就是始终一起使用，可以当做一个单元用兼容性测试工具进行测试。第二个级别是一个一致性单元列表组成的协议子集。在 OPC UA 产品认证中，一个协议子集必须完全实现，并且被作为一个完整集合测试。在客户端和服务器连接创建的时候，交换支持并使用的协议子集列表。从而允许程序确定对方是否支持自己需要的功能。

在 [UA 第 8 部分] 中，DA 信息模型定义了如何表示和使用自动化数据以及指定类似工程单位一样的特性。

在 [UA 第 9 部分] 中，AC 信息模型指定了过程报警、监视指定状态机的状态和事件类型。

在 [UA 第 10 部分] 中，程序信息模型定义了执行、处理和监视程序的一个基础状态机。



在 [UA 第 11 部分] 中, HA 信息模型指定了历史访问服务的使用, 以及怎样呈现数据和事件历史的配置信息。

在 [UA 第 12 部分] 中, 定义了怎样在网络中查找服务器, 以及客户端怎样得到必需的信息, 以便建立一个连接到某服务器。

在 [UA 第 13 部分] 中, 指定了用来从原始数据样本计算聚合值, 以及用于历史访问和当前值的监测的聚合方式。

## 1.6 OPC UA 软件层

OPC UA 使用类似经典 OPC 的客户端/服务器概念。一个希望对其他应用暴露自己的信息的应用程序被称为 UA 服务器。而一个想要使用其他应用程序的信息的应用程序被称为 UA 客户端。但是, 期望与经典 OPC 相比, 更多的应用是在一个应用程序中包含 UA 服务器和 UA 客户端。其中一个原因是, 越来越多的 UA 服务器将被直接集成在设备中, 同时实现一个 UA 客户端使设备到设备的通信变得可行。另一个原因是, OPC UA 用作配置接口, UA 客户端同时也是一个可以通过 OPC UA 进行配置的 UA 服务器。

一个典型的 OPC UA 的应用是由图 1-10 所示的三个软件层次组成的。整个完整的软件栈可以使用 C/C++、.NET 或 Java 实现。OPC UA 不限定只使用这些编程语言和开发平台, 但目前只有这些环境下的 OPC 基金会的 UA 栈的实现可以交付使用。

一个 OPC UA 应用程序是一个要公开或使用 OPC UA 数据的系统。它包含该应用程序指定的功能, 以及通过使用 OPC UA 栈和 OPC UA 的软件开发工具包 (SDK), 从该功能到 OPC UA 的映射。

实现 OPC UA 公共功能的客户端或服务器 SDK 是应用层的一部分, 因为 UA 栈只实现通信通道。OPC UA SDK 减少了开发工作, 并促进了 OPC UA 应用更快速的互操作性。

一个 OPC UA 栈实现了 [UA 第 6 部分] 定义的不同 OPC UA 传输映射。该栈是用来调用跨进程或网络边界的 UA 服务。OPC UA 定义了三个栈层并为每层定义了不同的配置。消息编码层定义一个二进制和一个 XML 格式的服务参数序列化方式。消息安全层指定通过使用 Web 服务安全标准或 UA 的二进制版 Web 服务标准来保证消息的保密。消息传输层定义了使用的网络协议, 它可以是 UA TCP、HTTP

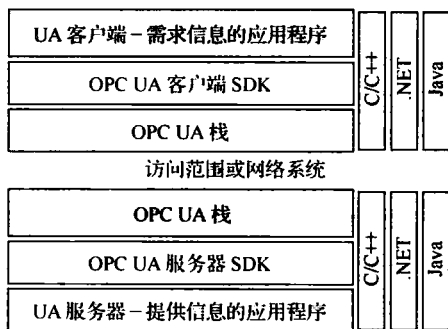


图 1-10 OPC UA 软件层次

或 Web 服务使用的 SOAP。图 1-11 显示了不同的 UA 通信栈层。UA 栈中各层的实现和该应用程序的 API 不是 OPC UA 规范的一部分。UA 栈提供与语言无关的 API 给 UA 客户端和服务端应用程序，但是服务及其参数是类似的，而且基于 [UA 第 4 部分] 中的抽象服务定义。

随着 ANSI（美国国家标准学会）C/C++、.NET 和 Java 的实现，主要的开发环境和编程语言已被 OPC 基金会开发和维护的 UA 栈覆盖了。

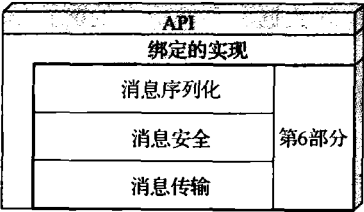


图 1-11 UA 通信栈层在 [UA 第 6 部分] 中定义

1.7 进化而非革命

OPC UA 更加灵活，而且功能比所有经典 OPC 规范加在一起还要多得多，但它整合了现有的 OPC 规范的所有成功的观念，解决了现有标准中的已知问题，并增加了对大量新增用例的标准化。

一个重要的设计目标是允许从经典 OPC 到 OPC UA 的轻松迁移。基于这个原因，大多数经典 OPC 的功能可以在 OPC UA 中看到，有时只是使用稍微不同的术语。不可能通过经典 OPC 接口公开 UA 的所有功能，但是映射经典 OPC 的功能到 OPC UA 是没有问题的。

OPC UA 允许一个简单的映射，并提供迁移策略，集成基于前一个 OPC 标准的 OPC 产品。该迁移策略的一部分，甚至不要求改变现有产品。OPC 基金会提供的包装器和代理能够将不同的经典 OPC 接口翻译成 OPC UA 的，反之亦然。迁移战略的第一层让供应商可以用来使现有产品支持 OPC UA。

到 OPC UA 的第二层次的迁移是直接将 OPC UA 集成到现有产品，不增加 OPC UA 的特定功能。这一步并不需要改变系统之间和 OPC 通信组件之间的接口。如果一个系统的现有接口并没有改变，集成新的组件是非常容易的。相对于包装器或代理的优势是通过移除附加软件层达到更好的使用性能，以及使用较少的配置和工程工作量。直接整合将使它更容易消除包装器和代理方式潜在的限制，并允许以迭代开发的方式一步一步地加入 OPC UA 的功能。

第三层次可能需要改变产品的内部接口，支持 OPC UA 中所有感兴趣的功能。OPC UA 将允许系统通过一个标准的方式使产品功能可用，这些功能无法以不包含 OPC UA 提供的新选项的方式公开。

对于终端用户，重要的是有包装器和代理可以为大量已安装的基于经典 OPC 的产品提供到 OPC UA 的迁移。终端用户可以通过安装包装器和代理，使经典 OPC 通过隧道穿越防火墙，包括在互联网上的安全传输和认证访问，因此它们简单地

现有的工业解决方案增加了价值。

这些组件只是第一步。而对于供应商显得更为重要的是 OPC UA 在标准应用的各个领域提供抽象、模块化和简单的基础概念，允许现有的 OPC 功能容易地迁移到 OPC UA。扩展这个基础是一个非常强大的概念，使供应商可以通过 OPC UA 公开越来越多的系统功能，这引导了一个迭代式开发和改进的过程。

## 1.8 小结

### 1.8.1 关键信息

OPC 基金会提供工业自动化的数据交换标准。这包括非常成功的针对当前数据的 OPC DA 规范，以及针对报警和事件的 OPC A&E 与针对历史数据的 OPC HDA。所有这些规范都基于过时的 COM/DCOM 技术。向最先进的技术迈出的第一步是 OPC XML-DA，它只使用 XML 作为数据传输的格式，因此不符合典型的 OPC 应用的性能要求。

通过 OPC XML-DA 的经验教训，OPC 基金会创造了一个新标准，称为统一架构（UA）。在这里，传输既可以利用防火墙友好的标准的 Web 服务，如 SOAP 和 HTTP，也可以使用优化的高性能通信的二进制 TCP。OPC UA 为应用程序之间提供了互操作的、平台独立的、高性能的、可扩展的、安全和可靠的通信。从微软的 COM/DCOM 切换到最先进的平台独立的传输协议，允许 OPC UA 的应用程序运行在智能设备和控制器上，同时也运行在 DCS 和 SCADA 系统上，以及达到企业级水平运行在 MES 和 ERP 系统之上。相对于经典 OPC 应用，这大大扩大了使用范围。

除了通信，OPC UA 的第二大成果是信息建模。通过在同一地址空间公开当前数据、事件通知以及两者的历史，OPC UA 使不同的经典 OPC 规范的功能得到了统一。另外，它采用面向对象的概念，提供了一个丰富的、可扩展的信息模型。允许应用程序的元数据以及复杂的数据被公开出来。可扩展的机制允许其他组织可以定义使用 OPC UA 通信基础设施的标准信息模型，从而使焦点集中在要公开信息的标准化上。已经有了定义标准信息模型的举措，例如，描述设备的 FDI 通过映射 EDDL 和 FDT 到 OPC UA，或通过 PLCopen 的 PLC 编程语言。像传输能力一样，OPC UA 的建模能力也可以很好地进行规模扩展。可以使提供信息保持简单，类似于经典 OPC，但也可以用了一个类型系统来丰富该信息，从而在很多应用场景下提供更多非常有用的信息。

### 1.8.2 在哪里能找到更多的信息

OPC 的信息可以在经典 OPC 基金会网站（[www.opcfoundation.org](http://www.opcfoundation.org)）找到。还有一本经典的 OPC [IL06] 书籍。几个供应商提供经典 OPC 的产品和工具包。可以在 [www.opcconnect.com](http://www.opcconnect.com) 找到一个列表，在这里会发现 OPC 的概要信息，包括

OPC UA。其他链接和信息可在本书的网站 [www.opcuabook.com](http://www.opcuabook.com) 找到。

关于 OPC UA 的一般信息可在 OPC 基金会网站的 ua 部分 ([www.opcfoundation.org/ua](http://www.opcfoundation.org/ua)) 中找到。该网站还提供了 OPC UA 规范的访问, 其中 [UA 第 4 部分] 和 [UA 第 6 部分] 关注数据传输, 而 [UA 第 3 部分] 和 [UA 第 5 部分] 关注建模信息。

### **1.8.3 接下来是什么**

在接下来的三章中, 将集中讨论如何使用 OPC UA 提供信息, 然后才是关于如何访问这些数据的细节 (从第 5 章开始)。

在第 2 章和第 3 章中, 介绍了基础建模概念, 并给出了这些概念的例子和最佳实践。标准的信息模型在第 4 章中描述。这包括如何建立模型的介绍: OPC 定义的模型的描述和其他组织定义的标准信息模型的概述。

在第 5 章中, 将介绍 OPC UA 的抽象服务, 第 6 章描述了这些服务到具体技术如 Web 服务的映射。

## 第 2 章 信息建模：概念

### 2.1 为什么要对信息建模

OPC UA 的基础是数据传输和信息建模。相对于传统的 OPC，数据传输已经是艺术级的、平台独立的、安全的、技术可靠的技术了，同时信息建模的能力也获得了大幅度提高。传统 OPC 只能提供纯数据，例如，由温度传感器测量出来的温度。要了解已提供的数据是什么语义，可以利用的信息就只是测点的名称和一些类似测量值的工程单位的基本信息。OPC UA 提供更有效的展示数据语义的可能性。除了由传统 OPC 提供的数据，它允许对外展示这样的信息：测量的温度是由一个特定类型的传感器设备提供的，并允许暴露该种设备支持的类型层次。因此，OPC UA 的客户端可以获得它们在处理不同地方的同类设备的信息。通过暴露更多的语义，OPC UA 的服务器允许客户端通过解释所提供的数据的语义来处理非常复杂的任务。它包括自动集成由 OPC UA 服务器提供的数据，就像在一个普通的 OPC UA 客户端上设计 OPC UA 服务器一样。

基本的 OPC UA 规范只提供模型信息的基础设施，这些信息可以由供应商建模。当然，这会导致对类似的信息有不同的建模方式，从而给 OPC UA 客户端造成困难。为了避免这种情况，OPC UA 规范提供了基于 OPC UA 规范定义信息模型的可能性。OPC 基金会已经开始创建这些规范。例如，已经尝试定义一个基本模型，用来展示 OPC UA 的设备信息和设备类型 [UA 设备]。供应商将使用此基础模型，并且使用与设备相关的供应商特定信息来扩展这个模型。客户可以用相同的方法访问不同的、供应商特定的 OPC UA 服务器提供的设备信息，因为它们是在一个类似的、使用相同基本模型的方式暴露出来的。此外，因为都使用相同的基本模型，供应商可以把通过 OPC UA 提供数据的第三方设备轻松地无缝整合到自己的服务器中。当然这并不只适用于设备模型，也适用于其他方案。例如，通过在 OPC UA 暴露 ISA 95 [ISA95] 模型提供数据给 MES 或 ERP 系统。

OPC UA 信息建模的基础原则如下：

- 使用面向对象的技术，包括类型层次结构和继承。类型化的实例允许客户端以相同的方式处理同一个类型的所有实例。类型层次结构允许客户端使用基础类型、忽略更多的特定信息。
- 类型信息对外暴露，并且能够以访问实例一样的方式访问。类型信息是由

OPC UA 服务器提供的，并且可用访问实例一样的机制访问。这类似于关系数据库系统的信息概要，数据库表中的信息在数据库表中被管理，并且通过普通的 SQL 语句访问 [ISO08b]。

- **全网状的节点网络，允许信息以各种方式连接。** OPC UA 允许支持几种不同的层次结构，暴露不同的语义，并且这些层次结构的节点可以互相引用。因此，同样的信息可能以不同的方式暴露，可以根据不同的使用情况，提供不同的路径和方法来组织同一台服务器的信息。
- **类型层次结构以及节点间引用类型的扩展性。** OPC UA 可以通过几种方法扩展信息建模。通过定义子类型，它允许采用一些方法来扩展 OPC UA 的功能。例如指定附加的引用类型、定义节点和方法之间的关系。
- **没有限制如何对信息建模，以便为所提供的数据建立适当的模型。** OPC UA 服务器的目标是：系统已包含了丰富的信息模型，可以通过 OPC UA “原生”地暴露模型，而不是将该模型映射到一个不同的模型中。
- **OPC UA 的信息建模工作总是在服务器侧完成的。** OPC UA 的信息模型始终存在于 OPC UA 的服务器，而不是在客户端。它们可以被 OPC UA 客户端访问和修改，并且 OPC UA 服务器也可以作为客户端访问其他 OPC UA 服务器。但在一般情况下，OPC UA 的客户端不需要有一个集成的 OPC UA 信息模型，它并不一定要提供这些信息给 OPC UA 服务器。

下面通过一个例子来看看 OPC UA 的建模能力。设备供应商提供了一个传感器，如图 2-1 所示。该装置具有一定的配置参数和一些测量值，这些测量值可能因为配置不同而不同。一个使用之前提到的基础设备模型的 OPC UA 服务器提供了这些信息。

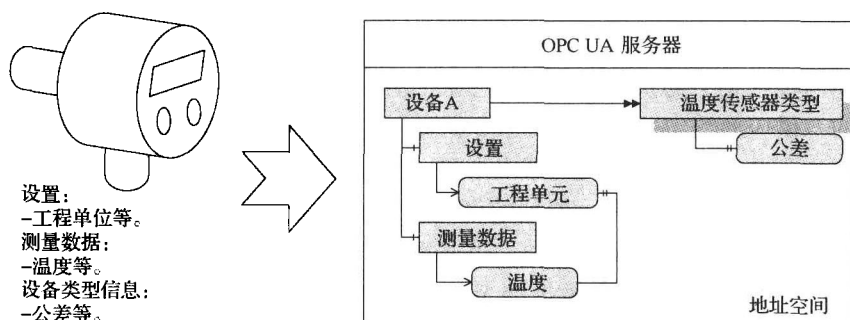


图 2-1 温度传感器和在 OPC UA 中提供的数据

要理解这个例子，重要的是找到 OPC UA 服务器正在哪里运行，以及它提供了什么额外的信息。在最简单的情况下，OPC UA 的服务器将直接运行在设备上，对于一个像温度传感器这样简单的设备，OPC UA 的服务器更可能是运行在控制器或

控制器上层的 PC 上。任何 OPC UA 的客户端可以提供相应的用户界面展示，并通过服务器提供的设备模型配置设备。该模型可以包括事件和历史数据。客户端可以使用一个通用的用户界面（显示参数列表）或使用一个专门的界面呈现该设备和主要参数的图形。这些特定的用户界面，可以基于类型信息创建，它们只需要对每个类型实现或配置一次，就可以被该类型或其子类型的每个实例使用。因此，这样的用户界面可以针对设备模型的基本类型定义，不暴露任何特定供应商的扩展。也可以被定义为针对特定供应商的类型，以适应供应商特定的设备及其扩展。整合设备数据的最常见的用途是，在一个 DCS 上汇总这些数据并提供给 DCS 的客户端。在这种情况下，DCS 应作为一个 OPC UA 的客户端接收数据，并作为 OPC UA 服务器对外提供数据。任何 OPC UA 的客户端可以通过 DCS 设备访问数据，而不损失任何功能。服务器包含多个设备的场景，说明了基于类型信息编程的能力。例如，一个图形中可以多次使用同样的图形元素，以显示工厂的流程。图 2-2 概括了上述 OPC UA 客户端的有关用例。

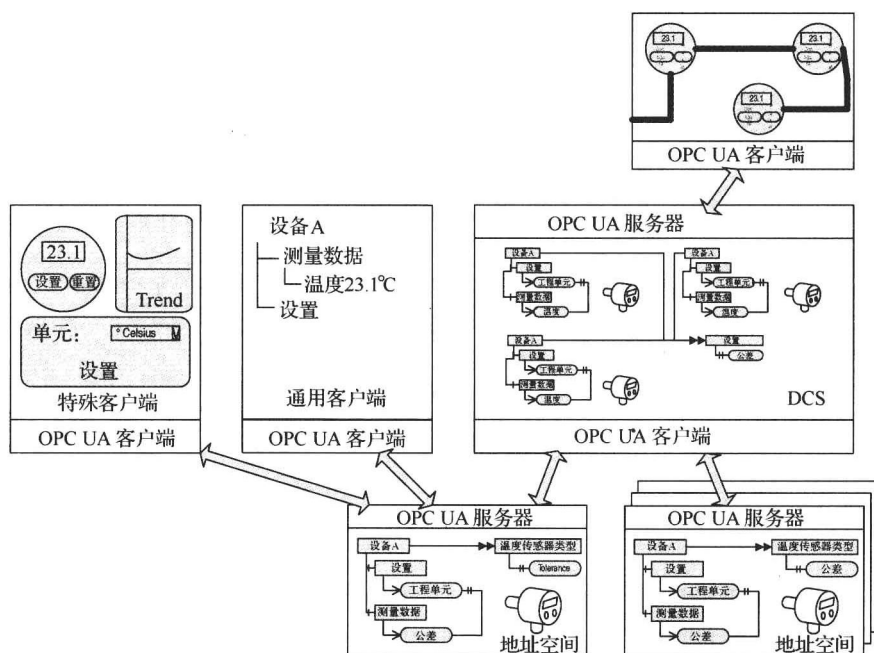


图 2-2 如何使用 OPC UA 访问设备数据的不同场景

本章概述了在 OPC UA 中如何对信息建模。以下各节描述了用于建模的 OPC UA 的概念。从节点以及节点间引用的基本概念开始，解释了 OPC UA 的对象模型，包括变量、方法和事件，并将在接下来的一章详细解释它们的差别，并提供了一个关于怎样对信息建模的详细例子和一些最好的实践。

2.2 节点和引用

OPC UA 建模的基本概念是节点以及节点之间的引用。节点可以根据不同的用途归属于不同的节点类别（NodeClass）。一些节点代表实例，另一些代表类型，等等。属性（Attribute）被用来描述节点，根据节点类别，一个节点可以有不同的属性集。图 2-3 给出了一个例子。节点 1、节点 2 和节点 3，以及所有包含的属性被几个引用（引用 1 ~ 6）连接起来。

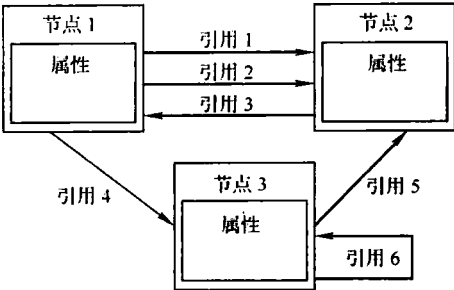


图 2-3 节点和节点间的引用

以下几节介绍了 OPC UA 的不同 NodeClass。一个节点的属性取决于其 NodeClass。不过，每个节点也有一些通用属性。在表 2-1 中，对这些属性进行了小结。

表 2-1 通用属性

属 性	数 据 类 型	说 明
NodeId	NodeId	在 OPC UA 服务器内惟一确定一个节点，并且在 OPC UA 服务中定位该节点
NodeClass	NodeClass	一个定义 NodeClass 的枚举，例如对象（Object）或方法（Method）
BrowseName	QualifiedName	浏览 OPC UA 服务器时定义节点。它是非本地化的
DisplayName	LocalizedText	包含节点的名字，用来在用户接口中显示名字。因此，它是本地化的
Description	LocalizedText	这个可选属性包含节点的本地化描述
WriteMask	UInt32	这是可选的，指示哪个节点属性是可写的，即可被 OPC UA 客户端修改
UserWriteMask	UInt32	这是可选的，指示哪个节点属性可以被当前连接到服务器上的用户修改

NodeId 在服务器中惟一标识一个节点。NodeId 用来引用节点，是定位和在服务间交换信息的最重要概念。浏览或查询地址空间时，服务器返回 NodeId，然后客户端在服务调用时使用 NodeId 来定位节点。一个节点可以有一个规范的 NodeId 和多个可选的 NodeId 用来定位节点。规范的 NodeId 可以通过读取节点的 NodeId 属性来获得，即使该节点是通过可选的 NodeId 访问的。NodeId 包含一个命名空间，允许不同的命名机构确定惟一的名字。命名机构可以是一个组织、供应商或系统。关于 NodeId 及其数据类型的细节参见 2.8.5 节。

浏览名称（BrowseName）仅用于浏览目的，不应该用于显示节点的名称。浏



览名称有一个性质特殊的意义（见 2.6 节），并且提供类型信息（见 2.5.4 节），以方便编程。类似于 NodeId，浏览名称是一个包含一个命名空间（Namespace）和非本地化字符串的结构（见 2.8.5 节）。

DisplayName 和 Description 是本地化的。2.8.5 节介绍了有关本地化和 LocalizedText 数据类型的细节。

从理论上讲，引用（Reference）描述了两个确定节点之间的关系。因此，源节点、目标节点、引用的语义（即 ReferenceType，见 2.3 节）和引用的方向惟一确定一个引用。

在实践中，一个服务器可能只在一个方向上暴露引用，引用可能指向另一个 OPC UA 服务器的节点或一个不存在的节点。因此，可以这样去理解一个引用：在一个节点中的指针，通过保存其他节点的 Id 来指向另一个节点。在图 2-4 中，显示了对引用的这种观点。

这样的引用指针包含被引用节点的 NodeId，管理被引用节点的 OPC UA 服务器，定义了引用语义的引用类型（见 2.3 节）和引用的方向。引用是区分对称和非对称引用的。非对称引用，例如，“有父节点的”是一个方向，“是子节点”是另一个方向；而对称引用在两个方向上有同样的语义，如“是兄弟节点”。引用的方向在第一种情况下是重要的，在第二种情况下则并不重要。

把引用看做一个指向其他节点的指针，可以帮助理解对 OPC UA 引用定义的约束或缺失的约束。虽然引用连接两个节点，服务器有可能只对外暴露一个方向，例如在图 2-4 中，只有节点 1 到节点 2 的指针。如果只有一个方向被暴露，则引用被称为单向的，否则就被称为双向的。指针可引用不存在（或不再存在）于服务器的节点，也可以引用其他服务器上的节点，该节点在某一时间点处于不可用状态或根本不再存在了。客户端必须能够预期到它们也许能够在一个方向浏览引用，但不能通过反方向浏览，或者被引用的节点是不存在的。

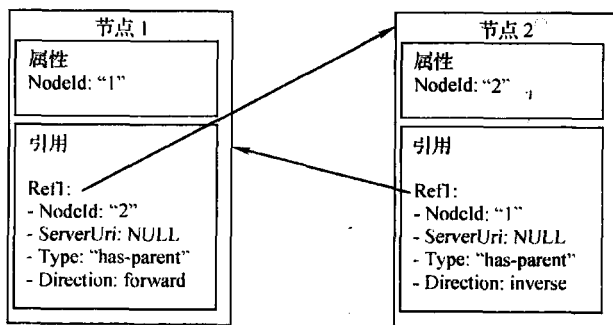


图 2-4 引用类似于指向节点的指针

⊖ 此处原文为节点 1，但译者认为这是原文的一处纰漏，因此在译文中改为节点 2。——译者注

引用是不排序的，即两次请求某节点的引用可能返回两个不同方式排序的引用集合。不过也有引用类型定义引用的顺序，如 `HasOrderedComponent` [UA 第 3 部分]。服务器始终以相同的顺序返回这些引用。

### 什么是允许的，什么是不允许的？

节点的属性集是由 OPC UA 规范定义的，而且不能被扩展。如果描述一个节点的其他信息是必要的，必须用特性（Property，见 2.6 节）来替代（译者注：由于 OPC UA 中同时出现了 `Attribute` 和 `Property` 两个常译为“属性”的术语，我们把前者译为“属性”，而后者译为“特性”）。

由 `WriteMask` 表示可写的属性集必须相同或大于 `UserWriteMask` 表示的可写属性集，因为 `WriteMask` 定义了任何用户可以修改的属性。一般来说，引用的完整性要求非常低。引用可指向同一服务器不存在的节点，也可以指向其他服务器上的节点，即使客户端不能访问指向的服务器。引用可能只在一个方向上暴露而且可能会导致循环引用，客户端必须能够处理这些不准确的数据。具体的引用可能会更严格限制或者定义更高的约束（见 2.3 节）。

引用不包含任何属性或特性。在某些情况下，应增加附加信息来表示两个节点的关系，必须通过添加和两个节点都连接的代理（Proxy）节点而不是使用单个引用（关于怎样使用代理节点参见 3.3.8 节）。

## 2.3 引用类型

引用是两个节点之间的连接。引用不能直接访问，只能间接地通过浏览节点访问。引用并没有表示为节点，不能包含任何属性或特性。引用用于暴露节点如何连接的不同语义。OPC UA 使用引用类型（`ReferenceType`）来暴露引用的语义。而引用都具有一种类型，因此是有明确语义的。OPC UA 规范定义了一套引用类型，其中一些是用于非常基本的地方，例如，暴露类型层次结构，但引用类型的概念是一个可扩展的概念，也就是说，OPC UA 服务器可以定义自己的引用类型暴露特定的引用语义。为了组织引用类型，它们在类型层次结构中被管理。

虽然引用不是节点，也没有属性，但是引用类型在地址空间中被当作节点。这使得客户端可以通过访问 OPC UA 服务器地址空间中的节点获得一个 OPC UA 服务器使用的引用信息。在表 2-2 中，对用来描述引用类型的属性进行了总结。

表 2-2 引用类型的附加属性

属 性	数 据 类 型	描 述
包含所有在表 2-1 中定义的通用属性		
IsAbstract	布尔	指定引用类型是否可用于引用，或只用于组织引用类型层次结构
Symmetric	布尔	指示引用是否对称，即是正反方向的意思是否相同
InverseName	LocalizedText	这个可选属性指定引用在反方向上的语义。它只能适用于非对称的引用，并且如果引用类型不是抽象，则必须提供这个属性

除了这些属性，表 2-1 描述的所有节点的通用属性都对引用类型有效。其中一些属性引用类型使用时有一些其他约束。一个引用类型的 BrowseName 必须在一个 OPC UA 服务器中是惟一的，以避免同名的引用类型语义不同而导致混乱。DisplayName 必须包含 BrowseName 的本地化表示。BrowseName 和 DisplayName 确定了引用类型的正向语义，而 InverseName 定义了反向的语义。

引用在一个引用类型层次结构中被管理，这允许用特定的类型对现有引用类型进行特定化。图 2-5 展示了由 OPC UA 规范定义的基本引用类型层次结构。它采用了附录 A 所述的 OPC UA 符号来形象化 OPC UA 的相关信息，附录 C 是基本 OPC UA 规范定义的完整引用类型层次结构。

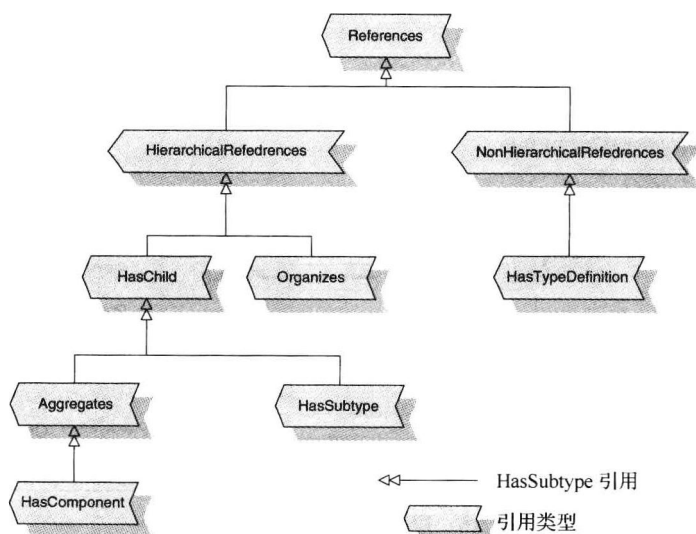


图 2-5 基本引用类型层次结构

使用斜体字母的引用类型是抽象的，只用于组织和过滤。第 5 章介绍了可以应用过滤的地方。如果过滤器包括所有的引用，则使用“References”引用类型。引用分

为分层次化引用和非层次化引用两个子类型。当需要建立层次模型时，应当使用层次化引用；其他一些情况应使用非层次化引用，如暴露不同层次结构间的关系。例如，非层次化的 HasTypeDefinition 从一个实例引用到类型层次中的类型。客户端将根据这个来使用引用，例如在一个树形控件显示层次化引用，并过滤掉非层次化引用。

为了简化对 ReferenceTypes 过滤，ReferenceType 层次结构只支持单一继承，即每个引用类型只能有一个父类型<sup>①</sup>。这保证每个引用类型<sup>②</sup>要么是一个层次化引用类型，要么是非层次化引用类型，而不可能同时属于两者。

在图 2-5 中显示了一些层次化引用的子类型。组织（Organizes）引用类型仅仅表示两个节点需要通过层次的方式连接起来，不定义任何附加的语义。一个使用这个引用类型的例子是，当把文件系统映射到 OPC UA，文件夹可以被当作节点，使用 Organizes 引用来引用其子文件夹。Organizes 引用或任何层次化引用并不保证不形成环<sup>③</sup>。在图 2-6 中，给出了这样的实例。在这个例子中，存在两个层次结构。一个组织设备，一个包含

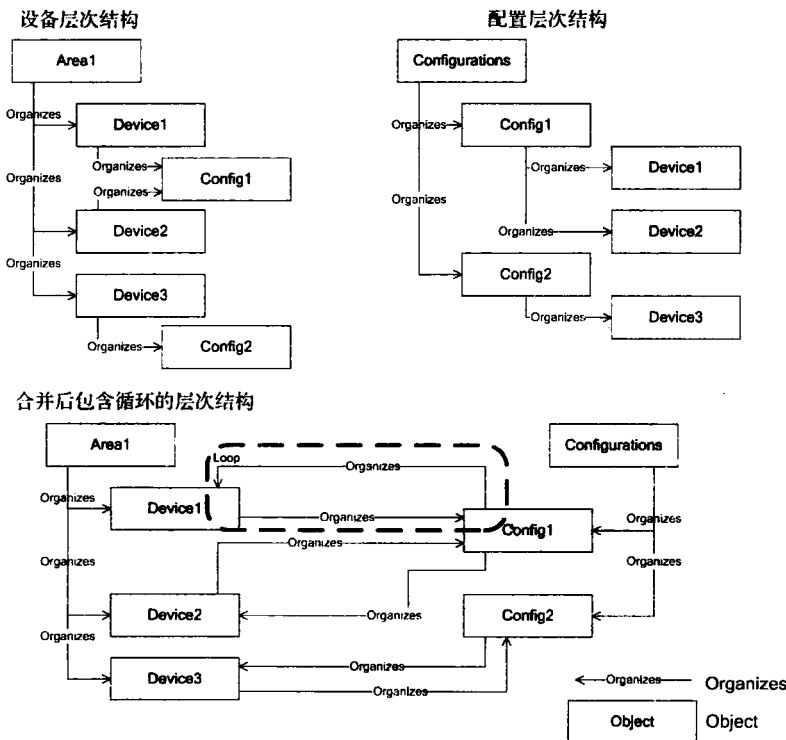


图 2-6 层次结构中的循环示例

- ① 当然，基本的 References 引用类型没有父类型，因为它是该层次结构的根。
- ② 另一个保证这个约束的需求是，只有 HierarchicalReferences 和 NonHierarchicalReferences 可以直接继承 References。
- ③ 不过，不允许节点直接通过 Hierarchical References 引用自己。

配置信息。这些设备引用一些包含它们配置信息的配置对象，但几个设备可以使用同一个配置，如 Device1 和 Device2，都使用配置 Config1。在配置层次结构里，使用某一配置的设备被列于配置的下方。如果在一个合并后的视图里看这些层次结构，则可以看到在该层次结构中形成了循环。这个例子也表明，组织（Organiz）并不总是层次化方式引用节点的最佳选择。在这个例子中，已经应用了一些更特定的引用类型。我们将在介绍更多引用类型后再回头看看这个例子。

抽象的 HasChild 引用，在仅仅沿着它的子类型前进时不允许产生循环。因此 HasChild 引用类型定义了无环层次结构。节点也可以组织为另一层次结构，而这样跟随层次化引用将会如图 2-7 所示那样通向节点。在这种情况下，使用了 HasComponent 引用类型，将配置作为该设备的组件对外暴露。作为抽象的聚合（Aggregates）引用类型的具体子类型，它继承了“is-part-of”（是……的一部分）语义。is-part-of 关系的性质是不能产生循环的，“A”的一部分不能包含作为子部分的“A”。注意，不允许循环并不意味着一个节点不能有两个父节点，例如 Config1 是 Device1 和 Device2 的组件，从而在这个层次结构中有两个父节点。

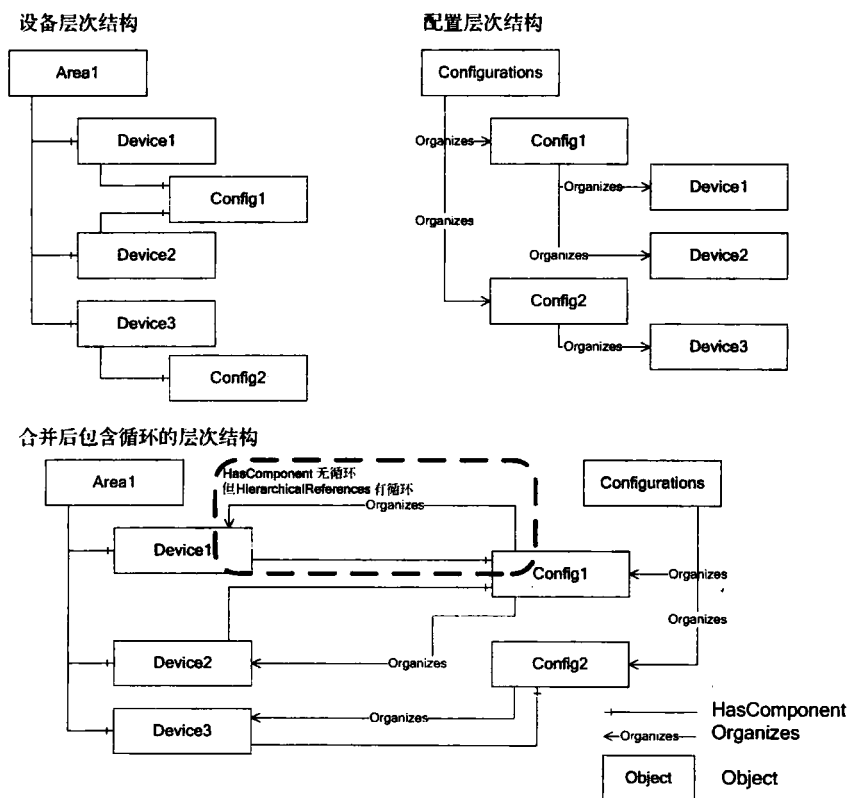


图 2-7 循环的和无循环的层次结构示例

与配置层次结构合并后组合仍然有可能循环。配置层次结构中不能使用 HasComponent 引用, 否则继承的 HasChild 引用类型的无循环约束将被打破。由于逻辑上使用该配置的设备不是配置的一部分, 使用 HasComponent 引用类型是没有意义的。

另外一个 HasChild 引用的例子是 HasSubtype 引用类型暴露类型层次结构 (例如, 在图 2-5 中的引用类型层次)。在这种情况下, 循环没有任何意义, 因为类型不能是自己的 (间接) 子类型。

之所以允许层次化引用形成循环, 是因为服务器可能会通过不同的层次结构暴露相同的节点, 正如图 2-7 所示。服务器可以在 HierarchicalReferences 之下定义其他自有的引用类型来暴露附加的层次结构。它们可以选择让这些层次是无循环的, 但与其他层次结构结合可能是有循环的, 例如, 使用 HasChild 引用的层次结构。如果没有额外的层次结构需要定义, 只有一个更具体的引用语义, 新增引用类型应作为 HasChild 引用类型的子类型。

### 什么是允许的, 什么是不允许的?

节点之间的引用是严格的。不允许在相同的节点之间提供两次类型和方向都相同的引用。这包括该引用类型的子类型, 但不包括父类型。例如, 允许具有不同类型的 Hierarchical References, 但不能使用 HasComponent 与其子类型。关于同一个类型的约束是模型固有的, 因为引用是由源节点、目标节点、引用类型定义的。为了避免这种情况, 服务器提供一个“A”类型的引用, 也可以提供“A”的更特定化的子类型“B”的引用, 但不能同时提供这两种引用。只有最适当的引用类型应该被使用。使用引用类型总是意味着父类型也是有效的。客户端在地址空间上使用过滤器浏览或查询时必须考虑这一点。

特定的引用类型进一步限制了引用类型的使用。这可能包括允许的源节点和目标节点 (例如, 限制节点类别或具体类型)、一个引用在同一节点上使用次数 (例如, 每个引用类型只能是一个 HasSubtype 引用<sup>⊖</sup>的目标)、是否允许循环如 HasChild 类型指定的等。子类型只能进一步限制引用的使用, 父类型的限制在子类型中仍然有效。

没有标准化的方式在地址空间暴露引用类型的约束。然而一般来说, 描述 (Description) 属性是放置约束的文字描述的好地方。

## 2.4 对象、变量和方法

在 OPC UA 中, 最重要的节点类别 (NodeClass) 是对象、变量和方法。这些

⊖ 请注意: HasSubtype 引用被建模为从父类型指向子类型。它是一个层次化的引用。

概念在面向对象编程领域也为人熟知。对象拥有变量和方法，而且可以触发事件。

节点类别为变量的节点代表一个值，该值的数据类型取决于该变量。客户端可以对这个值进行读取、写入和订阅其变化。一个变量在如下情况下被使用：代表温度传感器测量的温度或管理一些控制应用的设置点（Setpoint），但一般用于暴露在地址空间中，除了引用和节点属性之外的任意其他数据。包括配置数据或描述节点的附加元数据。

节点类别为方法的节点代表服务器中一个由客户端调用并返回结果的方法。每个方法指定客户端应使用的输入参数和客户端希望得到的结果即输出参数。使用方法的目的是它执行得比较快。客户端使用调用服务（Call Service）调用方法（见第5章），服务调用的响应已经包含了结果。当服务器需要提供长时间运行的过程给客户端起动和控制时，应该使用程序（Programs）（见4.8节）。例如打开阀门或起动电动机的方法，以及一些对输入值进行模拟计算之类的更复杂的任务。一般来说，当一组参数被用作输入或输出或者同时是输入输出时，或者服务器会按预定的方式触发一个特定的动作时，使用方法是可行的。OPC UA 的方法只提供一个方法的签名。没有标准化的方式来获取或设置 OPC UA 服务器上的方法实现。

节点类别为对象的节点用于地址空间结构。除了 DisplayName（显示名称）、Description（描述）等描述节点的属性外，对象不包含数据。对象使用变量来对外提供值。对象不像变量一样拥有 Value（值）属性。对象可以用于分组管理变量、方法或其他对象。虽然 OPC UA 没有定义一个清晰的所有权的概念，方法和变量总是属于一个对象（或对象类型，见2.5节）。方法总是在对象的上下文中被调用。除了包含的方法和变量，对象也可以是一个事件通知器。客户端可以订阅事件通知器来接收事件（见2.10节）。

在图2-8中，总结了一个对象包含对象、变量、方法和生成事件的概念。Motor（电动机）对象包含一个 Status（状态）变量以确定电动机是否运行。客户端可以订阅这个变量，从而在每次电动机的状态发生改变时得到通知。此外，Motor 有一些配置变量，在另一个对象 Configuration（配置）中管理着。客户端可以读取或订阅这些变量，而且还可以通过写入变量来改变配置。客户端可以调用 Start 和 Stop 方法来起动或停止电动机。此外，客户可以订阅 Motor 的事件。该电动机可以在进入维护状态、不能正常工作时生成事件。Motor 对象可以使用特定引用类型连接到其他对象，在这个例子中，只使用 MyReferenceType 引用到了 Object1 对象。在这种情况下，它引用的对象不被认为是电动机的一部分。

对象惟一的附加属性是用来识别对象是否可以作为事件通知器，即客户端是否可以订阅对象来接收事件，或者读取、更新事件的历史。该属性见表2-3。

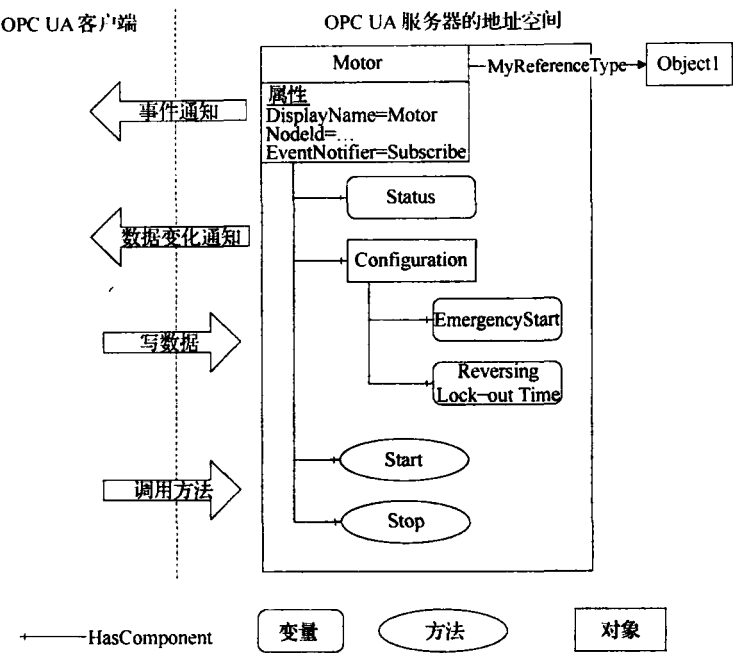


图 2-8 对象、变量和方法概览

表 2-3 对象的附加属性

属 性	数 据 类 型	说 明
包含所有在表 2-1 中定义的通用属性		
EventNotifier	Byte	这是一个位掩码属性，标识对象是否可以用来订阅的事件，事件的历史是否会被访问和改变

变量提供真实的数据，因此提供的附加属性数量要高得多。表 2-4 总结了这些属性。

表 2-4 变量的附加属性

属 性	数 据 类 型	说 明
包含所有在表 2-1 中定义的通用属性		
Value	不固定，由其他属性指定	变量的实际值。该值的数据类型由 DataType、ValueRank 和 ArrayDimensions 属性指定
DataType	NodeId	数据类型表示为地址空间中的节点。此属性包含一个这种节点的 NodeId，从而定义了 Value 属性的数据类型
ValueRank	Int32	标识值是否是一个数组，如果它是一个数组，它允许指定数组的维度



(续)

属 性	数 据 类 型	说 明
ArrayDimensions	UInt32 []	这个可选属性允许指定数组的大小，只能在值是一个数组时使用。对于数组的每一个维数对应的条目定义维度的长度
AccessLevel	Byte	一个位掩码 Value 属性的当前值是否可读可写，以及 Value 的历史是否可读和可改变
UserAccessLevel	Byte	与 AccessLevel 包含相同的信息，但需要考虑到用户的访问权限
MinimumSamplingInterval	时间区间	这个可选属性提供信息指明 OPC UA 服务器需要多长时间能检测到 Value 属性的变化。对于服务器不直接管理的值，例如，一个温度传感器的温度，服务器可能需要扫描设备变化（轮询），因此无法比最小时间间隔更快检测到变化
Historizing	布尔	指示服务器目前是否收集了 Value 的历史。AccessLevel 属性不提供这些信息，它仅指定是否有历史可用

Value 属性的数据类型由 DataType、ValueRank 和 ArrayDimensions 属性定义。之所以有三个属性，是因为 OPC UA 内置支持了多维数组。客户可以读取或写入数组的一部分，也可以只订阅数组的一部分。因此数据类型只指定基本类型，而其他属性定义是不是使用了该数据类型的数组或矩阵，并且也可以定义数组或矩阵的大小。有关数据类型的更多细节在 2.8 节中描述。不使用引用来表示数据类型的原因是，一些变量可能会经常改变数据类型，因此客户可能需要订阅它们。如 2.11.3 节所述，在 OPC UA 中跟踪引用的变化更难。

为了避免混淆定义值的数据类型的几个属性，表 2-5 提供了使用这些属性的例子。

表 2-5 变量中定义 Value 属性类型的属性的示例

可能的值	数据类型	ValueRank	ArrayDimensions
"Just a String"	String	-1 (标量)	—
{1, 2, 3} {4, 7, 9, 12}	Int16	1 (一维)	—
{1, 2, 3} {3, 4, 8}	UInt16	1 (一维)	{3}
1 {1, 4, 9} {1, 2} {1, 5}	UInt32	-2 (任意)	—
{3, 4} {1, 2} {3, 4}	Int32	2 (二维)	{2, 3}
{123, 123} {1, 2} {1, 1} {2, 4}	UInt64	0 (一维或多维)	—

WriteMask 和 AccessLevel 以及 UserWriteMask 和 UserAccessLevel 之间的主要区别是：AccessLevel 只与 Value 属性相关。WriteMask 除了指明写入权限，还记录了