ICS 25.040
N 10

中华人民共和国国家标准

GB/T 33863.4—2017/IEC 62541-4:2011

OPC 统一架构　第 4 部分:服务

OPC unified architecture—Part 4:Services

(IEC 62541-4:2011,IDT)

2017-07-12 发布　　2018-02-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　　次

前　言

GB/T 33863《OPC 统一架构》由以下各部分组成：

——第 1 部分：概述和概念；

——第 2 部分：安全模型；

——第 3 部分：地址空间模型；

——第 4 部分：服务；

——第 5 部分：信息模型；

——第 6 部分：映射；

第 7 部分：规约；

——第 8 部分：数据访问；

第 9 部分：报警和条件；

第 10 部分：程序；

——第 11 部分：历史访问；

——第 12 部分：发现；

——第 13 部分：聚合。

本部分是 GB/T 33863 的第 4 部分。

本部分按照 GB/T 1.1—2009 给出的规则起草。

本部分使用翻译法等同采用 IEC 62541-4:2011《OPC 统一架构　第 4 部分：服务》。

本部分由中国机械工业联合会提出。

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

本部分起草单位：机械工业仪器仪表综合技术经济研究所、北京三维力控科技有限公司、上海自动化仪表有限公司、重庆川仪自动化股份有限公司、西南大学、中国工程物理研究院动力部。

本部分主要起草人：王麟琨、王春喜、李云、丁露、王玉敏、丁研、张庆军、姚杰、刘枫、郑秋平。

引　言

本部分为OPC统一架构应用开发者提供了规范。本标准给出了为开发标准接口而进行分析和设计的过程。该标准接口可加快由多个供应商完成的应用开发，并实现内部操作的无缝连接。

OPC 统一架构　第 4 部分:服务

1　范围

GB/T 33863 的本部分定义 OPC 统一架构(OPC UA)的服务(Services)。所描述的服务是抽象的远程过程调用(RPC)的集合。服务由 OPC UA 服务器实现,被 OPC UA 客户端调用。OPC UA 客户端和服务器之间的所有交互都通过这些服务进行。由于本部分没有定义特定的 RPC 实现机制,所以本部分定义的服务是抽象的。IEC 62541-6 为实现规定了一种或几种具体的映射。例如,IEC 62451-6 中有一种到 XML Web 服务的映射。在这种映射中,本部分定义的服务作为 WSDL 合约中的 Web 服务方法出现。

并非所有的 OPC UA 服务器都需要实现所有定义的服务。IEC 62541-7 定义了行规,指出为了与特定行规兼容,需要实现哪些接口。

本部分用于指导开发 OPC UA 客户端或服务器应用。

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。

IEC/TR 62541-1:2010　OPC 统一架构　第 1 部分:概述和概念(OPC unified architecture　Part 1:Overview and concepts)

IEC/TR 62541-2　OPC 统一架构　第 2 部分:安全模型 (OPC unified architecture—Part 2:Security model)

IEC 62541-3　OPC 统一架构　第 3 部分:地址空间模型(OPC unified architecture—Part 3:Address space model)

IEC 62541 5　OPC 统一架构　第 5 部分:信息模型(OPC unified architecture—Part 5:Information model)

IEC 62541-6　OPC 统一架构　第 6 部分:映射(OPC unified architecture　Part 6:Mappings)

IEC 62541-7　OPC 统一架构　第 7 部分:规约(OPC unified architecture　Part 7:Profiles)

IEC 62541-8　OPC 统一架构　第 8 部分:数据访问(OPC unified architecture　Part 8:Data access)

ISO/IEC 7498(所有部分)　信息处理系统　开放系统互连　基本参考模型(Information processing systems　Open systems interconnection　Basic reference model)

3　术语、定义、缩略语和约定

3.1　术语和定义

IEC/TR 62541-1、IEC/TR 62541-2 和 IEC 62541-3 界定的以及下列术语和定义适用于本文件。

3.1.1

死区 Deadband

值发生改变但不会触发数据改变通知的允许范围。

注：当提前为变量分配死区时，死区可用作滤波器，避免噪声信号不必要地更新客户端。本部分定义了绝对死区(AbsoluteDeadband)作为通用滤波器。IEC 62541-8定义了附加死区滤波器。

3.1.2

端点 Endpoint

允许客户端访问一个或多个由服务器提供的服务的网络上可用的物理地址。

注：每个服务器可有多个端点，端点地址必须包含一个主机名称(HostName)。

3.1.3

网关服务器 Gateway Server

为一个或多个服务器充当中介的服务器。

注：可使用网关服务器限制外部访问，并提供协议转换或提供底层服务器不支持的特性。

3.1.4

主机名称 HostName

网络上主机的惟一标识符。

注：本标识符在局域网内应是惟一的，但它也可以是全局惟一的。

3.1.5

安全令牌 Security Token

加密密钥集的标识符。

注：所有安全令牌属于一个安全环境，对于OPC UA该安全环境是安全通道。

3.1.6

软件证书 SoftwareCertificate

软件产品的数字证书，能安装在多个主机上以描述软件产品的能力。

注：一个软件产品的不同部件可以有相同的软件证书。

3.2 缩略语

下列缩略语适用于本文件。

API：应用程序接口(Application Programming Interface)

BNF：巴科斯范式(Backus-Naur Form)

CA：证书授权(Certificate Authority)

CRL：证书撤销列表(Certificate Revocation List)

CTL：证书信任列表(Certificate Trust List)

DA：数据访问(Data Access)

UA：统一架构(Unified Architecture)

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

URL：统一资源定位器(Uniform Resource Locator)

3.3 约定

OPC UA服务包含在客户端和服务器之间传送的参数。OPC UA服务规范使用表格来描述服务的参数，如表1所示。该表中将这些参数组织为请求参数和响应参数。

表 1　服务定义列表

名称	类型	描述
请求		定义服务的请求参数
简单参数名称		该参数的描述
结构化参数名称		该结构化参数的描述
组件(成员)参数名称		该组件参数的描述
响应		定义服务的响应参数

名称、类型和描述列中包含每一参数的名称、数据类型和描述。所有的参数都是必备的，尽管有些参数在特定条件下不会使用。描述列中规定了当某个参数未被使用时需要提供的值。

这些表中定义了简单参数和结构化参数两类。简单参数具有简单的数据类型，如布尔型或字符串型。

结构化参数有两个或多个成员参数(component parameter)构成，成员参数可以是简单或结构化类型。成员参数名称在结构化参数名称的下边，并带有缩进。

这些表中使用的数据类型可以是基本类型、多个服务的通用类型、或服务特定的类型。基本数据类型定义在 IEC 62451-3 中。服务使用的基本数据类型列在表 2 中。第 7 章中定义了多种服务通用的数据类型。服务特定的数据类型定义在该服务的参数表中。

表 2　IEC 62451-3 定义的参数类型

参数类型
BaseDataType
NodeId
QualifiedName
LocaleId
Boolean
ByteString
Double
Guid
Int32
String
UInt16
UInt32
UInteger
UtcTime
XmlElement
Duration

本标准适用的术语"服务(Services)"与 ISO/IEC 7498 中的定义是一致的。请求和指示服务原语的参数在表 1 中表示为请求参数。同样地,响应和证实服务原语的参数在表 1 中表示为响应参数。因为,所有的请求和响应参数在发送者和接收者之间无修改地传输。所以,无需为请求、指示、响应和证实的参数值单独提供一列。故意进行了省略,以提高可读性。

4 概述

4.1 服务集(Service Set)模型

本条规定了 OPC UA 的服务。OPC UA 服务的定义是抽象描述,并不是实现的规范。抽象描述和这些服务的通信栈(Communication Stack)之间的映射在 IEC 62541-6 中定义。当实现为 web 服务时,OPC UA 的服务(OPC UA Services)对应于 Web 服务,一个 OPC UA 服务对应于一个 Web 服务的操作。

这些服务被组织为服务集。每一服务集定义了一组相关的服务。服务集的组织在本标准中是逻辑分组的,并不用于实现。

图 1 描述了 Discovery 服务集定义的服务。这些服务允许客户端发现服务器实现的端点(Endpoints),并读取这些端点的安全(Security)配置。

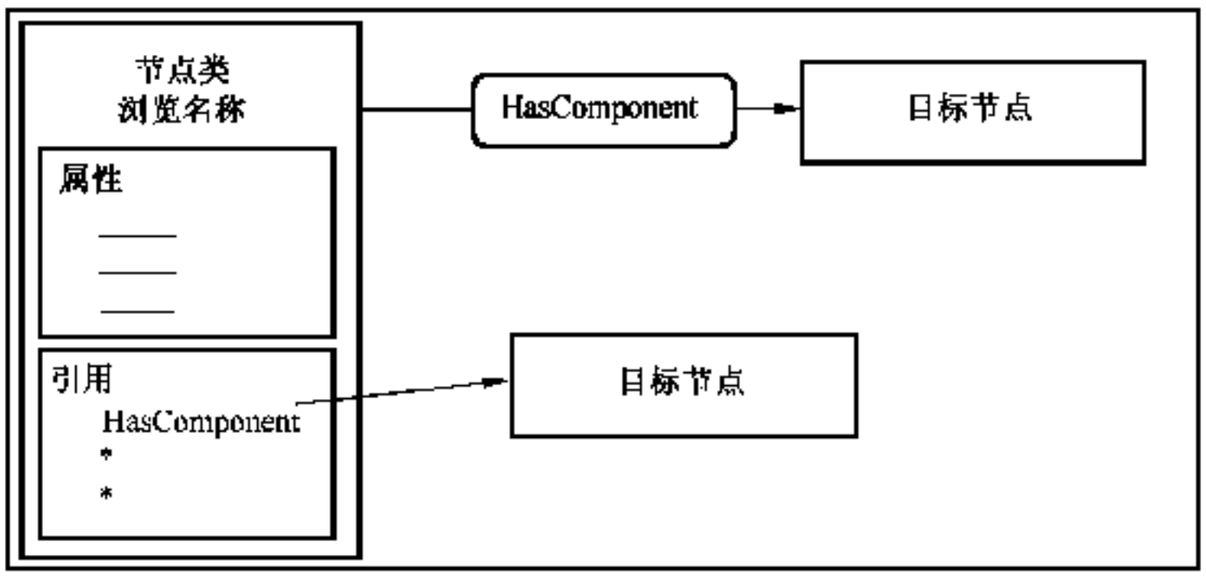

图 1 Discovery 服务集

图 2 描述了 SecureChannel 服务集定义的服务。这些服务允许客户端建立通信通道,以确保和服务器交换消息的私密性(Confidentiality)和完整性(Integrity)。

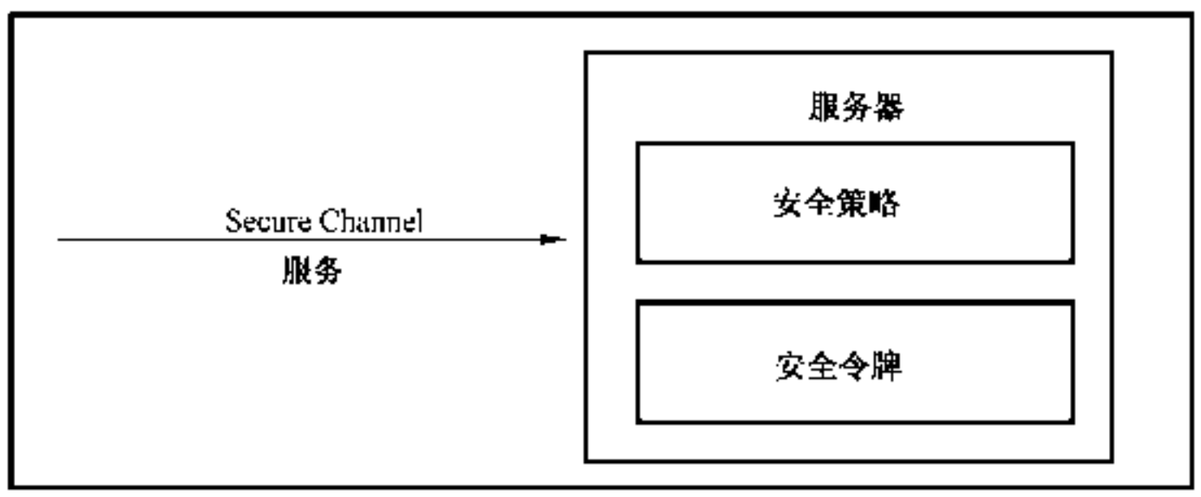

图 2 SecureChannel 服务集

图 3 描述了 Session 服务集定义的服务。这些服务允许客户端对用户的行为进行鉴别,并管理会话。

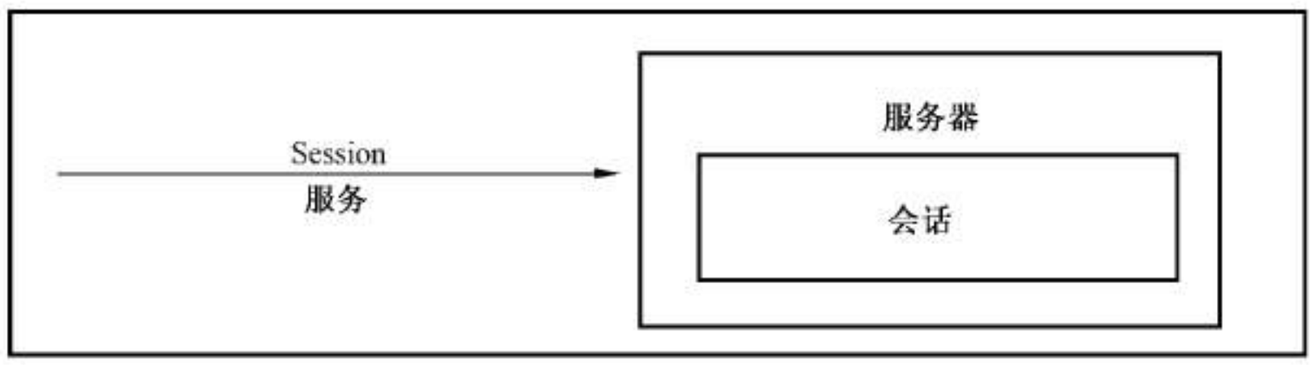

图 3 Session 服务集

图 4 描述了 NodeManagement 服务集定义的服务。这些服务允许客户端在 AddressSpace 中添加、修改和删除节点。

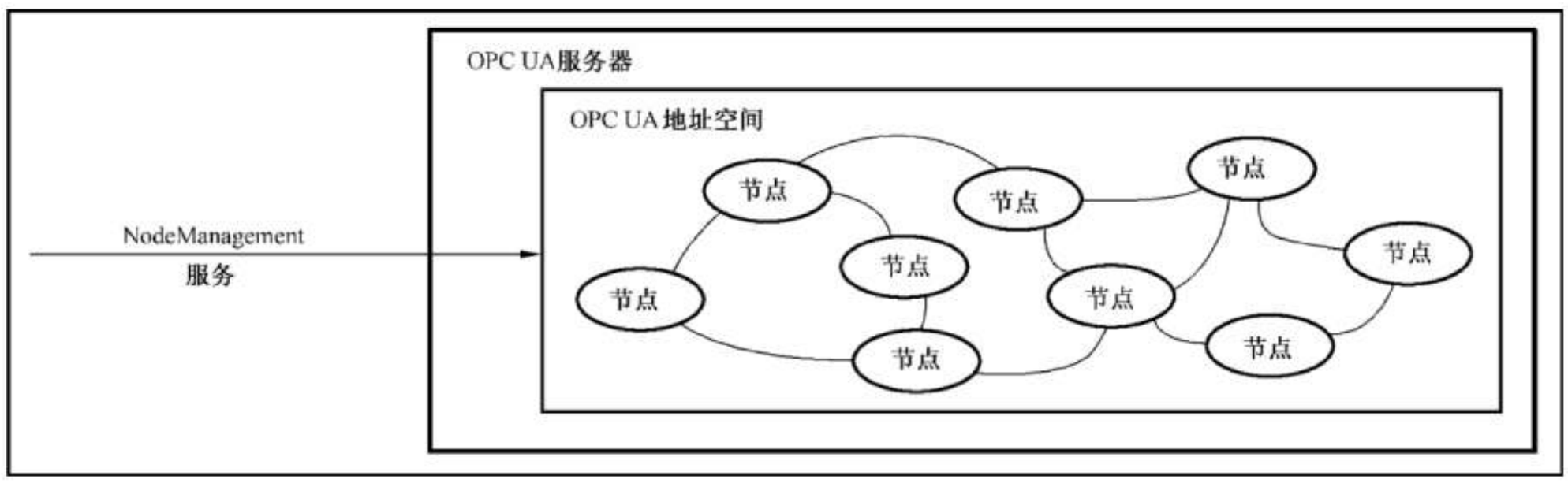

图 4 NodeManagement 服务集

图 5 描述了 View 服务集定义的服务。这些服务允许客户端通过 AddressSpace 或 AddressSpace 子集调用视图(View)。Query 服务集允许客户端通过 AddressSpace 或 View 获取数据子集。

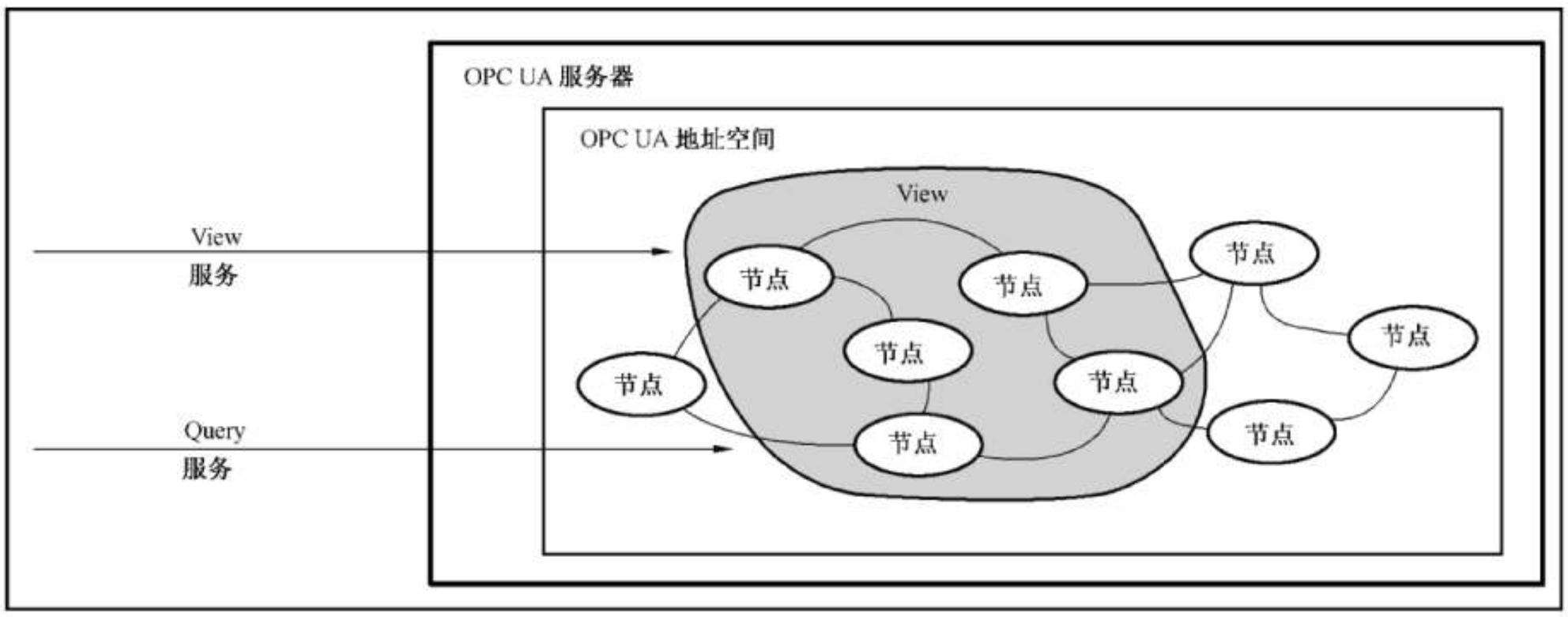

图 5 View 服务集

图 6 描述了 Attribute 服务集。它定义了允许客户端读写节点属性(包括它们的历史值)的服务。因为变量的值被模型化为属性,所以这些服务允许客户端读写变量的值。

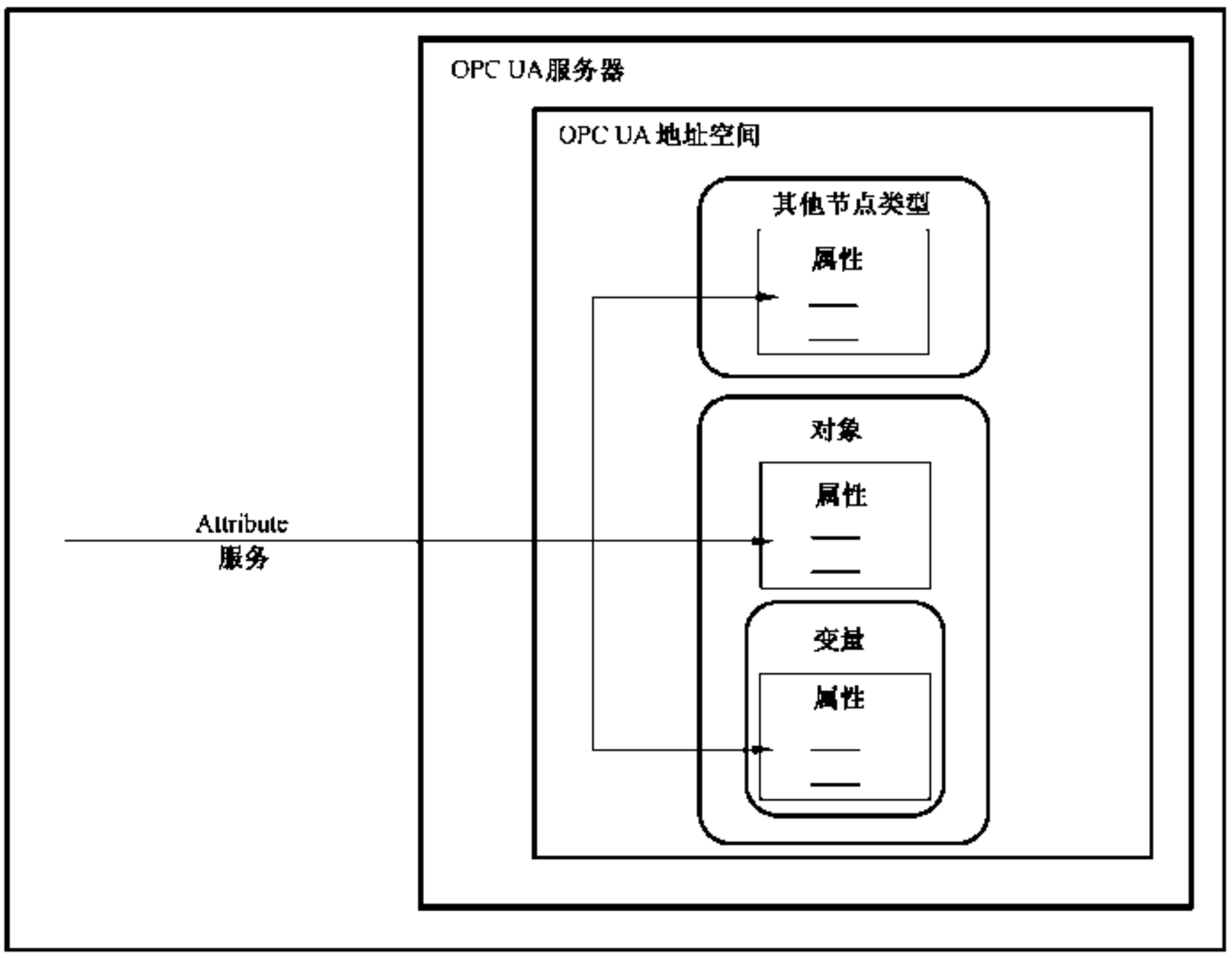

图 6　Attribute 服务集

图 7 描述了 Method 服务集，该服务集定义了允许客户端对方法进行调用的服务。当调用时方法开始运行直到运行结束。方法可以使用特定的输入参数进行调用，并返回特定的输出参数。

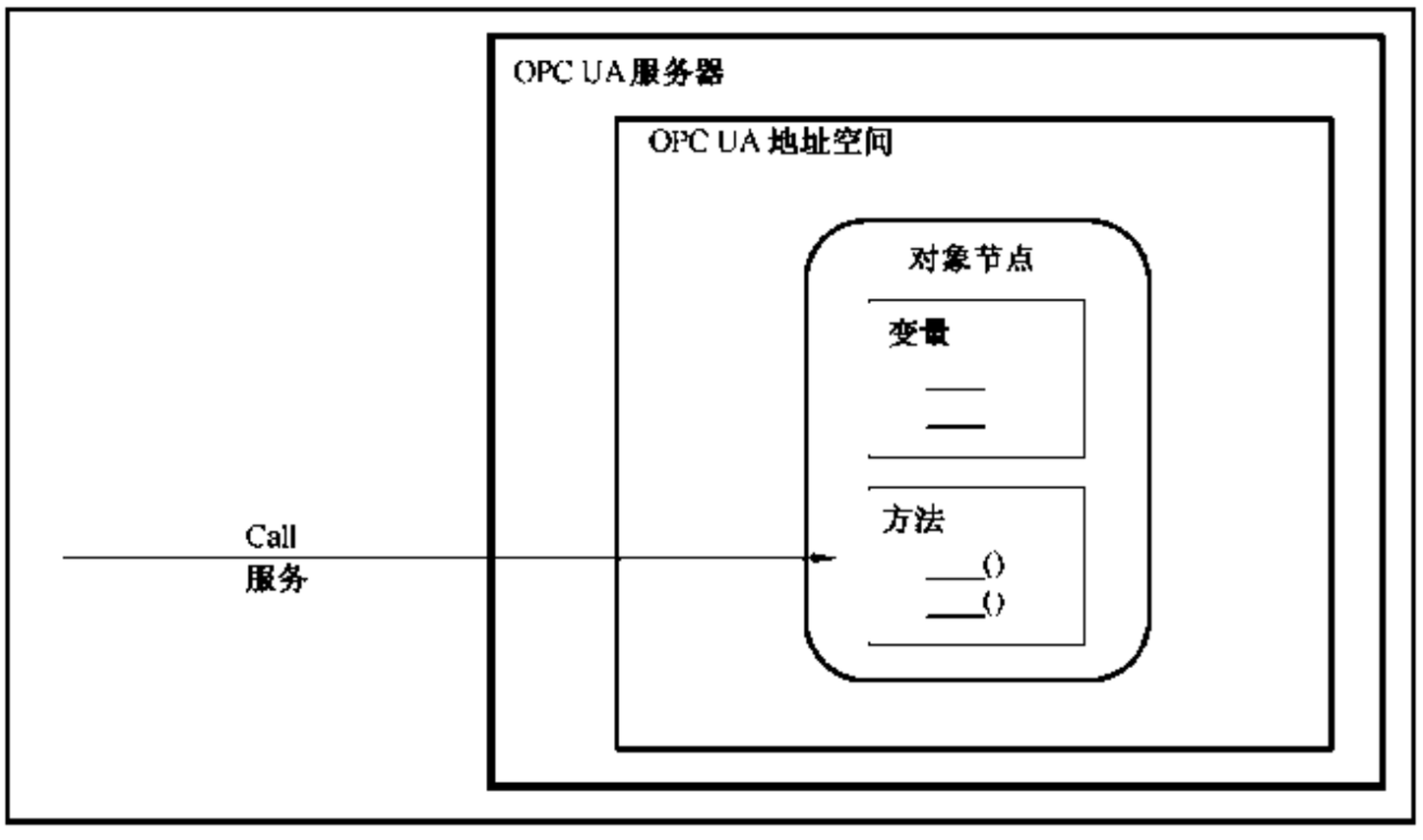

图 7　Method 服务集

图 8 描述了 MonitoredItem 服务集和 Subscription 服务集，它们结合使用以订阅 OPC UA 地址空间中的节点。

MonitoredItem 服务集定义的服务允许客户端创建、修改和删除用于监视值改变的属性和事件对象的 MonitoredItems。

这些通知(Notifications)由订阅(Subscription)以排队方式传输到客户端。

Subscription 服务集定义的服务允许客户端创建、修改和删除订阅。订阅发送由 MonitoredItems

产生的通知(Notifications)给客户端。

Subscription 服务集也为客户端提供从丢失的报文和通信故障中进行恢复。

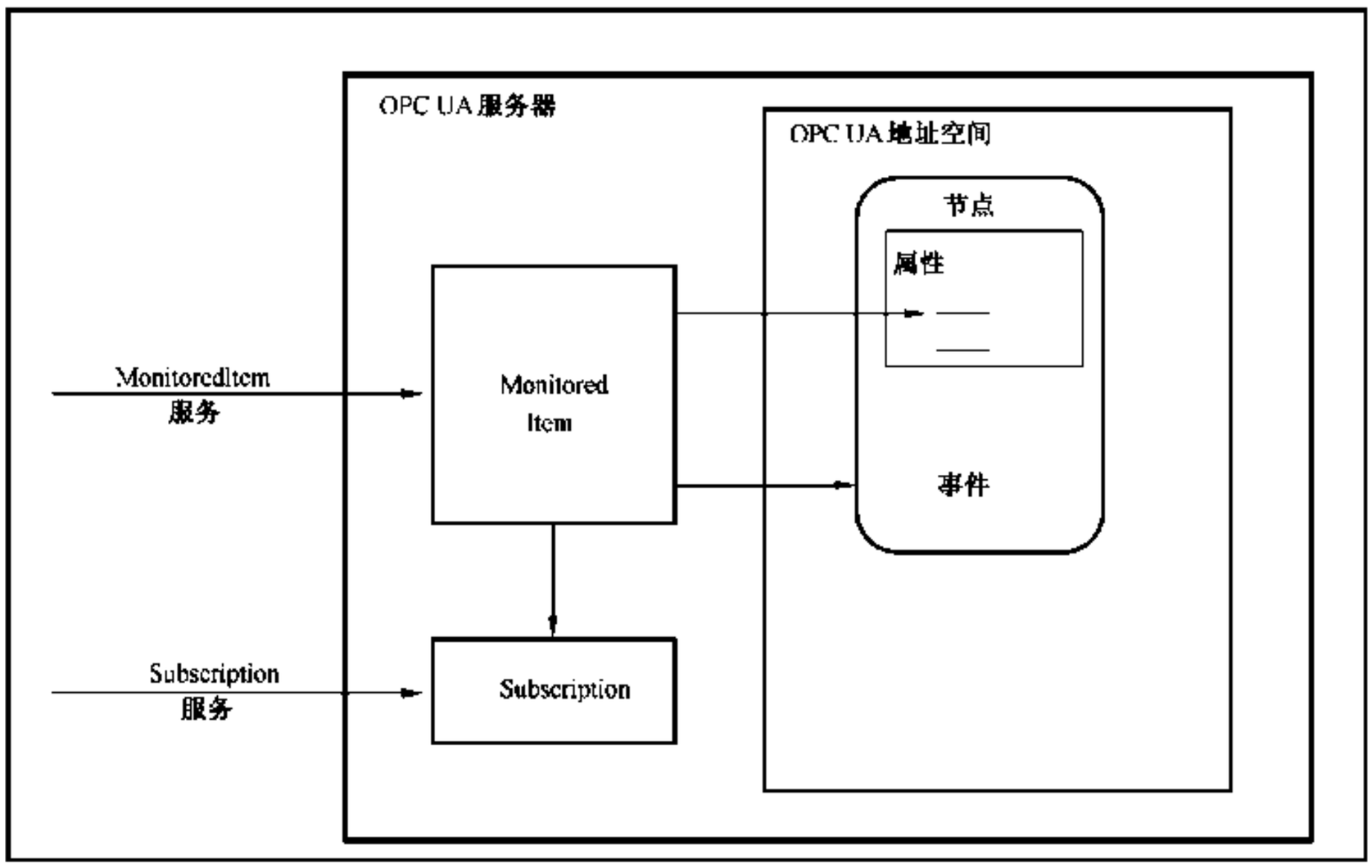

图 8 MonitoredItem 和 Subscription 服务集

4.2 请求/响应服务规程

请求/响应服务规程描述了由服务器接收的请求的处理以及随后的返回响应。当客户端向服务器提交一个服务请求报文时,该规程开始执行。

当收到请求时,服务器通过两个步骤处理报文。第一步,它尝试解码并定位要执行的服务。这一步的错误处理与所使用的特定通信技术有关,在 IEC 62541-6 中进行描述。

如果第一步成功,它尝试访问在请求中所标识的每一个操作并执行所请求的操作。对于请求中的每个操作,它产生一个独立的成功/失败代码。该代码包含在一个正响应报文中并和其他数据一起返回。

为了执行这些操作,客户端和服务器都可以使用通信栈的 API 来建立和解释报文,并访问请求的操作。

实现每个服务请求或响应处理时,应检测每个服务参数是否在该参数所规定的范围内。

5 服务集

5.1 概述

本章定义 OPC UA 服务集及其服务。第 7 章包含这些服务所使用的公共参数的定义。6.2 描述所有服务的审计需求。

服务器是否支持一个服务集,或服务是否在服务集中,由行规定义。这些行规描述在 IEC 62541-7 中描述。

5.2 服务请求和响应首部

每个服务请求有一个 RequestHeader。每个服务响应有一个 ResponseHeader。

RequestHeader 的结构定义在 7.26 中。它包含通用的请求参数，如 authenticationToken、timestamp 和 requestHandle。

ResponseHeader 的结构定义在 7.27 中。它包含了通用的请求参数，如 serviceResult 和 diagnosticInfo。

5.3 服务结果

服务结果分两个层次在 OPC UA 响应中返回。一层指出服务调用的状态，另一层指出该服务请求的每个操作的状态。

服务结果通过 StatusCode 定义(见 7.33)。

服务调用的状态由包含在 ResponseHeader(见 7.27)中的 serviceResult 表示。返回这些参数的机制与用于传送服务响应的通信技术有关，并定义在 IEC 62541-6 中。

在请求中的单个操作状态由单个 StatusCodes 表示。

这些参数在下列情形时使用。

a) 如果服务本身失败，在 serviceResult 中返回“bad(坏)”的代码。在这种情况下，返回 7.28 中定义的 ServiceFault。

b) 如果服务完全或部分成功，serviceResult 返回“good(好)”的代码。在这种情况下，返回其他的响应参数。客户端应总是检测响应参数，特别是与每个操作相关的 StatusCodes。这些 StatusCodes 指示在该服务调用中请求的一个或多个操作的“bad”的或不确定的结果。

对于在请求中具有多组操作的所有服务，如果数组为空，应在 serviceResult 中返回“bad”的代码。

服务定义各种特定的 StatusCodes，并且服务器应按照服务的描述使用这些 StatusCodes。客户端应能够处理这些服务特定的 StatusCodes。此外，客户端应能处理表 165 和表 166 中定义的其他通用的 StatusCodes。客户端处理其他特定 StatusCodes 的细节在 IEC 62541-7 中定义。

如果服务器通过某种不同信号传输机制发现用来创建 Session 或 SecureChannel 的应用，或用户资格证书已经损坏，那么服务器应立即终止使用这些资格证书的所有会话和通道。在这种情况下，服务结果代码应是 Bad_IdentityTokenRejected 或 Bad_CertificateUntrusted。

如果报文包含的数据超出了接收者所支持范围，或具有语法错误，报文的解析可能失败。当这种情况发生时，接收者应拒绝报文。如果接收者是服务器，那么它应返回一个带有 Bad_DecodingError 或 Bad_EncodingLimitsExceeded 结果码的 ServiceFault。如果接收者是客户端，那么通信栈应向客户端应用报告这些错误。

许多应用将限定报文的大小和/或报文内包含的数据元素。例如，服务器可能拒绝包含字符串长度超过某一长度的请求。这些限制典型地由管理员设定，并适用于客户端和服务器之间的所有连接。

从服务器接收到 Bad_EncodingLimitsExceeded 错误的客户端可能必须重新修改他们的请求。如果客户端从服务器接收到带有这种错误的响应，管理员可能需要为该客户端增大限值。

在某些情况下，解析错误可能是致命的的且不能返回一个故障。例如，输入的报文可能超过接收者的存储能力。在这种情况下，这些错误可能被视为通信故障，要求重新建立 SecureChannel(见 5.5)。

客户端和服务器通过在 CreateSession 服务中交换报文大小的限值，来减少出现这种致命错误的几率。这使得其各方避免发送导致通信故障的报文。如果连续的响应报文超过了客户端指定的报文大小，服务器应返回 Bad_ResponseTooLarge 错误。类似地，客户端通信栈应在发出超过服务器限值的报文之前，向应用报告 Bad_ResponseTooLarge 错误。

需注意的是，报文大小的限值只用于原始报文主体，不包括首部或应用安全技术而引起的改变。这意味着，报文主体比指定最大值小，也可能会导致致命错误。

5.4 Discovery(发现)服务集

5.4.1 概述

该服务集定义了一些服务,用于发现服务器实现的端点,并读取其安全配置。发现服务由单独的服务器和专门的发现服务器(Discovery Server)来实现。IEC 62541-12 描述了如何使用专用的发现服务器的发现服务。

每一个服务器都应有一个客户端不用建立会话即可访问的发现端点。该端点可以是也可以不是客户端用于建立 SecureChannel 的同一会话端点。客户端通过调用发现服务端点上 GetEndpoints 服务读取建立 SecureChannel 所需的安全信息。

此外,服务器可以通过众所周知的发现服务器,使用 RegisterServer 服务自行注册。之后客户端能够通过调用发现服务器上的 FindServers 服务发现任何已注册的服务器。

完整的发现过程如图 9 所示。

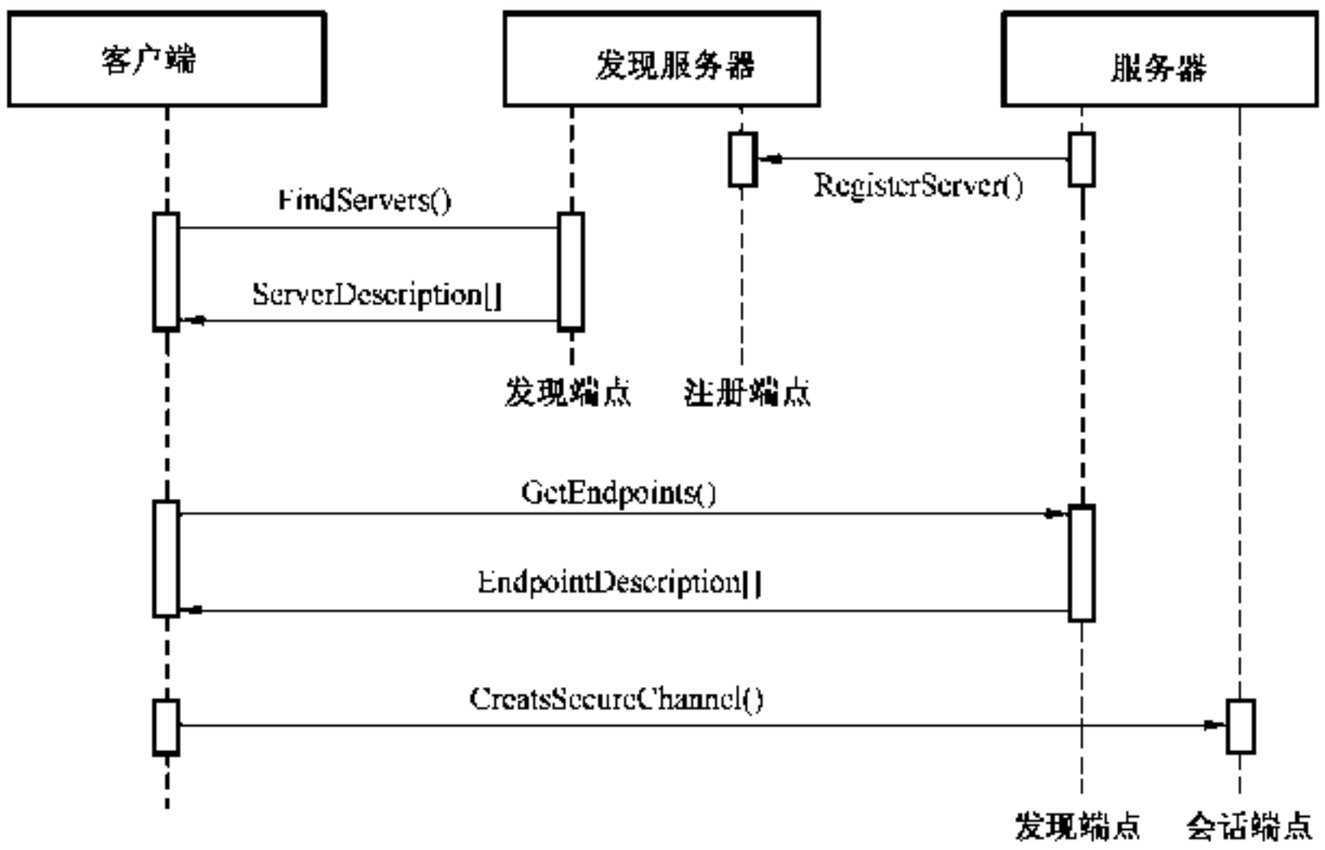

图 9 发现过程

发现端点的 URL 应提供客户端必需的所有信息,以连接到该发现端点。

一旦客户端检索到该端点,该客户端可以保存这些信息,并可再次使用它来直接连接到服务器,而不用再经历发现过程。如果客户端发现不能连接,则可能是服务器配置改变了,客户端应再次经历整个发现过程。

发现端点应不要求任何报文级别的安全性,但可以要求传输层安全性。在生产系统中,由于安全原因管理员可以禁用发现服务,客户端应使用缓存的 EndpointDescriptions。为了对禁用了发现服务的系统提供支持,客户端应允许管理员手动更新用于连接服务器的 EndpointDescriptions。服务器应允许管理员禁用发现端点。

由于没有采用安全措施,所以客户端应谨慎使用从发现端点返回的信息。为此,客户端可以通过对发现端点返回的信息和 CreateSession 响应返回的信息进行比较。客户端应验证:

a) 由服务器证书指定的 HostName 与 EndpointDescription 提供的 endpointUrl 中所包含的 HostName 是相同的;

b) 在 CreateSession 响应中返回的服务器证书与创建 SecureChannel 使用的证书是一样的证书;

c) 发现端点返回的 EndpointDescriptions 与 CreateSession 响应返回的 EndpointDescriptions 是同一 EndpointDescriptions。

如果客户端检测到上述要求之一没有得到满足,该客户端应关闭 SecureChannel 并报告错误。

5.4.2 FindServers

5.4.2.1 描述

该服务返回服务器或发现服务器已知的服务器。IEC 62541-12 详细描述了发现服务器的行为。

客户端可以通过指定过滤条件,减少返回的结果数。如果没有与客户端指定条件相匹配的服务器,则发现服务器将返回空集。5.4.2.2 描述了该服务支持的过滤条件。

每个服务器应提供一个支持这种服务的发现端点。然而服务器应只返回描述自身的单个记录。网关服务器应返回其支持访问的每一服务器的记录,以及允许提供按照普通 OPC UA 服务器访问网关服务器的记录。

每个服务器应具有一个全局惟一标识符,称为 serverUri。该标识符应是一个完全合格的域名,然而,它可以是一个 GUID 或相似的结构,以确保其全球惟一性。该服务返回的 serverUri 应与 ServerArray 属性(见 IEC 62541-5)的索引 0 中出现的值具有相同的值。

每个服务器也应有一个人可读的标识符,称为 ServerName。它不必是全局惟一的。该标识符可以出现在多个地点。

服务器可以有多个 HostName。为此,客户端应向该服务传递连接到端点所使用的 URL。实现该服务应使用该信息返回客户端可以通过所提供的 URL 访问的响应。

该服务不需要任何报文级别的安全,但它可能要求传输层安全。

一些服务器可以通过网关服务器访问,并且应具有一个在其 ApplicationDescription 中用于 gatewayServerUri 的特定值(见 7.1)。ApplicationDescription 提供的 discoveryUrls 应属于网关服务器。如果通过多种路径可以访问服务器,一些发现服务器可能返回同一服务器的多个记录。

在没有任何安全保证下可以使用该服务,因此它非常容易受到 DOS(拒绝服务)的攻击。服务器应使发送此服务的响应所必需的数据处理量最小化,这可以通过预先准备的结果来实现。

5.4.2.2 参数

表 3 定义了该服务的参数。

表 3 FindServers 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数。总是忽略 authenticationToken,7.26 定义了 RequestHeader 类型
endpointUrl	String	客户端用于访问发现端点网络地址。 服务器将该信息用于诊断并确定在该响应中返回什么 URL 如果不能认出 URL 中的 HostName,服务器应返回一个缺省的合适的 URL
localeIds [ ]	LocaleId	将使用的地点列表。 服务器应使用指定地点之一返回 ServerName。如果服务器支持多个请求地点,那么服务器应使用列表中的第一个。如果服务器不支持任何请求位置,则它返回一个缺省的合适的位置。 如果该列表为空,服务器返回合适的缺省位置
serverUris [ ]	String	返回服务器列表。 如果列表为空,返回所有已知服务器

表 3（续）

名称	类型	描述
响应		
responseHeader	ResponseHeader	通用响应参数。 7.27 定义的 ResponseHeader 类型
servers []	ApplicationDescription	满足请求中指定标准的服务器列表。 如果没有服务器满足条件，列表为空。 7.1 中定义了 ApplicationDescription 类型

5.4.2.3 服务结果

表 165 中定义了通用 *StatusCodes*。

5.4.3 GetEndpoints

5.4.3.1 描述

该服务返回服务器支持的端点和建立安全通道和会话所需要的所有配置信息。

该服务不要求任何报文级别的安全，但要求传输层安全。

客户端可以通过指定过滤条件来减少返回结果的数目。如果没有与客户端指定条件相匹配的端点，服务器返回空列表。5.4.3.2 中描述了该服务所支持的过滤条件。

服务器可以为同一 Endpoint 支持多个安全配置。在这种情况下，服务器应对每个可用的配置返回单独的 EndpointDescription 记录。即使物理 URL 相同，客户端也应将每个配置视为不同的端点。

一个端点的安全配置有四个组成部分：

服务器应用实例证书；

——报文安全模式；

安全策略；

——支持的用户身份令牌。

ApplicationInstanceCertificate 用来确保 OpenSecureChannel 请求安全（见 5.5.2）。MessageSecurityMode 和 SecurityPolicy 告诉客户端如何通过 SecureChannel 确保报文安全发送。UserIdentityTokens 告诉客户端在 ActivateSession 请求（见 5.6.3）中什么类型的用户证书应传递给服务器。

每个 EndpointDescription 也为端点支持的行规指定一个 URI。传输层行规规定诸如报文编码格式和协议版本，定义在 IEC 62541-7 中。如果它们想发现服务器所支持的完整列表，客户端应该去取回该服务器的 SoftwareCertificates（见 7.30）。

在消息发送之前，利用标准密码系统算法对其进行加密，可以确保消息的安全，使用的具体算法集取决于端点的 SecurityPolicy。IEC 62541-7 对于通用 SecurityPolicies 定义了行规，并为它们分配一个惟一 URI。要求应用具有行规支持的 SecurityPolicie 内嵌知识，因此，在 EndpointDescription 中只规定了 SecurityPolicy 的行规 URI。客户不能连接到不支持已识别的 SecurityPolicy 的端点。

EndpointDescription 可以指定报文安全模式为 NONE。如果应用不是物理上独立的网络，入侵的风险特别小，不推荐使用这种配置。如果报文安全为 NONE，对客户端来说，可能有意或无意地劫持其他客户端创建的会话。

服务器可以有多个 HostName。为此，客户端应向该服务传递连接到端点所使用的 URL。实现该服务应使用该信息返回客户端可以通过所提供的 URL 可访问的响应。

由于在没有任何安全措施情况下可以使用该服务，因此服务器它非常容易受到 DOS（Denial Of Service）的攻击。服务器应使发送此服务的响应所必需的数据处理量最小化，这可以通过预先准备结果来实现。

在响应中返回的某些 EndpointDescriptions 应指定可用于访问其他服务器的网关服务器的端点信息。在这种状况下，在 EndpointDescription 中指定的 gatewayServerUri 以及所有用于验证证书的安全检测应使用 gatewayServerUri（见 6.1.4）而不是 serverUri。

为了通过网关连接到一个服务器，客户端首先应建立带有网关服务器的 SecureChannel。然后，客户端应调用 CreateSession 服务并向网关服务器传递在 EndpointDescription 中指定的 serverUri。网关服务器然后连接到代表客户端的下层服务器。通过网关服务器连接到服务器的过程如图 10 所示。

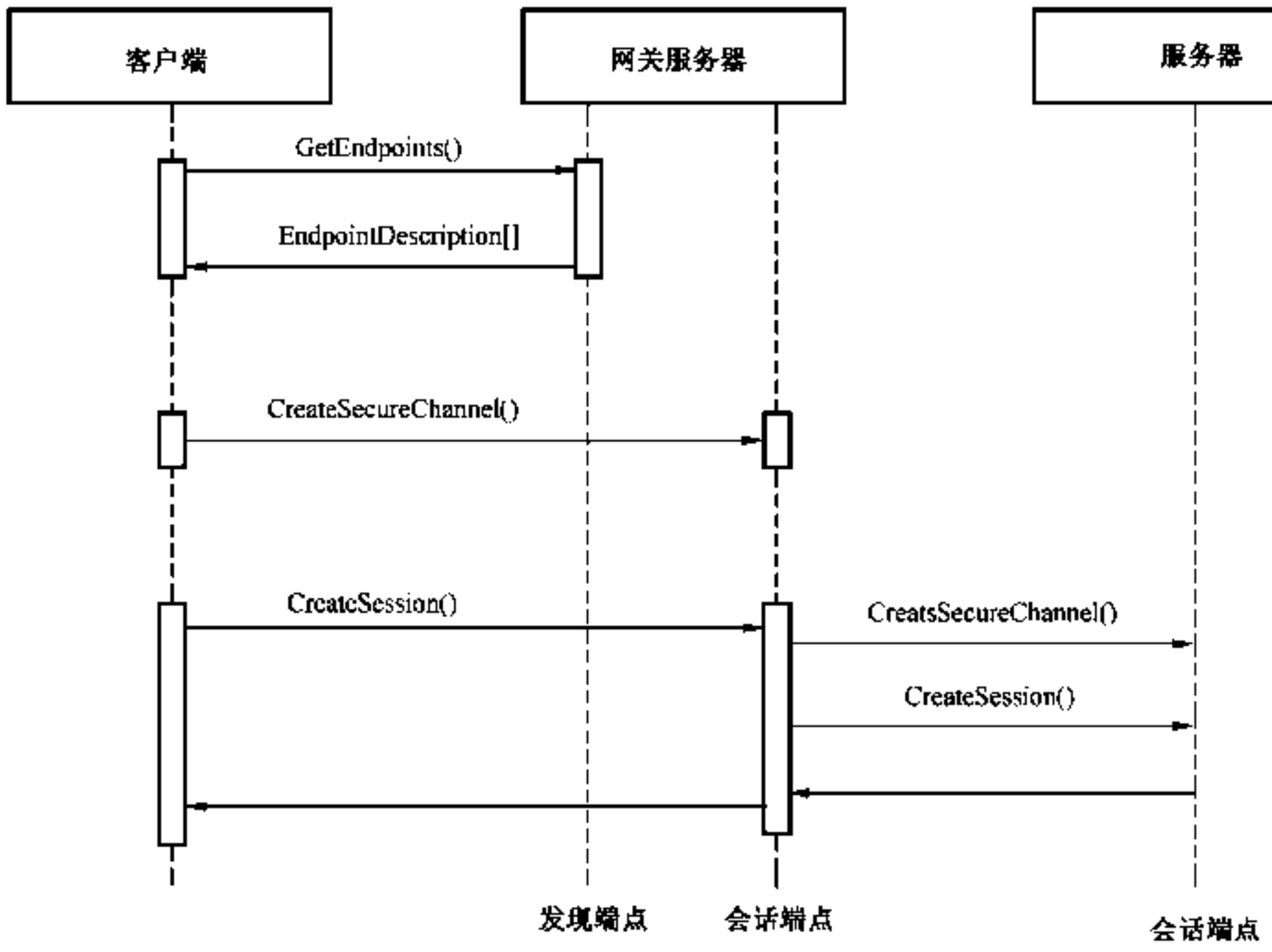

图 10　使用网关服务器

5.4.3.2　参数

表 4 定义了该服务的参数。

表 4　GetEndpoints 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数。 总是忽略 authenticationToken 。 7.26 定义了 RequestHeader 类型
endpointUrl	String	客户端用于访问发现端点的网络地址。 服务器用于诊断的信息，同时用于决定在响应中返回什么 URL。 如果服务器不能识别 URL 中的 HostName，服务器应返回一个缺省的 URL

表 4（续）

名称	类型	描述
localeIds []	LocaleId	将使用的地点列表。 当返回人可读字符串时，指定使用的地点。5.4.4.2 描述了该参数
profileUris []	String	返回的端点应支持的行规列表。如果行规列表为空，则返回所有端点
响应		
responseHeader	ResponseHeader	通用响应参数。7.27 定义了 ResponseHeader 类型
Endpoints []	EndpointDescription	满足请求指定规则的端点列表。如果没有端点满足条件，则列表为空。7.9 定义了 EndpointDescription 类型

5.4.3.3 服务结果

表 165 定义了通用的 *StatusCodes*。

5.4.4 RegisterServer

5.4.4.1 描述

该服务用于向发现服务器注册一个服务器。该服务由服务器或单独的配置功能集调用。客户端不能使用该服务。

在调用该服务之前，服务器应与发现服务器建立安全通道。5.5 中给出了安全通道的描述。服务器管理者应向服务器提供发现服务器的端点描述(EndpointDescription)，作为配置过程的一部分。如果提供的 serverUri 与用于创建安全通道的服务器证书中的 applicationUri 不匹配，发现服务器应拒绝注册。

服务器仅向发现服务器提供其 serverUri 以及发现端点的 URL。客户端使用 GetEndpoints 服务直接从该服务器取回最新的配置信息。

服务器应在其支持的所有地点为其提供一个本地化的名称。

运行在同一台机器上的服务器可以用发现服务器注册其自身。具体机制取决于发现服务器实现，其描述见 IEC 62541-6。

存在两种类型的服务器应用：手动启动的服务器应用和当客户端连接时自动启动的服务器应用。服务器应使用的注册过程取决于所属类别。

手动启动服务器的注册过程如图 11 所示。

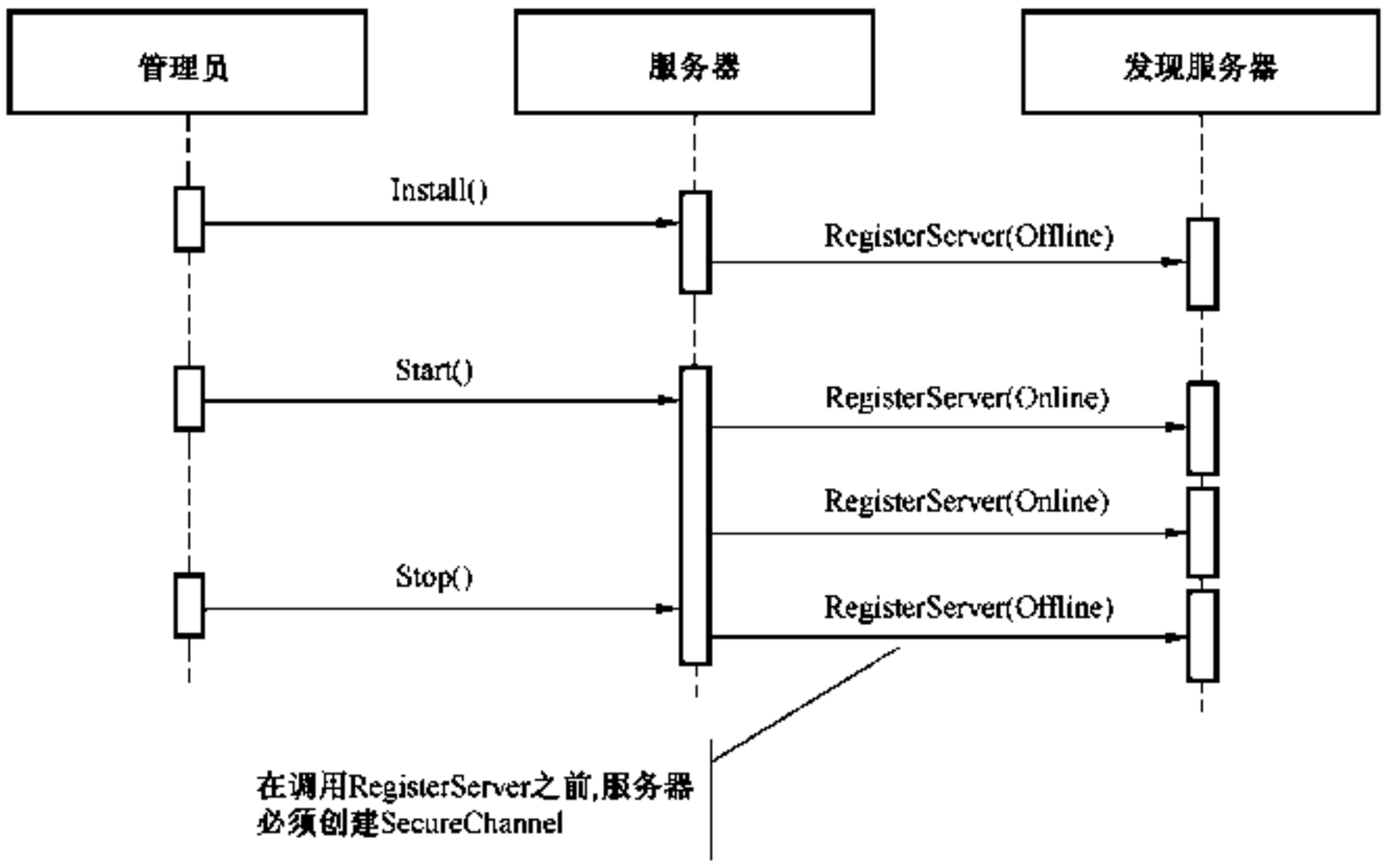

图 11　注册过程——手动启动服务器

自动启动服务器的注册过程如图 12 所示。

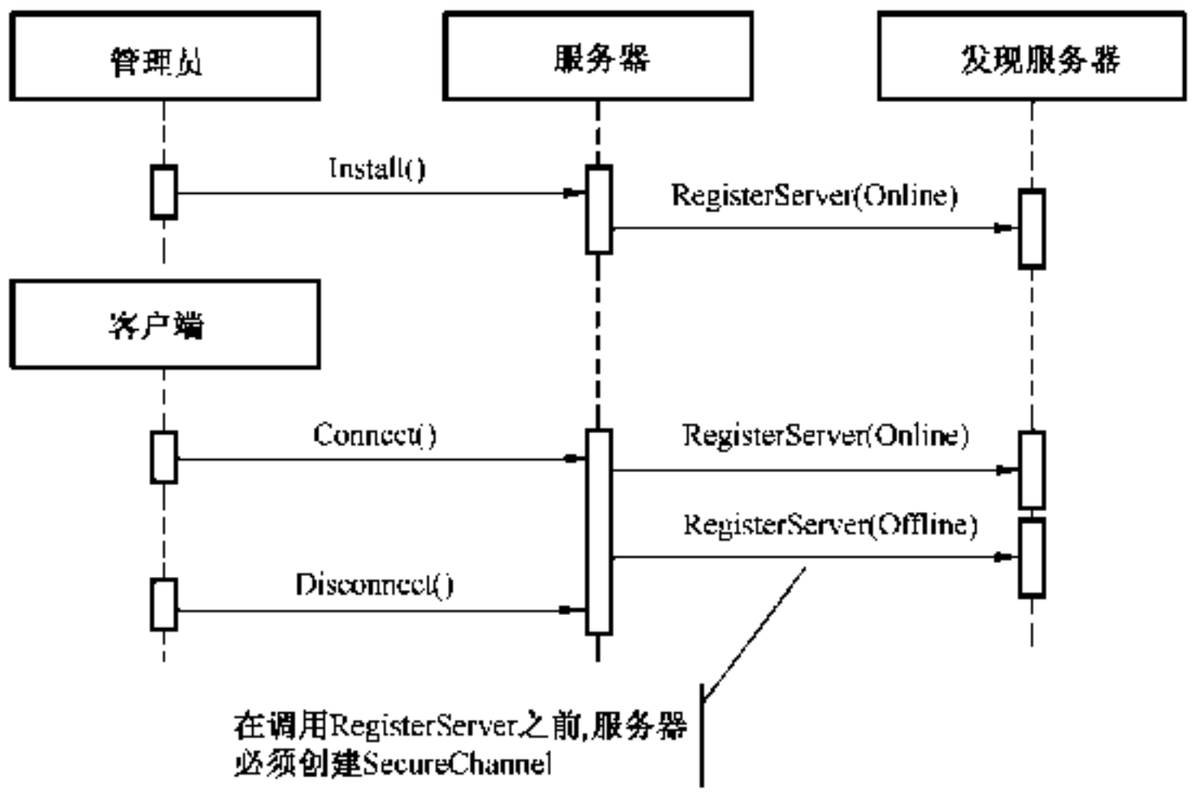

图 12　注册过程——自动启动服务器

注册过程应设计成平台独立的、健壮的并能最小化由误配置所引起的错误。鉴于此，服务器应多次注册自己。

在正常条件下，只要服务器能够从客户端接收到连接，它们都能用发现服务器周期地注册。如果服务器下线，那么应立即注册自身并指出自己即将下线。注册的频率应是可配置的，通常缺省为 10 min。

如果在注册中(如，发现服务器没有运行)出错，那么服务器应周期性地重新尝试注册。这些尝试频率应从 1 s 开始而后逐步增加，直到注册频率与没有错误发生时的注册频率相同。推荐方法为每次双倍尝试间隔时间，直到达到最大值。

当服务器用发现服务器注册时，可能选择提供一种信标(semaphore)文件，发现服务器能用该文件确定服务器是否从机器上卸载。发现服务器能够对包含该文件的文件系统进行读取访问。

5.4.4.2　参数

表 5 定义了该服务的参数。

表 5　RegisterServer 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数。 总是忽略 authenticationToken。 7.26 定义了 RequestHeader 类型
server	RegisteredServer	注册的服务器
serverUri	String	服务器实例的全球惟一标识符
productUri	String	服务器产品的全球惟一标识符
serverNames []	LocalizedText	服务器的可描述的局部化的名称列表。该列表至少应有一个有效的条目
serverType	Enum ApplicationType	应用的类型。 枚举值定义在表 103 中。不允许 "CLIENT_1"值（应用是客户端）
gatewayServerUri	String	网关服务器的 URI 与 discoveryUrls 相连。 该值只被指定为能提供访问希望注册服务器的网关服务器。 对于不作为网关服务器的服务器，该参数为空
discoveryUrls []	String	服务器发现端点列表。该列表至少有一个有效条目
semaphoreFilePath	String	用于标识服务器实例的发出信号的文件路径。 发现服务器在向客户端返回 ApplicationDescription 时，应检测该文件存在与否。 如果相同的发出信号文件被其他服务器使用，那么应删除该注册并被正在传递的替代，并作为该服务调用的一部分。 如果该值无效或为空，DiscoveryServer 不会试图验证该文件存在性
isOnline	Boolean	如果服务器当前能够接收客户端的连接请求，则该值为 True
响应		
ResponseHeader	ResponseHeader	通用响应参数。ResponseHeader 类型定义在 7.27 中

5.4.4.3　服务结果

表 6 定义了该服务的特定服务结果。表 165 定义了通用的 StatusCode。

表 6　RegisterServer 服务结果代码

代码名称	描述
Bad_ServerUriInvalid	见表 165 对该结果码的描述
Bad_ServerNameMissing	没有指定 ServerName
Bad_DiscoveryUrlMissing	没有指定 DiscoveryUrl
Bad_SempahoreFileMissing	指定的信标文件无效

5.5 SecureChannel(安全通道)服务集

5.5.1 概述

该服务集定义了用于打开通信通道的服务,确保与服务器交换的所有报文保密性与完整性。IEC/TR 62541-2 定义了 OPC UA 安全性的基本概念。

SecureChannel 服务与其他服务不太一样,因为它并不是由 OPC UA 应用直接实现。相反,它由 OPC UA 应用赖以建立的通信栈提供。例如,OPC UA 服务器可以建立在 SOAP 栈上,允许应用程序用 WS 安全会话规范建立安全通道。在这些情况下,OPC UA 应用应验证它接收到的报文是否在 WS 安全会话关系中。IEC 62541 6 描述了如何实现 SecureChannel 服务。

安全通道是单个客户端和单个服务器之间的一个长期有效的逻辑连接。该通道保存一套只有客户端和服务器知道的密钥,这些密钥用于在网络上认证和加密。SecureChannel 服务允许客户端和服务器安全地协商所使用的密钥。

EndpointDescription 告诉客户端如何用给定的端点建立安全通道。客户端可以通过一些非 UA 定义的目录服务器或从它自身的配置从发现服务器得到 EndpointDescription。

用于授权与加密报文的具体算法描述在 EndpointDescription 的 SecurityPolicy 域中。当创建安全通道时,客户端可以使用这些算法。

注意,在 IEC 62541-7 中定义的部分 SecurityPolicies 将关闭授权和加密导致安全通道不提供安全措施。

当客户端和服务器通过安全通道通信时,它们应验证所有输入报文已被签名或已按照 EndpointDescription 指定的需求进行加密。OPC UA 应用不处理不符合这些要求的任何报文。

安全通道与 OPC UA 应用之间的关系取决于相应的实现技术。IEC 62541-6 定义了任意依赖于所使用技术的任何需求。

OPC UA 应用中会话与安全通道之间的关系如图 13 所示。OPC UA 应用使用通信栈交换报文。第一步,SecureChannel 服务用于建立确保报文安全交换的两通信栈之间的安全通道。第二步,OPC UA 应用使用会话服务集建立 OPC UA 应用会话。

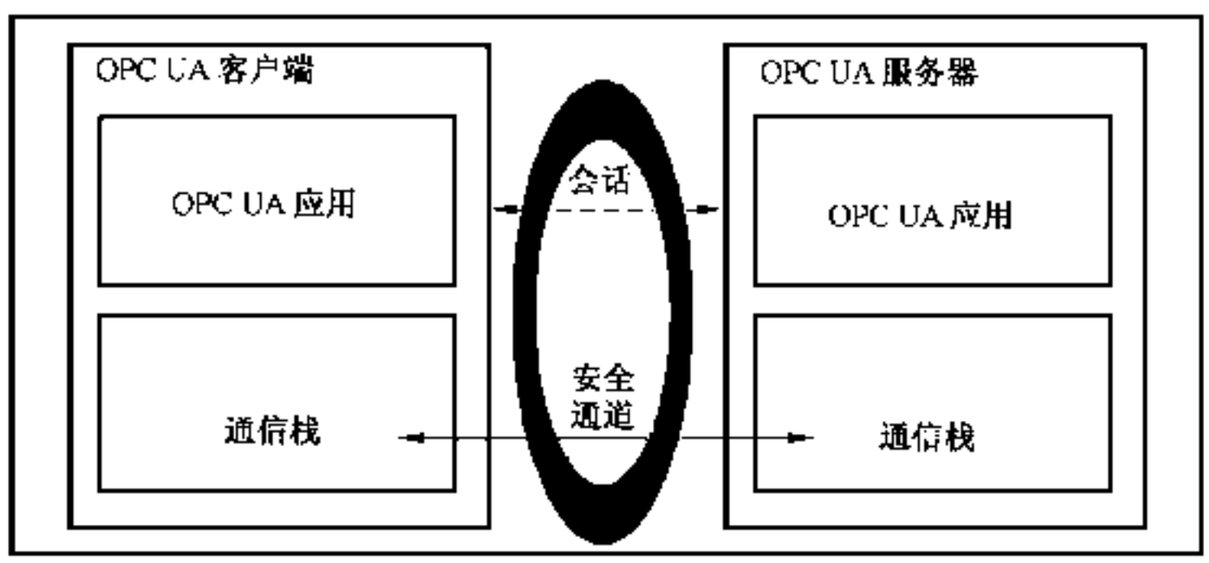

图 13 SecureChannel 和 Session 服务

一旦客户端已建立了一个会话,它可能希望从不同的 SecureChannel 访问该会话。客户端通过 5.6.3 中描述的 ActivateSession 服务确认新的 SecureChannel 来实现该功能。

如果服务器作为其他服务器的客户端,这通常叫做服务器链,那么服务器应能够保持用户级别的安全。通过这种方式,用户身份可以被传递到下层服务器或者它应被映射为下层服务器中合适的用户身份。忽略用户层的安全是不可接收的,这是为了确保安全维护和用户不能得到它们不能访问的信息。在任何时候,服务器都可能通过调用 ActiveSession 服务,传递客户端用户身份给下层服务器来赋予源客户端身份。如果身份赋予(impersonation)不可选,那么服务器应将源客户端用户身份映射为对应服

务器能识别的新的用户身份。

5.5.2 OpenSecureChannel

5.5.2.1 描述

该服务用于打开或新建一个能用于确保会话期间报文交换保密性和完整性的安全通道。该服务要求通信栈发送和接收报文时采用多种安全算法。IEC 62541-6 中描述了对于不同通信栈该服务的特定实现。安全令牌也有全局惟一标识符,该标识符被分配给每个用令牌确保安全的报文。

每个安全通道有一个全球惟一标识符,它对于客户端和服务器应用实例的特定组合是有效的。每个通道包含一个或多个 SecurityTokens,它标识用于加密或授权报文的密钥集。安全令牌也有全局惟一标识符,该标识符被分配给每个用令牌确保安全的报文,这使得授权的接收者知道如何加密和验证报文。

SecurityTokens 与该服务协商有限的生命周期。然而,不同主机系统时钟的差异以及网络延时的差异意味着有效的报文可能在令牌期满后到达。为了阻止抛弃有效消息,应用程序应有如下行为:

a) 客户端应在 SecurityToken 的生命周期已经过 75%时,重新请求一个新的 SecurityToken。这应确保客户端在旧的 SecurityToken 确实期满后接受新的 SecurityToken。
b) 服务器应使用当前的 SecurityToken 确保输出报文安全直到 SecurityToken 期满或服务器接受一个新 SecurityToken 确保安全的报文。这应确保客户端不拒绝新 SecurityToken,这个新 SecurityToken 在客户端接受新 SecurityToken 之前到达。
c) 客户端应接收已期满(超期整个令牌生命周期的 25%之内)的 SecurityToken 所确保安全的报文。这应确保令牌过期之前服务器发送的报文不能由于网络延时而被拒绝。

每个安全通道一直存在直到其被明确地关闭或直到令牌已过期或重叠周期已过去。

OpenSecureChannel 请求和响应报文应使用发送者的证书签名。这些报文总被加密。如果传输层不提供加密,那么这些报文应使用接收者的证书进行加密。

在 OpenSecureChannel 服务中使用的证书是应用实例的证书。客户端和服务器应验证在 CreateSession 和 ActivateSession 服务中使用相同的证书。

5.5.2.2 参数

表 7 定义了该服务的参数。

表 7 OpenSecureChannel 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数。总是忽略 authenticationToken。7.26 描述了 RequestHeader 类型
clientCertificate	ApplicationInstanceCertificate	标识客户端的证书。OpenSecureChannel 请求应使用该证书签名。7.2 中定义了 ApplicationInstanceCertificate 类型
requestType	enum SecurityTokenRequestType	SecurityToken 类型请求: 应是下列枚举值之一: ISSUE_0 为一个新 SecureChannel 创建一个新 SecurityToken。 RENEW_1 为一个现存的 SecureChannel 创建一个新 SecurityToken

表 7（续）

名称	类型	描述
secureChannelId	ByteString	安全通道的标识符，新令牌应属于安全通道。当创建安全通道时，参数为空
securityMode	Enum MessageSecurityMode	适用于该报文的安全类型。 7.14 中定义了 MessageSecurityMode 类型。 即使 securityMode 为 NONE，必须创建安全通道。具体行为取决于所使用的映射，见 IEC 62541-6
securityPolicyUri	String	通过安全通道发送报文安全时，对于 SecurityPolicy 所使用的 URI。已知的 URI 集合以及与其相关的 SecurityPolicies 定义见 IEC 62541-7
clientNonce	ByteString	一个随机数，不应在其他请求中使用它。由于每次新建一个安全通道，每次应产生一个新 clientNonce 。 该参数应与用于对称加密算法密钥具有相同的长度，相应的密钥算法定义在 securityPolicyUr
requestedLifetime	Duration	对于新 SecurityToken 请求的生命周期，以毫秒为单位。它指定客户端何时期望通过再次调用 OpenSecureChannel 服务新建安全通道。如果没有新建安全通道，那么所有使用当前 SecurityToken 发送的报文都应被接收者拒绝
响应		
responseHeader	ResponseHeader	通用响应参数（见 7.27 ResponseHeader 类型定义）
securityToken	ChannelSecurityToken	描述由服务器发布的 SecurityToken
channelId	ByteString	安全通道的惟一标识符。无论何时，只要新建安全通道，应提供该标识符
tokenId	ByteString	通道内单个 SecurityToken 的惟一标识符。该标识符应是与安全报文（由 SecurityToken 确保安全）一起传递的标识符
createdAt	UtcTime	创建 SecurityToken 的时刻
revisedLifetime	Duration	以毫秒为单位的 SecurityToken 的生命周期。计算该令牌的 UTC 期满时间，可以通过将生命周期加上 createdAt 时间
serverNonce	ByteString	一个随机数，不能用于其他请求中。对于每个新建的安全通道产生一个 serverNonce。该参数应与对称加密算法具有相同的长度，该加密算法由 securityPolicyUri 标识

5.5.2.3 服务结果

表 8 定义了该服务特定的服务结果。表 165 定义了公用 *statusCode* 。

表 8　OpenSecureChannel 服务结果代码

标识符	描述
Bad_SecurityChecksFailed	该结果代码描述见表 165
Bad_CertificateTimeInvalid	该结果代码描述见表 165
Bad_CertificateIssuerTimeInvalid	该结果代码描述见表 165
Bad_CertificateHostNameInvalid	该结果代码描述见表 165
Bad_CertificateUriInvalid	该结果代码描述见表 165
Bad_CertificateUseNotAllowed	该结果代码描述见表 165
Bad_CertificateIssuerUseNotAllowed	该结果代码描述见表 165
Bad_CertificateUntrusted	该结果代码描述见表 165
Bad_CertificateRevocationUnknown	该结果代码描述见表 165
Bad_CertificateIssuerRevocationUnknown	该结果代码描述见表 165
Bad_CertificateRevoked	该结果代码描述见表 165
Bad_CertificateIssuerRevoked	该结果代码描述见表 165
Bad_RequestTypeInvalid	安全令牌请求类型无效
Bad_SecurityModeRejected	安全模式不满足服务器设置需求
Bad_SecurityPolicyRejected	安全策略不满足服务器设置需求
Bad_SecureChannelIdInvalid	该结果代码描述见表 165
Bad_NonceInvalid	该结果代码描述见表 165

5.5.3　CloseSecureChannel

5.5.3.1　描述

该服务用来终止安全通道。

请求报文应使用与安全通道当前令牌相关的密钥签名。

5.5.3.2　参数

表 9 定义了该服务的参数。

表 9　CloseSecureChannel 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数。总是忽略 sessionId。7.26 定义了 RequestHeader 类型
secureChannelId	ByteString	待关闭安全通道的标识符
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)

5.5.3.3 服务结果

表 10 定义了该服务特定的服务结果。表 165 中定义了公用 StatusCodes。

表 10 CloseSecureChannel 服务结果代码

象征性 Id	描述
Bad_SecureChannelIdInvalid	这个结果代码描述见表 165

5.6 Session(会话)服务集

5.6.1 概述

该服务集定义了会话上下文中应用层建立连接的服务。

5.6.2 CreateSession

5.6.2.1 描述

OPC UA 客户端使用该服务创建会话。服务器返回惟一标识该会话的两个值。第一值是 sessionId,在审查日志和服务器地址空间中该值用于标识会话。第二个是 authenticationToken,用于使进入的求与会话关联起来。

调用该服务之前,客户端应使用 OpenSecureChannel 服务创建安全通道以确保在会话期间交换消息的完整性。安全通道有一个惟一标识符,服务器应将其与 authenticationToken 联系起来。只有当服务器与用于创建该会话的同一安全通道关联时,它们才可以接收带有用 authenticationToken 的请求。客户端可以通过调用 ActivateSession 方法,将新安全通道与会话关联起来。

安全通道总是由通信栈管理,这意味着通信栈应提供 API,服务器能使用它们发现任何给定请求所使用的 SecureChannel 的信息。通信栈至少应提供安全通道使用的 SecurityPolicy 和 SecurityMode。它也应该提供惟一标识安全通道的 SecureChannelId 或用于建立安全通道的客户端证书。服务器使用其中之一来标识用于发送请求的安全通道。7.29 描述了对于不同类型的通信栈如何创建 authenticationToken。

交换应用实例证书和 Nonce 可能是可选的,签名可能为空,这取决于安全通道的 SecurityPolicy 和 SecurityMode。SecurityPolicies 的定义和这些参数的处理见 IEC 62541-7。

服务器在响应中返回其 EndpointDescriptions。客户端使用该信息确定是否从发现端点上返回与服务器具有的端点匹配的 EndpointDescriptions 列表。如果存在不同,客户端应关闭会话并报告错误。服务器返回请求中客户端指定的 serverUri 的所有 EndpointDescriptions。客户端仅使用 transportProfileUri 去验证 EndpointDescriptions,该 transportProfileUri 与原 GetEndpoints 请求中指定的 profileUri 相匹配。如果由可信任的来源(如管理员)提供,客户端可以跳过该检测。

直到客户端调用 ActivateSession 服务并提供其 SoftwareCertificates,并证明拥有应用实例证书及其提供用户身份令牌,才使用该服务创建的会话。

响应也包含一个标识服务器能力的 SoftwareCertificates 列表。它也包含一个服务器所支持的 OPC UA 行规列表。OPC UA 行规定义见 IEC 62541-7。

可能包含其他组织发行的附加证书用来标识附加的服务器能力。这些行规的例子包含对特定信息模型的支持和对特定设备类型的访问支持。

当创建会话时,服务器在它的 SessionDiagnosticArray 变量中为客户端添加一个条目。该变量的描述见 IEC 62541-5。

创建会话独立于底层通信连接。因此，如果通信连接失败，会话不会立即受到影响。从底层通信连接错误恢复的确切机制取决于 IEC 62541-6 中描述的安全通道映射。

如果客户端不能在超时周期(超时周期由服务器在 CreateSession 服务响应中协商)内在会话上发布一个服务请求，服务器应自动终止会话。这能使服务器避免客户端失败，以及避免无法重建失效的底层连接的情况。客户端应准备按时间要求发出请求，以避免自动关闭会话。客户端可以使用 CloseSession 服务明确地终止会话。

当终止会话时，所有该会话上未解决的请求均被中止并向客户端返回 Bad_SessionClosed 状态代码。此外，服务器从其 SessionDiagnosticArray 变量中删除客户端的输入项并通知订阅此输入项的其他客户端。

如果客户端调用 CloseSession 服务，那么所有与该会话相关的订阅(deleteSubscriptions 标记设置为 TRUE)均被删除。如果服务器由于其他任何原因终止了该会话，与该会话相关的订阅不被删除。为了防止在会话终止的情况下数据丢失，每个订阅有一个自己的生命周期。在这些情况下，该订阅在生命周期期满之前可以重新分配给其他客户端。

某些服务器，如聚合服务器，也当作其他服务器的客户端。这些服务器典型地支持多个系统用户，可作为所代表服务器的代理。从两个层次支持这些服务器的安全性。

首先，每个 OPC UA 服务请求包含一字符串参数，该参数用于携带审查记录 ID。一个客户端或任一作为客户端运作的服务器(如聚合服务器)能为其发出的请求创建一个本地审查日志输入项。该参数允许客户端为请求中的该输入项传递标识符。如果服务器也维护一个审查日志，它能在审查日志输入项中包含该 ID。当检查日志时，发现该输入项，检查者直接将它与创建该输入项的客户端关联起来。这个能力允许对一个系统内的审计日志可追溯。对于审计的附加信息参见 IEC/TR 62541-2。维护审查日志的服务器通过该标准定义的事件报文提供审查日志输入项的信息。服务器可以选择通过事件报文仅提供审查信息。审查 EventType 定义在 IEC 62541 3 中。

其次，这些聚合服务器可以打开与下层服务器独立的会话，每个客户端能从下层服务器访问数据。图 14 给出了这个概念。

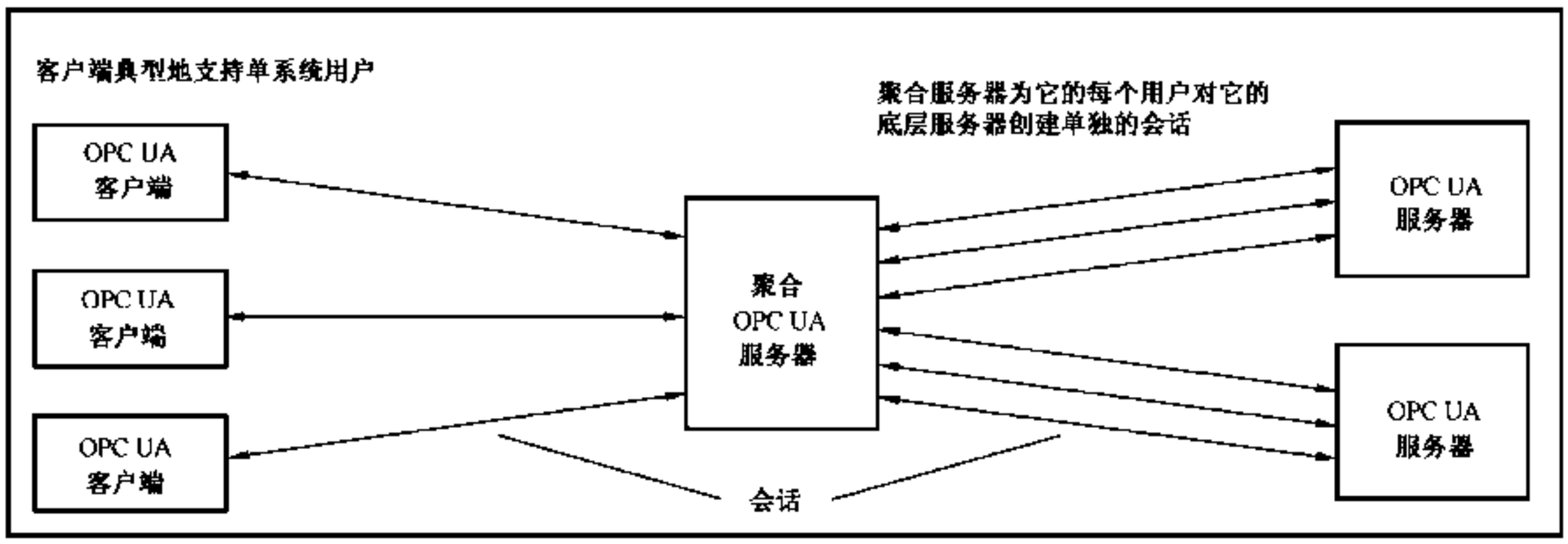

图 14　会话中的多用户

5.6.2.2　参数

表 11 定义了该服务的参数。

表 11　CreateSession 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数。AuthenticationToken 总是被忽略。类型 RequestHeader 在 7.26 中定义
clientDescription	ApplicationDescription	描述客户端应用的信息。 类型 ApplicationDescription 在 7.1 中定义
serverUri	String	此值仅当 EndpointDescription 配有 gatewayServerUri 时被指定。该值是源自 EndpointDescription 的 applicationUri，而 EndpointDescription 是用于底层服务器的 applicationUri。类型 EndpointDescription 在 7.9 中有定义
endpointUrl	String	客户用于访问会话端点的网络地址。URL 的主机名部分应当是服务器应用实例证书(见 7.2)中指明的应用的主机名中的一个。 当 URL 不匹配服务器的主机名时，服务器应发起一个 ApuditUrlMismatchEventType 事件。ApuditUrlMismatchEventType 事件类型在 IEC 62541-5 中定义。 服务器将这条信息用于诊断，并确定在响应中返回的 EndpointDescription 集合
sessionName	String	用于标识会话的可读字符串。为了诊断目的，服务器使该名称和 sessionID 在其地址空间中可见。客户端应提供对于客户端实例来说惟一的名称，如果该参数未被指定，那么服务器应分配一个值
clientNonce	ByteString	不应被用于任何其他请求的随机数。该数最小长度应为 32 字节。行规可能增加要求的长度。服务器应使用该值来证明响应中其应用实例证书的拥有权
clientCertificate	ApplicationInstanceCertificate	颁发给客户的应用实例证书。 ApplicationInstanceCertificate 类型在 7.2 中定义
RequestedSessionTimeout	Duration	没有活动，会话仍能保持开放所要求的最大 ms 数。如果客户端未能在此时间间隔中发出一个服务请求，那么服务器应自动终止客户端会话
maxResponseMessageSize	UInt32	任何响应消息主体的最大尺寸，以字节为单位。如果响应消息超过了该限制，服务器应返回 Bad_ResponseTooLarge 的服务故障。 此值为零时表示不使用该参数，5.3 提供了关于该参数使用的更多信息
响应		
responseHeader	ResponseHeader	公共响应参数(ResponseHeader 类型见 7.27)

表 11（续）

名称	类型	描述
sessionId	NodeId	服务器分配给会话的惟一 NodeId。该标识符用于访问服务器地址空间中会话的诊断信息。它也可以用在审核日志和报告有关会话信息的任何事件。会话的诊断信息在 IEC 62541 5 中描述。审核日志及其相关事件在 6.2 中描述
authenticationToken	SessionAuthenticationToken	服务器分配给会话的惟一标识符。该标识符应在每个请求的 RequestHeader 中被传递，并与 SecureChannelId 一起使用，以确定客户端是否可以访问会话。该标识不应以客户端或服务器有可能与以前的或现有会话混淆的方式重用。 SessionAuthenticationToken 类型在 7.29 中描述
revisedSession Timeout	Duration	没有活动，会话仍能保持开放的实际最大 ms 数。会话应保持开放没有活动的实际最大毫秒数。服务器应尽量满足该参数的客户端请求，但可以向上或向下修改该值以满足其自身的约束
serverNonce	ByteString	不应被用于任何其他请求的随机数。 该数字最小长度应为 32 字节。 客户应使用该值来证明在 ActivateSession 请求中其应用实例证书的拥有权。 该值也可被用于证明在 ActivateSession 请求中指定的 userIdentityToken 的拥有权
serverCertificate	ApplicationInstanceCertifacate	颁发给服务器的应用实例证书。 服务器应使用私钥签名客户端在请求中提供的随机数来证明拥有权。客户应验证该证书是否与用于创建安全通道的证书一致。 ApplicationInstanceCertificate 类型在 7.2 中定义
serverEndpoints []	Endpoint Description	服务器支持的端点列表。 服务器应返回一组用于在请求中指定的 serverUri 的 EndpointDescription。EndpointDescription 类型在 7.9 中定义。如果客户端使用一个发现端点来获取 EndpointDescription，那么客户端应与来自发现端点的列表来验证该列表
serverSoftwareCertificates []	SignedSoftwareCertificate	已颁布给服务器应用的软件证书。 软件证书中的 productUri 应与客户端连接到服务器时所使用 EndpointDescription 中的 productUri 相匹配。不匹配 procutUri 的证书应被忽略。如果客户端不满意服务器提供的软件证书，应调用 CloseSession。 SignedSoftwareCertificate 类型在 7.31 中定义

表 11（续）

名称	类型	描述
serverSignature	SignatureData	服务器证书相关私钥生成的签名。该参数的计算方法是追加客户端随机数到客户证书并在结果字节序列签名。 签名算法应为端点安全策略中指定的非对称签名算法。 SignatureData 类型在 7.30 中定义
maxRequestMessageSize	UInt32	任何请求消息主体的最大尺寸，以字节为单位。 如果请求消息超过此限制，客户端通信栈应向应用返回 Bad_RequestTooLarge 错误。 零值表示不使用该参数。 5.3 提供了使用该参数更多的信息

5.6.2.3 服务结果

表 12 定义此服务特定的服务结果。表 165 定义了常用状态代码。

表 12 CreateSession 服务结果代码符号

标识符	描述
Bad_SecureChannelIdInvalid	对于该结果代码的描述见表 165
Bad_NonceInvalid	对于该结果代码的描述见表 165
Bad_SecurityChecksFailed	对于该结果代码的描述见表 165
Bad_CertificateTimeInvalid	对于该结果代码的描述见表 165
Bad_CertificateIssuerTimeInvalid	对于该结果代码的描述见表 165
Bad_CertificateHostNameInvalid	对于该结果代码的描述见表 165
Bad_CertificateUriInvalid	对于该结果代码的描述见表 165
Bad_CertificateUseNotAllowed	对于该结果代码的描述见表 165
Bad_CertificateIssuerUseNotAllowed	对于该结果代码的描述见表 165
Bad_CertificateUntrusted	对于该结果代码的描述见表 165
Bad_CertificateRevocationUnknown	对于该结果代码的描述见表 165
Bad_CertificateIssuerRevocationUnknown	对于该结果代码的描述见表 165
Bad_CertificateRevoked	对于该结果代码的描述见表 165
Bad_CertificateIssuerRevoked	对于该结果代码的描述见表 165
Bad_TooManySessions	服务器已达到其最大会话数
Bad_ServerUriInvalid	对于该结果代码的描述见表 165

5.6.3 ActivateSession

5.6.3.1 描述

客户端利用该服务提交其软件证书给服务器用以验证，并指定与会话相关联的用户的身份。该服务请求应由客户端在产生会话后并在其发出任何其他服务请求前发出。否则将导致服务器关闭会话。

当客户端调用该服务时，客户端应证明其与调用 CreateSession 服务是相同的应用，客户通过创建与 CreateSession 请求中指定的客户端证书相关的私钥签名来完成这一过程。通过追加由服务器提供的上一个 serverNonce 给服务器证书，并计算结果字节序列的签名，来创建该签名。

serverNonce 一旦被使用，便不能被重用。因此，每次调用 ActivateSession 服务时，服务器应返回一个新的 serverNonce。

当首次调用 ActivateSession 服务时，如果安全通道与 CreateSession 请求相关的安全通道不相同，那么服务器应拒绝该请求。ActivateSession 的后续调用，可能与不同的安全通道相关。这种情况下，服务器应验证客户端用于创建新的安全通道的证书，是否与用于创建原始的安全通道的证书相同。此外，服务器应验证客户端提供的 UserIdentityToken，是否与会话当前相关的令牌相同。一旦服务器接受了新的安全通道，它应拒绝通过旧的安全通道发送的请求。

ActivateSession 服务用于将会话和用户身份相关联。当客户端提供了一个用户身份，那么它应提供被授权使用该用户身份的证明。用于提供证明的具体机制，取决于 UserIdentityTokey 的类型。如果令牌是 UserNameIdentityTokey，那么证明是令牌中包含的密码。如果令牌是 X509IdentityToken，那么证明则是与证书相关的私钥生成的签名。签名数据通过追加最后的服务随机数到 CreateSession 响应中指定的服务器证书来创建，如果令牌包含一个机密，那么就应该使用来自服务器证书的公钥进行加密。

客户端可以通过调用 ActivateSession 服务来更改与会话相关的用户身份。服务器验证请求提供的签名，然后验证新用户的身份。如果没有错误发生，服务器替换会话的用户身份。更改会话的用户身份，可能导致活动订阅的不连续性，因为服务器可能不得不断开与底层系统的连接，并使用新的凭证重新建立连接。

当客户端在请求中提供地区 Id 列表时，每个地区 Id 都应包含语言组件。它可以选择性地包含＜country/region＞ 组件。服务器返回响应，它也可能返回语言和国家/地区或只是作为它的缺省地区 Id 的语言。

当服务器向客户端返回一个字符串，它首先确定是否有可用的翻译，如果有，服务器返回地区 Id 与在客户端提供的清单中优先级最高的地区 Id 完全匹配的字符串。

如果没有精确的匹配，服务器会忽略地区 Id 的＜country/region＞组件，并返回＜language＞组件与客户端提供的清单中优先级最高的地区 Id 的＜ language ＞组件匹配的字符串。

如果仍然没有匹配，服务器返回包含该地区 Id 的字符串。

当网关服务器连接到底层的服务器时，它被期望模仿由客户端提供的用户。这意味着它应当使用底层服务器提供的随机数对 UserIdentityToken 重新计算签名。如果客户端提供的 UseridentityToken 不支持模拟，网关服务器将不得不使用自己的用户凭据。

5.6.3.2 参数

表 13 定义了服务参数。

表 13　ActivateSession 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数。类型 RequestHeader 在 7.26 中定义
clientSignature	SignatureData	与客户端证书相关的私钥生成的签名。 签名算法应为端点安全策略中指定的非对称签名算法。SignatureData 类型在 7.30 中定义
clientSoftwareCertificates []	SignedSoftwareCertificate	客户端应用已颁布的软件证书。 在软件证书包含的 productUri 应匹配客户端在 CreateSession 请求中传递的 ApplicationDescription 中的 productUri。不匹配 productUri 的证书应被忽略。 如果服务器不满意客户端提供的软件证书，那么它可能拒绝来自客户端的连接。 此参数只需要在为单一应用会话的首次调用 ActivateSession 时指定。 SignedSoftwareCertificate 类型在 7.31 中定义
localeIds []	LocaleId	地区 Id 用于本地化的字符串的优先顺序列表。在列表中的第一个 localeId 具有最高的优先权。如果服务器返回给客户端本地化的字符串，那么服务器应返回它可以翻译的最高优先级。如果此列表中的任一地区 Id 都没有翻译，那么它应返回包含地区 Id 的字符串值。地区 Id 的更多细节见 IEC 62541 3。如果客户端未能指定至少一个地区 Id，那么服务器应使用它有的任何一个地区 Id。 此参数只需要在单一应用会话的首次调用 ActivateSession 时指定。如果没有指定，服务器应继续使用会话当前的 localeId
userIdentityToken	扩展参数 UserIdentityToken	与客户端应用相关的用户凭据。服务器使用这些凭据来确定是否允许客户端激活一个会话，以及客户端在此次会话期间可以访问哪些资源。 UserIdentityToken 是在 7.35 中定义的可扩展参数类型。 EndpointDescription 指定服务器应接受何种 UserIdentityToken
userTokenSignature	SignatureData	如果客户端指定支持数字签名的用户身份令牌，那么它应创建一个签名，并作为其参数传递。否则该参数被忽略。 签名算法取决于身份令牌类型。SignatureData 类型在 7.30 中定义
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)
serverNonce	ByteString	不应被用于任何其他请求的随机数。这个数最小长度 32 字节。 客户端应使用此值在下次调用 ActivateSession 请求时证明其应用实例证书的持有
results []	StatusCode	软件证书的验证结果列表(StatusCode 定义见 7.33)
diagnosticInfos []	DiagnosticInfo	软件证书验证错误的诊断信息列表(DiagnosticInfo 定义见 7.8)。 如果在请求首部中没有要求诊断信息，或者在请求处理中没有遇到诊断信息，那么此列表为空

5.6.3.3 服务结果

表 14 定义此服务特定的服务结果。表 165 中定义了常用状态代码。

表 14 ActivateSession 服务的结果代码

标识符	描述
Bad_IdentityTokenInvalid	对于该结果代码的描述见表 165
Bad_IdentityTokenRejected	对于该结果代码的描述见表 165
Bad_UserAccessDenied	对于该结果代码的描述见表 165
Bad ApplicationSignatureInvalid	由客户端应用提供的签名丢失或无效
Bad_UserSignatureInvalid	用户令牌签名丢失或无效
Bad_NoValidCertificates	客户端没有提供至少一个有效并满足服务器行规要求的软件证书
Bad_IdentityChangeNotSupported	服务器不支持更改分配给会话的用户身份

5.6.4 CloseSession

5.6.4.1 描述

使用此服务终止会话。当服务器接收到 CloseSession 请求时会采取以下措施：

1) 停止接受会话请求。从会话收到的所有后续请求被丢弃。

2) 向当前未完成的所有请求返回否定响应 StatusCode Bad_SessionClosed，以保证对 CloseSession 响应的及时反应。

3) 为客户在其 SessionDiagnosticArray 变量中删除登录项。

5.6.4.2 参数

表 15 定义了用于该服务的参数。

表 15 CloseSession 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
deleteSubscriptions	Boolean	如果该值为 TRUE，那么服务器应删除所有与会话相关的订阅。如果该值为 FALSE，那么服务器应根据其时限保留与会话相关的订阅，直到他们超时
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)

5.6.4.3 服务结果

表 16 定义此项服务特定的服务结果。表 165 定义了常用状态代码。

表 16 CloseSession 服务结果代码

标识符	描述
Bad_SessionIdInvalid	对于该结果代码的描述见表 165

5.6.5 Cancel

5.6.5.1 描述

该服务用于取消未完成的服务请求。成功取消服务请求应以 Bad_RequestCancelledByClient 作为响应。

5.6.5.2 参数

表 17 定义了该服务所用的参数。

表 17 Cancel 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
requestHandle	IntegerId	分配给一个或多个应取消的请求的 requestHandle。 应取消所有匹配 requestHandle 的未完成请求
响应		
responseHeader	ResponseHeader	公共响应参数(ResponseHeader 定义见 7.27)
cancelCount	UInt32	被取消的请求的数目

5.6.5.3 服务结果

常用状态代码在表 165 中定义。

5.7 NodeManagement(节点管理)服务集

5.7.1 概述

该服务集定义了添加和删除地址空间节点及其之间引用的服务。即使创建节点的客户端已从服务器断开,然而所有添加的节点仍在地址空间中持续存在。

5.7.2 AddNode

5.7.2.1 描述

此服务用于添加一个或多个节点到地址空间层次。使用该服务,每个节点作为分层引用(HierarchicalReference)的目标节点添加,以确保地址空间完全连接并且节点作为子节点添加到地址空间层次(见 IEC 62541-3)。

5.7.2.2 参数

表 18 定义了该服务所用的参数。

表 18　AddNodes 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
nodesToAdd []	AddNodesItem	添加的节点列表。所有节点使用层次引用类型作为引用被添加到现有的节点
parentNodeId	ExpandedNodeId	用于引用的父节点的 ExpandedNodeId。ExpandedNodeId 类型在 7.10 中定义
referenceTypeId	NodeId	用于从父节点到新节点的分层引用类型的 NodeId
requestedNewNodeId	ExpandedNodeId	客户端请求的新添加节点的 ExpandedNodeId。ExpandedNodeId 中的 ServerIndex 应为 0。 如果服务器不能使用该 NodeId，那么它拒绝该节点，并返回相应的错误代码。 如果客户端不希望请求一个 NodeId，那么它设置该参数值为空 ExpandedNodeId。 如果添加的节点是一个引用类型的节点，那么其 NodeId 应为一个数字 Id。引用类型的 NodeId 的描述见 IEC 62541 3
browseName	QualifiedName	添加节点的浏览名称
nodeClass	NodeClass	添加节点的节点类
nodeAttributes	可扩展参数 NodeAttributes	专用于节点类的属性。NodeAttributes 参数类型是 7.18 中规定的一个扩展参数类型。 客户端允许省略部分或全部属性值。如果一个属性值被省略，那么服务器应使用 TypeDefinitionNode 的缺省值。如果未提供 TypeDefinitionNode，服务器应选择一个合适的缺省值。 服务器仍然可以添加一个带有适当缺省值的可选属性到该节点，即使客户未指定一个值
typeDefinition	ExpandedNodeId	新添加节点的 TypeDefinitionNode 的 NodeId。对非对象和变量的所有节点类，该参数应为空值。在对象和变量节点类的情况下，应当提供该值
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 的定义见 7.27)
results []	AddNodesResult	新添加节点的结果列表。列表的大小和顺序匹配 nodesToAdd 参数的大小和顺序
statusCode	StatusCode	新添加节点的 StatusCode(StatusCode 定义见 7.33)
addedNodeId	NodeId	服务器分配的新添加节点的 NodeId。如果操作失败，则为空 NodeId
diagnosticInfos []	DiagnosticInfo	新添加节点的诊断信息列表(DiagnosticInfo 的定义见 7.8)。列表的大小和顺序匹配 nodesToAdd 请求参数的大小和顺序。如果在请求首部中没有请求诊断信息，或者在请求处理中没有遇到诊断信息，那么该列表为空

5.7.2.3 服务结果

表 19 定义了该服务特定的服务结果。表 165 定义了常用状态代码。

表 19 添加节点的服务结果代码

标识 ID	描述
Bad_NothingToDo	对于该结果代码的描述见表 165
Bad_TooManyOperations	对于该结果代码的描述见表 165

5.7.2.4 状态代码

表 20 定义了该服务特定的操作层面状态代码参数值。表 166 定义了常用状态代码。

表 20 添加节点的操作层面结果代码

标识 ID	描述
Bad_ParentNodeIdInvalid	ParentNodeId 未指向一个有效节点
Bad_ReferenceTypeIdInvalid	对于该结果代码的描述见表 166
Bad_ReferenceNotAllowed	引用无法创建，因为它违反了数据模型的约束
Bad_NodeIdRejected	所请求的 NodeId 被拒绝，因为它是无效的，或服务器不允许客户端指定节点 Id
Bad_NodeIdExists	所请求的 NodeId 已被另一个节点使用
Bad_NodeClassInvalid	对于该结果代码的描述见表 166
Bad_BrowseNameInvalid	对于该结果代码的描述见表 166
Bad_BrowseNameDuplicated	共享相同的父关系的节点间的浏览名称不惟一
Bad_NodeAttributesInvalid	节点的属性是无效的节点类
Bad_TypeDefinitionInvalid	对于该结果代码的描述见表 166
Bad_UserAccessDenied	对于该结果代码的描述见表 165

5.7.3 AddReferences

5.7.3.1 描述

该服务用于添加一个或多个引用到一个或多个节点。节点类是用于验证要添加的引用是否与目标节点的节点类相匹配的输入参数。如果引用指向远程服务器中的目标节点，那么该参数将不被验证。

在某些情况下，增加新的引用到地址空间，需要该服务器添加新的服务器 Id 到服务器的 ServerTable 变量。因此，远程服务器通过其 URI 而不是 ServerTable 索引进行标识。这允许服务器添加远程服务器的 URI 到其 ServerTable。

5.7.3.2 参数

表 21 定义了该服务的参数。

表 21　AddReferences 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
referencesToAdd [ ]	AddReferencesItem	添加到源节点的引用实例列表。列表中的每个引用的目标节点类应匹配目标节点的节点类
sourceNodeId	NodeId	要添加引用的节点的 NodeId。源节点应总是存在于服务器中用以添加引用。如果目标节点在本地服务器且源节点在远程服务器,那么 isForward 参数可以设置为 FALSE
referenceTypeId	NodeId	定义引用的引用类型的 NodeId
isForward	Boolean	如果该值为 TURE,那么服务器创建一个前向引用。如果该值为 FALSE,那么服务器创建一个反向引用
targetServerUri	String	远程服务器的 URI。如果该参数不为空,那么它覆盖目标 NodeId 中的 serverIndex
targetNodeId	ExpandedNodeId	目标节点的 ExpandedNodeId。ExpandedNodeId 类型在 7.10 中定义
targetNodeClass	NodeClass	目标节点的 NodeClass。客户应指定该值,因为目标节点可能无法被服务器直接访问
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)
results []	StatusCode	需添加的引用的 StatusCode 列表(StatusCode 定义见 7.33)。列表的大小和顺序应匹配 referencesToAdd 请求参数的大小和顺序
diagnosticInfos [ ]	DiagnosticInfo	需添加的引用的诊断信息列表(DiagnosticInfo 的定义见 7.8)。列表的大小和顺序应匹配 referencesToAdd 请求参数的大小和顺序。如果在请求首部中没有请求诊断信息,或者如果在请求处理中没有遇到诊断信息,那么该列表为空

5.7.3.3　服务结果

表 22 定义了该服务特定的服务结果。表 165 定义了常用状态代码。

表 22　AddReferences 服务结果代码

标识符	描述
Bad_NothingToDo	对于该结果代码的描述见表 165
Bad_TooManyOperations	对于该结果代码的描述见表 165

5.7.3.4　状态代码

表 23 定义了该服务特定的结果参数值。常用状态代码定义见表 166。

表 23　AddReferences 操作层面结果代码

标识符	描述
Bad_SourceNodeIdInvalid	对于该结果代码的描述见表 166
Bad_ReferenceTypeIdInvalid	对于该结果代码的描述见表 166
Bad_ServerUriInvalid	对于该结果代码的描述见表 165
Bad_TargetNodeIdInvalid	对于该结果代码的描述见表 166
Bad_NodeClassInvalid	对于该结果代码的描述见表 166
Bad_ReferenceNotAllowed	引用无法创建，因为它违反了该服务器上的数据模型强制的约束
Bad_ReferenceLocalOnly	该引用类型对于远程服务器的引用无效
Bad_UserAccessDenied	对于该结果代码的描述见表 165
Bad_DuplicateReferenceNotAllowed	节点间的引用类型已被定义
Bad_InvalidSelfReference	服务器不允许对该节点进行这种类型的自引用

5.7.4　DeleteNodes

5.7.4.1　描述

该服务用于从地址空间删除一个或多个节点。

当调用该服务删除的节点之一是一个引用的目标节点时，那么这些引用在 deleteTargetReferences 参数的基础上将无法解决。

当调用该服务删除的节点之一正在被监视时，那么含有状态代码 Bad_NodeIdUnknown 的通知将发送到监视客户端以指示该节点已被删除。

5.7.4.2　参数

表 24 定义了用于该服务的参数。

表 24　DeleteNodes 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
nodesToDelete []	DeleteNodesItem	删除节点的列表
nodeId	NodeId	删除节点的 NodeId
deleteTargetReference	Boolean	具有下列值的 Boolean 参数： TRUE　删除在目标节点中引用了被删除节点的引用。 FALSE　仅删除被删除节点为源节点的引用。 如果此参数为 TRUE，服务器不能保证它能删除所有目标引用
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)

表 24（续）

名称	类型	描述
results [ ]	StatusCode	需删除节点的 StatusCode 列表（StatusCode 定义见 7.33）。列表的大小和顺序与 nodesToDelete 请求参数的大小和顺序匹配
diagnosticInfos [ ]	DiagnosticInfo	需删除节点的诊断信息列表（DiagnosticInfo 定义见 7.8）。列表的大小和顺序与 nodesToDelete 请求参数的大小和顺序匹配。如果在请求首部中没有请求诊断信息，或者如果在请求处理中没有遇到诊断信息，那么该列表为空

5.7.4.3 服务结果

表 25 定义了该服务特定的服务结果。表 165 定义了常用状态代码。

表 25 DeleteNodes 服务结果代码

标识 ID	描述
Bad_NothingToDo	对于该结果代码的描述见表 165
Bad_TooManyOperations	对于该结果代码的描述见表 165

5.7.4.4 状态代码

表 26 定义了该服务特定的结果参数值。常用状态代码定义见表 166。

表 26 删除节点的操作层面结果代码

标识 ID	描述
Bad_NodeIdInvalid	对于该结果代码的描述见表 166
Bad_NodeIdUnknown	对于该结果代码的描述见表 166
Bad_UserAccessDenied	对于该结果代码的描述见表 165
Bad_NoDeleteRights	对于该结果代码的描述见表 166
Uncertain_ReferenceNotDeleted	服务器不能删除所有目标引用

5.7.5 DeleteReferences

5.7.5.1 描述

该服务用于删除节点的一个或多个引用。

当调用该服务删除的任何引用包含在一个视图中，那么当支持 ViewVersion 属性时应更新该属性。

删除引用应触发 ModelChange 事件。

5.7.5.2 参数

表 27 定义了该服务的参数。

表 27 DeleteReferences 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
referencesToDelete []	DeleteReferencesItem	被删除的引用列表
sourceNodeId	NodeId	包含需删除的引用的节点的 NodeId
referenceTypeId	NodeId	定义需删除的引用的引用类型的 NodeId
isForward	Boolean	如果该值为 TRUE,那么服务器将前向引用。如果该值为 FALSE,那么服务器将删除反向引用
targetNodeId	ExpandedNodeId	引用的目标节点的 NodeId。 如果服务器索引表明,目标节点是一个远程节点,那么 NodeId 应包含绝对的命名空间 URI。如果目标节点是一个本地节点,那么 NodeId 应包含命名空间索引
deleteBidirectional	Boolean	具有下列值的 Boolean 参数: TRUE 从目标节点删除指定的引用及其反向引用。如果目标节点位于远程服务器上,那么服务器只允许删除指定的引用。 FALSE 只删除指定的引用
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)
results []	StatusCode	需删除引用的 StatusCodes 列表(StatusCode 定义见 7.33)。列表的大小和顺序匹配 referencesToDelete 请求参数的大小和顺序
diagnosticInfos []	DiagnosticInfo	需删除引用的诊断信息列表(DiagnosticInfo 定义见 7.8)。列表的大小和顺序匹配 referencesToDelete 请求参数的大小和顺序。如果在请求首部中没有请求诊断信息,或者如果在请求处理中没有遇到诊断信息,那么该列表为空

5.7.5.3 服务结果

表 28 定义了该服务特定的服务结果。表 165 定义了常用状态代码。

表 28 DeleteReferences 服务结果代码

标识符	描述
Bad_NothingToDo	对于该结果代码的描述见表 165
Bad_TooManyOperations	对于该结果代码的描述见表 165

5.7.5.4 状态代码

表 29 定义了该服务特定的结果参数值。表 166 定义了常用状态代码。

表 29　DeleteReferences 操作层面结果代码

标识符	描述
Bad_SourceNodeIdInvalid	对于该结果代码的描述见表 166
Bad_ReferenceTypeIdInvalid	对于该结果代码的描述见表 166
Bad_ServerIndexInvalid	服务器索引无效
Bad_TargetNodeIdInvalid	对于该结果代码的描述见表 166
Bad_UserAccessDenied	对于该结果代码的描述见表 165
Bad_NoDeleteRights	对于该结果代码的描述见表 166

5.8　View(视图)服务集

5.8.1　概述

客户端使用 View 服务集的浏览服务查询全部地址空间,或查询作为地址空间子集的一个 View。

View 是由服务器创建的地址空间的一个子集。本部分的未来版本也会定义服务来创建客户端定义的 View。地址空间中视图组织的描述见 IEC 62541-5。

5.8.2　Brower

5.8.2.1　描述

该服务用于发现指定节点的引用。View 的使用可以进一步限制浏览。该 Brower 服务还支持原始的过滤能力。

5.8.2.2　参数

表 30 定义了该服务的参数。

表 30　Browse 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
view	ViewDescription	被浏览的视图的描述(ViewDescription 定义见 7.37)。一个空 ViewDescription 值表示整个地址空间。空 ViewDescription 值的使用导致 nodeToBrowse 的所有引用被返回。任何其他 View 的使用只会导致该视图定义的 nodeToBrowse 的引用被返回
requestedMax ReferencesPerNode	Counter	表示对于请求中指定的每个开始节点,返回的引用的最大数量。值 0 表示客户没有施加任何限制(Counter 定义见 7.5)
nodesToBrowse[ ]	BrowerDescription	被浏览的节点列表
nodeId	NodeId	被浏览节点的 NodeId。被传递的 nodeToBrowse 应为被传递视图的一部分

表 30（续）

<table>
<tr><th>名称</th><th>类型</th><th>描述</th></tr>
<tr><td>browseDirection</td><td>enumBrowerDirection</td><td>列举指定引用方向的枚举值。它具有以下值：
FORWARD_0　只选择前向引用。
INVERSE_1　只选择反向引用。
BOTH_2　选择前向和反向引用。
返回的引用指明了 ReferenceDescription 的 IsForward 参数中服务器遵循的方向。
对称引用常被认为是正方向，因此 IsForward 标志总是被设置为 TRUE；并且如果 browerDirection 设置为 INVERSE_1，对称引用不返回</td></tr>
<tr><td>referenceTypeId</td><td>NodeId</td><td>指定需遵循的引用类型的 NodeId。只返回该引用类型或其子类型的实例。
如果没有指定，那么所有引用被返回</td></tr>
<tr><td>includeSubtypes</td><td>Boolean</td><td>指出引用类型的子类型是否应包括在浏览中。如果为 TURE，那么返回 referenceTypeId 及其所有子类型的实例</td></tr>
<tr><td>nodeClassMask</td><td>UInt32</td><td>指定目标节点的节点类。只返回选定的节点类的目标节点。节点类被分配以下位：
<table>
<tr><th>位</th><th>节点类</th></tr>
<tr><td>0</td><td>对象</td></tr>
<tr><td>1</td><td>变量</td></tr>
<tr><td>2</td><td>方法</td></tr>
<tr><td>3</td><td>对象类型</td></tr>
<tr><td>4</td><td>变量类型</td></tr>
<tr><td>5</td><td>引用类型</td></tr>
<tr><td>6</td><td>数据类型</td></tr>
<tr><td>7</td><td>视图</td></tr>
</table>
如果设置为零，则返回所有节点类</td></tr>
<tr><td>resultMask</td><td>UInt32</td><td>指定应返回引用描述结构的字段。字段分配为以下位：
<table>
<tr><th>位</th><th>结果</th></tr>
<tr><td>0</td><td>引用类型</td></tr>
<tr><td>1</td><td>是否前向
(IsForward)</td></tr>
<tr><td>2</td><td>节点类</td></tr>
<tr><td>3
4</td><td>浏览名称
显示名称</td></tr>
<tr><td>5</td><td>类型定义</td></tr>
</table>
ReferenceDescription 类型在 7.24 中定义</td></tr>
</table>

表 30（续）

名称	类型	描述
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)
results []	BrowerResult	BrowseResults 的列表。列表的大小和顺序匹配该请求中指定的 nodesToBrowse 的大小和顺序。 BrowseResult 类型在 7.3 中定义
diagnosticInfos []	DiagnosticInfo	结果的诊断信息列表(DiagnosticInfo 定义见 7.8)。列表的大小和顺序匹配结果响应参数的大小和顺序。如果在请求首部中没有请求诊断信息，或者如果在请求处理中没有遇到诊断信息，那么该列表为空

5.8.2.3 服务结果

表 31 定义了该服务特定的服务结果。表 165 定义了常用状态代码。

表 31 Browse 服务结果代码

符号 ID	描述
Bad_ViewIdUnknown	视图 Id 未指向有效的视图节点
Bad_ViewTimestampInvalid	该结果代码的描述见表 165
Bad_ViewParameterMismatchInvalid	该结果代码的描述见表 165
Bad_ViewVersionInvalid	该结果代码的描述见表 165
Bad_NothingToDo	该结果代码的描述见表 165
Bad_TooManyOperations	该结果代码的描述见表 165

5.8.2.4 状态代码

表 32 定义了该服务特定的结果参数值。表 166 定义了常用状态代码。

表 32 Browse 操作层面结果代码

符号 ID	描述
Bad_NodeIdInvalid	该结果代码的描述见表 166
Bad_NodeIdUnknown	该结果代码的描述见表 166
Bad_ReferenceTypeIdInvalid	该结果代码的描述见表 166
Bad_BrowseDirectionInvalid	该结果代码的描述见表 166
Bad_NodeNotInView	该结果代码的描述见表 166
Bad_NoContinuationPoints	该结果代码的描述见表 166
Uncertain_NotAllNodesAvailable	由于子系统不可用，Browse 结果可能是不完整的

5.8.3 BrowseNext

5.8.3.1 描述

该服务用于请求 Browse 或 BrowseNext 响应信息的下一个集合，该响应信息太大以致无法在单个响应中发送。这里"太大"是指该服务器无法返回更大的响应，或返回结果的数量超过原始浏览请求中由客户端指定的返回结果最大数量。BrowseNext 应在用于提交正进行中的 Browse 或 BrowseNext 的同一会话上提交。

5.8.3.2 参数

表 33 定义了该服务的参数。

表 33 BrowseNext 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
releaseContinuation-Points	Boolean	具有下列值的 Boolean 参数： TRUE 被传递的 continuationPoints 应被重置释放服务器上的资源。 FALSE 被传递的 continuationPoints 应用于获得下一组浏览信息。 客户端应总是使用 Browse 或 BrowseNext 响应返回的延续点，来释放服务器上延续点的资源。如果客户端不希望得到下一组浏览信息，那么 BrowseNext 应被调用并将该参数设置为 TRUE
continuationPoints []	ContinuationPoint	表示延续点的服务器定义的难理解值的列表。在先前 Browse 或 BrowseNext 响应中，延续点的值返回到客户端。使用这些值来识别先前处理的正在进行的 Browse 或 BrowseNext 请求，以及结果集中浏览响应继续的起始点。客户端可以混用来自不同 Browse 或 BrowseNext 响应的延续点。 ContinuationPoint 类型在 7.6 中描述
响应		
responseHeader	ResponseHeader	公共响应参数(ResponseHeader 定义见 7.27)
results []	BrowseResult	符合原始 Browse 请求中指定标准的引用列表。此列表的大小和顺序匹配 continuationPoints 请求参数的大小和顺序。 BrowseResult 类型在 7.3 中定义
diagnosticInfos []	DiagnosticInfo	结果诊断信息列表(DiagnosticInfo 定义见 7.8)。列表的大小和顺序匹配结果响应参数的大小和顺序。如果在请求首部中没有请求诊断信息，或者如果在请求处理中没有遇到诊断信息，则该列表为空

5.8.3.3 服务结果

表 34 定义了该服务特定的服务结果。表 165 定义了常用状态代码。

表 34 BrowseNext 服务结果代码

标识符	描述
Bad_NothingToDo	对于该结果代码的描述见表 165
Bad_TooManyOperations	对于该结果代码的描述见表 165

5.8.3.4 状态代码

表 35 定义了该服务特定的结果参数值。表 166 定义了常用状态代码。

表 35 Browsenext 操作层面结果代码

标识符	描述
Bad_NodeIdInvalid	该结果代码的描述见表 166
Bad NodeIdUnknown	该结果代码的描述见表 166
Bad_ReferenceTypeIdInvalid	该结果代码的描述见表 166
Bad_BrowseDirectionInvalid	该结果代码的描述见表 166
Bad_NodeNotInView	该结果代码的描述见表 166
Bad_ContinuationPointInvalid	该结果代码的描述见表 166

5.8.4 TranslateBrowsePathsToNodeIds

5.8.4.1 描述

该服务用于服务器将一个或多个浏览路径转换为 NodeId 的请求。每个浏览路径由一个起始节点和一个相对路径(RelativePath)组成。指定的起始节点标识该相对路径是基于哪个节点。相对路径包含一系列引用类型和浏览名称。

该服务的目的之一是允许对类型定义编程。由于浏览名称在类型定义中应是惟一的,所以客户端可以为类型定义创建一个有效的浏览路径,并在类型实例中使用该路径。例如,对象类型"锅炉"可能有一个"热传感器"变量作为实例声明。"锅炉"的图形元素编程可能需要显示"热传感器"的值。如果该图形元素被称为"锅炉 1","锅炉"的一个实例,那么它需要调用该服务,指定"锅炉 1"的 NodeId 为起始节点,"热传感器"的浏览名称为浏览路径。该服务将返回"锅炉 1"的"热传感器"的 NodeId,并且该图形元素可以订阅其 Value 属性。

如果一个节点有多个目标具有相同的浏览名称,那么服务器应返回一个 NodeId 列表。然而,由于该服务的主要目的之一是支持对类型定义的编程,所以基于起始节点的类型定义的节点的 NodeId 在列表中是返回的第一个 NodeId。

5.8.4.2 参数

表 36 定义了该服务的参数。

表 36 TranslateBrowsePathsNodeId 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
browsePaths []	BrowsePath	正被请求的 NodeId 的浏览路径列表
startingNode	NodeId	浏览路径的起始节点的 NodeId
relativePath	RelativePath	从起始节点开始遵循的路径。 relativePath 的最后一个元素应总是有一个指定的目标名称。这进一步限制了 RelativePath 类型的定义。如果目标名称丢失,那么服务器应返回 Bad_BrowseNameInvalid。 Relativepath 结构在 7.25 中定义
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)
results []	BrowsePathResult	浏览路径列表的结果列表。列表的大小和顺序匹配 browsePaths 请求参数的大小和顺序
statusCode	StatusCode	浏览路径的 StatusCode(StatusCode 定义见 7.33)
targets []	BrowsePathTarget	从起始节点开始的 relativePath 的目标列表。 一个服务器可能会遇到对另一个服务器中节点的引用,当它处理 RelatviePath 时不能处理该节点。如果发生这种情况,那么服务器应返回外部节点的 NodeId,并且设置 remainingPathIndex 参数来指示其 RelativePath 仍需被处理。为了完成该操作,客户端应连接到另一台服务器,并使用目标作为 startingNode 且未处理的元素作为 relativePath,再次调用该服务
targetId	ExpandedNodeId	RelativePath 的目标的标识符
remainingPathIndex	Index	RelativePath 中的第一个未处理元素的索引。 如果所有元素都已处理,那么该值应等于 Index 数据类型的最大值(Index 定义见 7.12)
diagnosticInfos []	DiagnosticInfo	浏览路径的诊断信息列表(DiagnosticInfo 定义见 7.8)。列表的大小和顺序匹配 browsePaths 请求参数的大小和顺序。如果在请求首部中没有请求诊断信息,或者如果在请求处理中没有遇到诊断信息,那么该列表为空

5.8.4.3 服务结果

表 37 定义了该服务特定的服务结果。常用状态代码在 7.33 中定义。

表 37　TranslateBrowsePathsToNodeIds 服务结果代码

标识符	描述
Bad_NothingToDo	对于该结果代码的描述见表 165
Bad_TooManyOperations	对于该结果代码的描述见表 165

5.8.4.4　状态代码

表 38 定义了该服务特定的操作层面状态代码参数值。常用状态代码在表 166 中定义。

表 38　TranslateBrowsePathsToNodeIds 操作层面结果代码

标识 ID	描述
Bad_NodeIdInvalid	对于该结果代码的描述见表 166
Bad_NodeIdUnknown	对于该结果代码的描述见表 166
Bad_NothingToDo	对于该结果代码的描述见表 165。 此代码表示 relativePath 中包含一个空列表
Bad_BrowseNameInvalid	对于该结果代码的描述见表 166。 此代码表示 relativePath 中缺少目标名称
Uncertain_ReferenceOutOfServer	路径元素有在另一个服务器上的目标
Bad_TooManyMatches	请求的操作有太多匹配返回。用户应使用大型结果集的查询。在返回这个错误代码前，服务器应允许至少 10 个匹配条件
Bad_QueryTooComplex	请求的操作需要服务器上的太多资源
Bad_NoMatch	请求的操作没有匹配返回

5.8.5　RegisterNodes

5.8.5.1　描述

服务器经常无法直接访问其管理的信息。变量或服务可能处于底层系统，需要额外的努力来建立到这些系统的连接。客户端可以使用 RegisterNodes 服务来注册他们需要重复访问(如写、调用)的节点。它允许服务器做任何需要的设置，因此访问操作效率会更高。客户端在使用注册的 NodeId 时，可能期望性能改进，但是优化措施是供应商特定的。对于变量节点，服务器应尽力优化 Value 属性。

注册的 NodeId 只保证在当前会话中有效。客户应立即注销不需要的 Id，以释放资源。

RegisterNodes 不验证来自请求的 NodeId。服务器在响应中将简单地复制未知的 NodeId。结构上的 NodeId 错误(数据长度错误、无效 Id 类型)将导致整个服务失败。

出于审核目的，服务器不应使用注册的 NodeId，而只能使用规范的 NodeId，即 NodeId 属性的值。

5.8.5.2　参数

表 39 定义了该服务的参数。

表 39 RegisterNodes 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
nodesToRegister []	NodeId	被注册的 NodeId 列表,客户端通过浏览、查询或其他方式检索该列表
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)
registeredNodeIds []	NodeId	客户端用于后续访问操作的 NodeId 列表。列表的大小和顺序与 nodesToRegister 请求参数的大小和顺序相匹配。 服务器可能从该请求或一个新(别名)NodeId 返回 NodeId。建议服务器返回一个数字 NodeId 以防混淆。 在节点不支持优化的情况下,服务器应返回来自请求的 NodeId

5.8.5.3 服务结果

表 40 定义了该服务特定的服务结果。常用状态代码在表 165 中定义。

表 40 RegisterNodes 服务结果代码

标识符	描述
Bad_NothingToDo	对于该结果代码的描述见表 165
Bad_TooManyOperations	对于该结果代码的描述见表 165
Bad_NodeIdInvalid	对于该结果代码的描述见表 166。 如果 nodesToRegister 参数中的任一在结构上无效,那么服务器应完全拒绝 RegisterNodes 请求

5.8.6 UnregisterNodes

5.8.6.1 描述

该服务用于注销已通过 RegisterNodes 服务获得的 NodeId。

UnregisterNodes 不验证来自请求的 NodeId。服务器应简单地注销已注册的 NodeId。任何在列表中列出但没有注册的 NodeId,会被简单地忽略。

5.8.6.2 参数

表 41 定义了该服务的参数。

表 41　UnregisterNodes 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(RequestHeader 定义见 7.26)
nodesToUnregister []	NodeId	已通过 RegisterNodes 服务获得的 NodeId 列表
响应		
responseHeader	ResponseHeader	通用响应参数(ResponseHeader 定义见 7.27)

5.8.6.3　服务结果

表 42 定义该服务特定的服务结果。常用状态代码在表 165 中定义。

表 42　UnregisterNodes 服务结果代码

标识 ID	描述
Bad_NothingToDo	对于该结果代码的描述见表 165
Bad_TooManyOperations	对于该结果代码的描述见表 165

5.9　Query(查询)服务集

5.9.1　概述

此服务集合是用来发布对服务器的查询。OPC 统一架构查询具有通用性,它提供一种与底层存储机制无关的查询能力,可用于访问不同类型的 OPC UA 数据存储和信息管理系统。OPC 统一架构查询允许客户端在不了解数据内部存储逻辑结构的情况下访问由服务器维护的数据。地址空间信息对于数据访问是充分的。

期望 OPC 统一架构应用来使用 OPC 统一架构查询服务来作为初始化过程或者偶然性信息同步步骤的一部分。例如:OPC 统一架构查询用于访问长期存储的大量数据,来初始化一个带有当前系统配置状态的分析应用。查询也可对一个报告用于初始化或者填充数据。

查询定义了在地址空间中一个或多个 TypeDefinitionNodes 实例应提供的属性集合。由服务器返回的结果形式是 QueryDataSets 数组。在每个 QueryDataSets 中选择的属性值来自选择的 TypeDefinitionNodes 或相关的 TypeDefinitionNodes,在结果中出现的顺序与查询中属性的顺序相同。查询基于属性值以及 TypeDefinitionNodes 之间的关系,也支持节点过滤。

查询示例参见附录 B。

5.9.2　QueryingViews

视图是服务器中可用的地址空间的子集。地址空间中视图的组织描述见 IEC 62541-5。

对于任何现存的视图,查询可用来返回来自视图的数据的子集。当一个应用发布针对视图的查询时,只有视图中定义的数据会被返回。不包含在视图内但包含在原有地址空间的数据不被返回。

查询服务支持对当前与历史数据的访问。该服务支持客户端查询地址空间的以前版本。客户端在对使用地址空间以前版本的查询中,可能会指定一个 ViewVersion(视图版本)或者一个 Timestamp(时间戳)。OPC UA 查询是对历史访问(Historical Access)的补充。前者用作查询一个在某个时间已存

在的地址空间,后者被用于过期的属性值的查询。在这种方式下,查询也可被用来恢复部分以前的地址空间。这样即使节点不在当前的地址空间中,通过历史访问也可访问属性值历史。

支持查询的服务器期望能使用与本地服务器相关的地址空间以及在本地服务器上可用的任何视图。如果视图或地址空间也引用远程服务器,那么查询也可以访问远程服务器的地址空间,但不是必需的。如果某个服务器确实要访问远程服务器,那么该访问应使用在 5.5.1 中描述的客户端的用户标识。

5.9.3 QueryFirst

5.9.3.1 描述

此服务用于向服务器发出一个查询请求。查询的复杂度可以从非常简单到高度复杂。查询可以从 TypeDefinitionNode 或者受过滤器指定限制的 TypeDefinitionNode 的实例简单地请求数据。另一方面,查询可以通过从一个初始 TypeDefinitionNode 指定 RelativePath(相对路径)方法,从相关节点类型实例请求数据。在过滤器里,可构建一个单独的路径集合以限制提供数据的实例。一个过滤器的路径可包含多个 RelatedTo(相关性)运算符,以此来定义在源实例和目标实例间的多跳路径。例如,可以对参加特殊学校的学生进行过滤,但是会返回那些有关学生和他们家庭的信息。在这种情况下,遍历学生学校的关系用于过滤,但遍历学生家庭关系则用于选择数据。关于内容过滤(ContentFilter)的完整描述见 7.4,简单示例参见 B.1,更多复杂的内容过滤和查询示例参见 B.2。

客户端提供了 NodeTypeDescription 数组,NodeTypeDescription 规定 TypeDefinitionNode 的节点,并选择在响应中返回哪些属性。通过从一个初始 TypeDefinitionNode 开始的类型系统,客户端也可提供一个相对路径集合。使用这些路径,客户端从那些与初始 TypeDefinitionNode 实例相关的节点,选择一组属性集合。此外,客户端也可请求服务器返回 TypeDefinitionNode 的子类型的实例。如果被选择的属性不在类型定义节点中,却在子类型中。那么在 TypeDefinitionNode 包含一个无效(null)值,这将不会构成一个错误条件,并将无效(null)值返回给该属性。

通过将属性和特性限制到某些值,客户端可使用过滤参数来限制结果集合。另一方面,通过指定如何实现实例关联的方法,客户端使用过滤器来限定结果的方面,另一种方法是运用相关运算符。在这种情况下,如果一个在相对路径顶层的实例不能跟随着每个指定跳跃的路径底层,那么就没有查询数据集合会返回给起始实例或任何一个中间实例。

在相关路径上查询相关实例时,客户端可以要求得到引用。通过仅包含一个引用类型的相关路径请求一个引用。如果要求获得所有的引用,则根引用类型应被列出。这些引用被返回作为 QueryDataSets 部分。

5.9.3.2 参数

表 43 定义了 QueryFirst 服务的请求参数,表 44 定义了 QueryFirst 服务的响应参数。

表 43 QueryFirst 请求的参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用请求参数(见 7.26 的 RequestHeader 定义)
view	ViewDescription	规定视图和与服务器的临时上下文(见 7.37 的视图描述定义)
nodeTypes[]	NodeTypeDescription	节点类型描述
typeDefinitionNode	ExpandedNodeId	返回数据所属实例的起始 TypeDefinitionNode 的 NodeId

表 43（续）

名称	类型	描述
includeSubtypes	Boolean	一个标志，它阐述在节点类型实例的列表中，服务器是否应该包含 TypeDefinitionNode 子类型的实例
dataToReturn[ ]	QueryDataDescription	从起始 TypeDefinitionNode 和已给的返回数据所属的相对路径，规定属性或引用
relativePath	RelativePath	浏览与起始节点相关的路径。这些路径标识了包含被请求数据的节点。其中起始节点是一个由类型定义节点定义可类型的节点实例。这个实例节点由作为调用一部分的过滤器进行限制。相对路径的定义，参见 7.25。 这个相对路径在引用结尾，在这种情况下，这个引用的引用描述作为值的一部分返回
attributeId	IntegerId	属性的 Id。应是有效的 Id。IntegerId 的定义见 7.13。属性的 IntegerId 的定义见 IEC 62541 6。如果相对路径结束于一个引用，那么这个参数是"0"并被服务器忽略
indexRange	NumericRanger	这个参数用于识别一个结构或数组的单个元素。如果规定了元素的范围，那么这些值作为一个组成部分返回的。第一个元素由索引"0"标识。NumericRange 类型在 7.21 中定义。如果规定的属性不是一个数组或一个结构，那么这个参数是无效的(null)。如果指定的属性是一个数组或一个结构，H 参数是无效的(null)，那么所有的元素包含在该范围内
filter	ContentFilter	得到的节点应限定为与过滤器定义的标准相匹配。Content Filter 在 7.4 中描述。如果提供了一个空的过滤器，那么整个地址空间应该受到检查，包含与请求的属性或引用相匹配的节点会被返回
maxDataSetsToReturn	Counter	客户端要求服务器在响应以及在每次后续的连续调用响应返回的 QueryDataSets 的数量值。服务器允许进一步限制响应，但不应超过该限值。 "0"值指示客户端是强加的不受限制
maxReferencesToReturn	Counter	客户端要求服务器在每个 QueryDataSets 的响应以及在每次后续的连续调用响应返回的引用个数。服务器允许进一步限制响应，但不应超过该限值。 "0"值指示客户端是强加的不受限制。 例如，结果返回 4 个节点，但是每个节点有 100 个引用，如果将该限值设置为 50，那么只有每个节点上的前 50 个引用将会在初始化调用是返回，并设置连续点以指示附加数据

表 44　QueryFirst 响应参数

名称	类型	描述
响应(response)		
response Header	ResponseHeader	通用响应参数(见 7.27 的响应首部定义)
queryDataSets[]	QueryDataSet	QueryDataSets 数组。如果没有节点或引用符合节点类型标准,那么数组为空。在这种情况下,continuationPoint 参数应为空。 QueryDataSet 类型定义见 7.22
contenuationPoint	ContinuationPoint	服务器定义的用于识别连续点的不透明值。 连续点只有在查询结果太庞大以至于在单个响应中不能返回时才使用。在上下文中的太庞大意味着这样一个事实:这个服务器不可能返回一个庞大的响应,或者返回的 QueryDataSets 超过了客户端请求中规定的返回 QueryDataSets 的最大个数。 连续点在 QueryNext 服务中使用。当不用的时候,这个参数值为空(null)。如果一个连续点被返回了,那么这个客户端应该调用 QueryNext 得到下个 QueryDataSets,或者来释放服务器中的连续点的资源。 直到客户端将连续点传给 QueryNext,或会话已关闭,连续点应该仍然是有效的。如果已达到最大连续点,那么最原始的连续点应被重置。 连续点类型在 7.6 中有描述
parsingResults[]	ParsingResult	QueryFirst 的解析结果列表。列表的大小和顺序与节点类型请求参数的大小和顺序向匹配。 列表中是由一些状态码构成的,这些状态码是与那些节点类型的处理相关的,且这个节点类型是查询的一部分。如果无错误,那么数组为空。如果任何一种节点类型遇到错误,那么所有的节点类型应该有一个相关的状态码
statusCodes	StatusCode	节点类型描述请求的解析结果
dataStatusCodes[]	StatusCode	dataToReturn 的结果列表。列表的大小和顺序与 dataToReturn 请求参数的数据的大小和顺序向匹配。如果没有错误,那么这个数组可能是空的
dataDiagnosticInfos[]	DiagnosticInfo	dataToReturn 的诊断信息数据列表(见 7.8 中的诊断信息定义)。列表的大小和顺序与 dataToReturn 请求参数的数据大小和顺序匹配。如果在请求头中没有请求诊断信息,或如果在查询请求的过程中无诊断信息,那么列表为空
diagnosticInfos	DiagnosticInfo	请求的 NodeTypeDescription 的诊断信息列表。如果在请求头中没有请求诊断信息,或如果在查询请求的过程中无诊断信息,那么列表为空
filterResult	ContentFilterResult	包含了与过滤器相关的任何错误的结构。 如果无错误发生,那么该结构为空。 内容过滤器结果类型在 7.4.2 中有定义

5.9.3.3 服务结果

如果查询是无效的或者不能被处理，那么查询数据集不会被返回，且只有一个服务结果，过滤结果，解析结果以及可选的诊断信息会被返回。表 45 定义了这个服务特定的服务结果。通常状态码在表 165 中定义。

表 45 QueryFirst 服务结果规范

标识符(symbolicId)	描述
Bad_NothingToDo	见表 165 的结果码描述
Bad_TooManyOperations	见表 165 的结果码描述
Bad_ContentFilterInvalid	见表 165 的结果码描述
Bad_ViewIdUnknown	见表 165 的结果码描述
Bad_ViewTimestampInvalid	见表 165 的结果码描述
Bad_ViewParameterMismatchInvalid	见表 165 的结果码描述
Bad ViewVersionInvalid	见表 165 的结果码描述
Bad_InvalidFilter	提供的过滤器无效，见特定错误的过滤器结果
Bad_NodelistError	提供的节点类型包含一个错误，见特定错误的解析结果
Bad_InvalidView	提供的视图描述不是一个有效的视图描述
Good_ResultsMayBeIncomplete	服务器本应该随着引用跳到一个远程服务器的节点，但未实现。这个结果集可能是不完整的

5.9.3.4 状态码

表 46 定义了这个服务特定的解析结果状态码参数的值。在表 166 中定义了通用的状态码。

表 46 QueryFirst 操作级结果码

标识符(Symbolic Id)	描述
Bad_NodeIdInvalid	见表 166 的结果码的描述
Bad_NodeIdUnknown	见表 166 的结果码的描述
Bad_NotTypeDefinition	提供的节点不是一个类型定义节点 Id
Bad_AttributeIdInvalid	见表 166 的结果码的描述
Bad IndexRangeInvalid	见表 166 的结果码的描述

5.9.4 QueryNext

5.9.4.1 描述

此项服务是用来请求下一个 QueryFirst 或 QueryNext 响应信息的集合，这种信息太庞大以至于不能在一个单独的响应中发到。此上下文中的"太庞大"意味着，该服务器不能返回一个庞大的响应，或者要返回查询数据集合数量超过了原有请求中客户端规定的返回查询数据集合的最大数。QueryNext 应该在用来递交正要进行中的 QueryFirst 或者 QueryNext 的相同会话中被递交。

5.9.4.2 参数

表 47 定义了该服务的参数。

表 47 QueryNext 服务参数

名称	类型	描述
请求(Request)		
请求首部(requestHeader)	Request Header	通用的请求参数(见 7.26 中的请求首部定义)
releaseContinuationPoint	Boolean	有以下值的布尔型参数: TRUE 传递的连续点应重置来释放服务器中的连续点资源。 FALSE 传递的连续点应用于得到 QueryDataSet 的下个集合。 客户端应该要总是使用 QueryFirst 或 QueryNext 返回的连续点,以释放服务器中连续点的资源。如果客户端不想得到查询信息的下个集合,那么该参数设为 TRUE 来调用 QueryNext。 如果参数设置成 TRUE,那么响应中的所有数组参数应包含空数组
continuationPoint	ContinuationPoint	服务器定义了表示连续点的不透明值。连续点值会在一个之前的 QueryFirst 或 QueryNext 响应中返回到客户端。该值用来标识这个先前已处理的正在进行中的 QueryFirst 或 QueryNext,以及正在进行中的浏览响应的结果集中的点。 在 7.6 中描述了连续点类型
响应(Response)		
响应首部(responseHeader)	Response Header	通用响应参数(见 7.27 中的响应首部定义)
queryDataSets []	QueryDataSet	QueryDataSet 数组,在 7.22 中定义了 QueryDataSet 类型
revisedContinuationPoint	ContinuationPoint	服务器定义的代表连续点的不透明值。只有在要返回的信息太庞大以至于不能包含在一个单独的响应的情况下使用。当它不使用时,或当设置 releaseContinuationPoint 时,该参数值是无效的(null)

5.9.4.3 服务结果

表 48 定义了这类服务指定的服务。公用的状态码在表 165 中已定义。

表 48 查询下一个服务结果码

标识符	描述
Bad_ContinuationPointInvalid	见表 166 中本结果码的描述

5.10 Attribute(属性)服务集

5.10.1 概述

该服务集提供了用于访问节点的属性。

5.10.2 Read

5.10.2.1 描述

该服务用来读一个或多个节点的一个或多个属性。对于其元素被索引的结构式属性值,比如一个数组,此服务允许客户端将整个索引值的集合作为一个组成部分来读取,允许客户端读个别元素或读组成部分的元素范围。

如果数据的备份比参数 maxAge 规定的还要年长,参数 maxAge 用于引导服务器从底层数据源获取值,比如一个设备。如果服务器不符合请求的最大时间,那么它将返回它的"最理想"值,而非拒绝请求。

5.10.2.2 参数

表 49 定义了服务参数。

表 49 读服务参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用的请求参数(见 7.26 请求首部定义)
maxAge	Duration	以毫秒表示的读取值的最大老化时间。值的老化时间是基于服务器的时间戳和服务器开始处理请求的时间点的差值。例如:如果客户端指定了 maxAge 为 500 ms,且直到服务器开始处理这个请求时已过去了 100 ms,那么返回值的老化时间可能是在它被请求时的时间点之前的 600 ms。 如果服务器有一个或多个属性值,且这个属性是在最大老化时间内,那么它可能返回值中的任何一个或者它从数据源中可读到一个新值。服务器的属性值个数取决于定义该属性的 MonitoredItems 个数。在任何情形下,客户端无需假定应返回哪种版本的数据。 如果服务器没有在最大老化时间内的值,那么它应该尝试去读来自数据源的新值。 如果服务器不能符合请求的 maxAge,它返回最理想值,而非拒绝请求。当服务器对数据进行访问后,对数据进行处理并返回新数据值所需的时间大于规定的最大老化时间时,会发生上述情况。 如果 maxAge 设置成 0,那么服务器应该尝试去读一个来自数据源的新值。 如果 maxAge 被设置成最大的 32 位整型值或更大,那么服务器应该尝试去获取一个缓存值。 负数值对 maxAge 来说是无效的

表 49（续）

名称	类型	描述
timestampsToReturn	Enum TimestampsToReturn	为每个请求的变量值属性规定了要返回的时间戳的枚举值。TimestampsToReturn 枚举型在 7.34 中定义
nodesToRead []	ReadValueId	要读取的节点和其属性列表。对于列表里的每一个条目，便会返回一个状态码。如果指示成功，那么属性值也会被返回。7.23 中定义了读值参数类型
响应(Response)		
responseHeader	ResponseHeader	通用响应参数(见 7.27 的 ResponseHeader 定义)
results []	DataValue	属性值列表(见 7.7 中的数据值定义)。列表的大小和顺序与要读的节点请求参数的大小和顺序相匹配。对于每一个包含在 nodesToRead 参数中的节点，列表中都有一个条目
diagnosticInfos []	DiagnosticInfo	诊断信息列表(见 7.8 中的诊断信息定义)。列表的大小和顺序与要读的节点请求参数的大小和顺序相匹配。对于每一个包含在 nodesToRead 参数中的节点，列表中都有一个条目。如果诊断信息在请求首部没被请求，或者如果在处理请求过程中无诊断信息，那么该列表为空

5.10.2.3 服务结果

表 50 定义了该服务特定的服务结果。通用状态码在表 165 中有定义。

表 50 Read 服务结果码

助记符	描述
Bad_NothingToDo	见表 165 中结果码描述
Bad_TooManyOperations	见表 165 中结果码描述
Bad_MaxAgeInvalid	maxAge 参数无效
Bad_TimestampsToReturnInvalid	见表 165 中结果码描述

5.10.2.4 状态码

表 51 定义了包含在每一个值元素的数据值结构中的操作等级状态码的值。通用状态码在表 166 中定义。

表 51　读操作等级结果码

符号 Id	描述
Bad_NodeIdInvalid	见表 166 中结果码的描述
Bad_NodeIdUnknown	见表 166 中结果码的描述
Bad_AttributeIdInvalid	见表 166 中结果码的描述
Bad_IndexRangeInvalid	见表 166 中结果码的描述
Bad_IndexRangeNoData	见表 166 中结果码的描述
Bad_DataEncodingInvalid	见表 166 中结果码的描述
Bad_DataEncodingUnsupported	见表 166 中结果码的描述
Bad_NotReadable	见表 166 中结果码的描述
Bad_UserAccessDenied	见表 166 中结果码的描述

5.10.3　HistoryRead

5.10.3.1　描述

该服务用来读一个或多个节点的历史值或历史事件。对于其元素被索引的结构化属性值，如一个数组，该服务允许客户端将整个有索引值的集合作为组成部分来读取，还允许客户端读个别元素或者读作为组成部分的元素范围。虽然历史值在地址空间中不可见，但是服务器可以使用该服务使历史值对于客户端可用。

在 IEC 62541-3 定义的 AccessLevel 属性指示节点对历史值的支持。若干个请求参数指示服务器怎样从底层历史数据源中访问历史值。在 IEC 62541-3 定义的 EventNotifier 属性为历史事件指示节点对历史事件的支持。

如果不是所有值都可在一个响应中返回，则 HistoryRead 中的 continuationPoint 参数被用来标记继续读取的点。对于客户端该值是不透明的，仅用于维护状态信息使得服务器从该点继续读取。如果时间戳是惟一的，那么服务器可以使用最后返回数据项的时间戳。这可减少在服务器中存储连续点状态信息的需求。

关于读历史数据和历史事件的更详细信息见 IEC 62541-11。

5.10.3.2　参数

表 52 定义了服务参数。

表 52　HistoryRead 服务参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用请求参数(见 7.26 节的 RequestHeader 定义)

表 52（续）

名称	类型	描述
historyReadDetails	Extensible Parameter HistoryReadDetails	详细定义了可被执行的读取历史数据的类型。historyReadDetails 参数类型是在 IEC 62541-11 形式上定义的。一个扩展参数类型。在 7.11 中定义了扩展参数类型
timestampsToReturn	enum TimestampsToReturn	枚举类型为要返回的时间戳指定了变量值属性。要返回的时间戳枚举型在 7.34 中有定义。 规定"NEITHER"的 timestampsToReturn 是无效的。在这种情况下，服务器应该返回一个 Bad_InvalidTimestampArgument 状态码
releaseContinuationPoints	Boolean	有以下值的布尔型参数： TURE 已传递的连续点应该重置以释放服务器中资源。 FALSE 已传递的连续点应该用于得到历史信息的下个集合。 客户端应该总是使用 HistoryRead 响应返回的连续点，以释放服务器中连续点占用的资源。如果客户端不想得到下一个历史信息集合，那么调用 HistoryRead 应将该参数设置为 TRUE
nodesToRead []	HistoryReadValueId	这个参数包含着项的列表，通过该列表可执行历史恢复
nodeId	NodeId	如果 HistoryReadDetails 是 RAW、PROCESSED、MODIFIED 或 ATTIME，那么该参数是其历史值将被读取的节点的 nodeId，返回的值应该总是包含时间戳。 如果 HistoryReadDetails 是 EVENTS，那么该参数是其历史事件将被读取的节点的 nodeId。 如果节点不支持对历史值或者历史事件的请求访问，那么应产生对所给节点的适当错误响应
indexRange	NumericRange	该参数用来标识一个数组的一个元素，或许多数组一个索引范围。如果规定了元素的范围，那么该值作为一个组成部分部分被返回。第一个元素会被索引(0)确定。NumericRange 类型在 7.21 中定义。 如果值不是一个数组，那么该参数无效。然而，如果值是数组，且参数为空，那么所有元素都包含在该范围内
dataEncoding	QualifiedName	规定了要返回的读取值的数据编码的 QualifiedName。(7.23 定义了如何规定数据编码) 当读事件历史时，忽略该参数

表 52（续）

名称	类型	描述
continuationPoint	ByteString	对于每一个要读节点，该参数指定了由先前 HistoryRead 调用中返回的连续点，它允许客户端从最后一个收到的值继续读取。 HistoryRead 用来选择一个有序的历史值的或事件序列。一个连续点标记了有序序列中的一个点，这样服务器可返回在该标记点之后的序列子集。 值为空指示该参数没有被使用。 关于连续点的更多细节见 IEC 62541 11
响应(Response)		
responseHeader	ResponseHeader	通用响应参数(见 7.27 中的 ResponseHeader)
results []	IHistoryReaderResult	读结果列表。列表的的大小和顺序与要读取节点请求参数的大小和顺序相匹配
statusCode	StatusCode	要读节点的状态码(见 7.33 的状态码定义)
continuationPoint	ByteString	仅当返回值的数目过大以至于不能在一个单独的响应中被返回的情况下，使用该参数。 当不使用该参数时，其值为空(null)。 对于每个会话，服务器应至少支持一个连续点。服务器规定了对于每个会话在 IEC 62541-5 定义的服务器能力对象能支持的最大历史连续点。在客户端将连续点传递到 IHistoryRead 中或会话已关闭前，连续点应该保持活动。如果已经达到最大连续点，那么最老的连续点应该复位
historyData	ExtensibleParameter IHistoryData	针对节点返回的历史数据。 历史数据参数类型是在 IEC 62541-11 中定义的一个扩展的参数类型。它指定了历史数据类型，这个历史数据时是能返回的
diagnosticInfos	DiagnosticInfo	诊断信息列表。列表的大小和顺序与要读节点请求参数的大小和顺序相匹配。列表中针对每一个在要读按参数中的节点都有一个入口。如果诊断信息没有在请求首部中请求或如果在请求的处理过程中遇到无诊断信息，那么这个列表便是空的

5.10.3.3 服务结果

表 53 定义了此项服务特定的服务结果。公用状态码在表 165 中有定义。

表 53 历史读服务结果码

助记符	描述
Bad_NothingToDo	见表 165 的结果码描述
Bad_TooManyOperations	见表 165 的结果码描述
Bad_TimestampsToReturnInvalid	见表 165 的结果码描述
Bad_HistoryOperationInvalid	见表 165 的结果码描述
Bad_HistoryOperationUnsupported	见表 165 的结果码描述

5.10.3.4 状态码

表 54 定义了运算水平状态码参数的值,这个参数是此项服务特定的。公用的状态码在表 166 中有定义。历史通道中特定的状态码在 IEC 62541-11 中有定义。

表 54 HistoryRead 操作等级结果码

符号 Id	描述
Bad_NodeIdInvalid	见表 166 中的结果码描述
Bad_NodeIdUnknown	见表 166 中的结果码描述
Bad_DataEncodingInvalid	见表 166 中的结果码描述
Bad_DataEncodingUnsupported	见表 166 中的结果码描述
Bad_UserAccessDenied	见表 166 中的结果码描述
Bad_ContinuationPointInvalid	见表 166 中的结果码描述
Bad_InvalidTimestampArgument	见表 166 中的结果码描述

5.10.4 Write

5.10.4.1 描述

该服务用于将值写入到一个或多个节点的一个或多个属性。对于自身元素被索引的结构化属性值,如:数组,该服务允许客户端将整个有索引值的集合作为其组成部分来写,还允许客户端写入组成部分的单个元素或者元素范围。

该值被写入到数据源,如:设备,且直到它写完值或认为该值是不可写之前,该服务不会返回。在某种情况下,服务器将会成功的写入到中间系统或服务器,且不知道数据源是否正确地更新。在这些情况下,服务器应该报告一个成功码,指示写操作未被验证。在服务器能验证已成功写入到数据源的情况下,它会报告一个无条件成功。

服务器可能会成功写一些属性,但不是所有的属性。回退(Rollback)是客户端的职责。

如果服务器允许对数据类型 LocalizedText 的属性进行写操作,客户端可通过写入带有相关 LocaleId 的文本,增加或重写现场文本。对现场写一个空的字符串应该删除该现场字符串。对文本写一个空的字符串与对 LocaleId 写一个无效的字符串应该删除所有现场入口。

5.10.4.2 参数

表 55 定义了该服务的参数。

表 55 写服务参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用请求参数(见 7.26 中的定义)
nodesToWrite	WriteValue	节点列表和它们要写的属性
nodeId	NodeId	包含属性的节点的 NodeId
attributeId	IntegerId	属性 Id。这应该是一个有效的属性 Id。IntegerId 定义见 7.13。IntegerId 的属性在 IEC 62541-6 中有定义

表 55（续）

名称	类型	描述
indexRange	NumericRange	这个参数是用来鉴定一个数组的一个单独的元素，或数组索引的一个单独的范围。第一个元素是由索引 0 定义的。NumericRange 的类型在 7.21 中定义
value	DataValue	节点属性值（见 7.7 中的数据值定义）。 如果索引范围参数是特定的，接着那么即使只有一个元素正被写入，值也应该是一个数组。 如果源时间戳或服务器时间戳是特定的，那么服务器应该使用这些值。如果服务器不支持时间戳写操作，那么它会返回 Bad_WriteNotSupported error 信息。 如果值的数据类型不一样或属性的数据类型的子类型不一样，那么服务器应该返回一个 Bad_TypeMismatch error。如果服务器不支持已传递数据编码的写操作，那么返回 Bad_DataEncodingUnsupported error
响应（Response）		
responseHeader	ResponseHeader	通用响应参数（见 7.27 中的响应头定义）
results[ ]	StatusCode	要写节点的结果列表（见 7.33 中的状态码定义）。列表的大小和顺序与要写节点请求参数的大小和顺序相匹配。对于包含在要写节点参数中的每一个节点在列表中都有一个入口
diagnosticInfos [ ]	DiagnosticInfo	要写节点的诊断信息列表（见 7.8 中的诊断信息定义）。列表的大小和顺序与要写节点请求参数的大小和顺序相匹配。如果诊断信息在请求首部中未被请求或在请求处理过程中无诊断信息遇到，那么这个列表是空的

5.10.4.3 服务结果

表 56 定义了此项服务特定的服务结果。通用状态码在表 165 中有定义。

表 56 写服务结果码

符号 Id	描述
Bad_NothingToDoHistoryRead	见表 165 的结果码描述
Bad_TooManyOperations	见表 165 的结果码描述

5.10.4.4 状态码

表 57 定义此项服务的特定结果参数的值。通用状态码在表 166 中有定义。

表 57 写操作级结果码

符号 Id	描述
Good_CompletesAsynchronously	见表 165 的结果码描述。 值成功地写入一个中间系统，但服务器不知道数据源是否已更新

表 57（续）

符号 Id	描述
Bad_NodeIdInvalid	见表 165 的结果码描述
Bad_NodeIdUnknown	见表 165 的结果码描述
Bad_AttributeIdInvalid	见表 165 的结果码描述
Bad_IndexRangeInvalid	见表 165 的结果码描述
Bad_IndexRangeNoData	见表 165 的结果码描述
Bad_WriteNotSupported	请求的写运算不支持。如果客户端尝试写任何值、属性(quality)和时间戳的组合，且服务器不支持请求的组合(这个组合可是一个单独址，比如时间戳)，则服务器不应该在这个节点上执行任何写操作，且应该返回这个节点的状态码
Bad_NotWritable	见表 165 的结果码描述
Bad_UserAccessDenied	见表 165 的结果码描述。当前用户无写属性操作的许可
Bad_OutOfRange	见表 165 的结果码描述
Bad_TypeMismatch	见表 165 的结果码描述
Bad_DataEncodingUnsupported	见表 165 的结果码描述

5.10.5 HistoryUpdate

5.10.5.1 描述

此项服务用来更新一个或多个节点的历史值或事件。若干个请求参数阐述了服务器是怎样更新值或事件的。有效的操作是插入、替代或删除。

5.10.5.2 参数

表 58 定义了该服务的参数。

表 58 HistoryUpdate 服务参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用的请求参数(见 7.26 中的定义)
historyUpdateDetails []	ExtensibleParameter HistoryUpdateDetails	为此更新定义的详情。HistoryUpdateDetails 参数类型是在 IEC 62541-11 中格式化定义的一个扩展参数类型。它规定了那些能执行的历史更新类型。ExtensibleParameter 类型在 7.11 中定义
响应(Response)		
responseHeader	ResponseHeader	通用的响应参数(见 7.27 中的定义)
results []	HistoryUpdateResult	历史更新详情的更新结果列表。列表的大小和顺序与在请求中规定的 HistoryUpdateDetails 参数的详情里的元素的大小和顺序相匹配

表 58（续）

名称	类型	描述
statusCode	StatusCode	节点更新的状态码(见 7.33 中的状态码定义)
operationResults []	StatusCode	在一个节点上执行操作的诊断信息列表(见 7.8 中的 DiagnosticInfo 定义)。列表的大小和顺序与任何由详情元素定义的大小和顺序相匹配。详情元素由该 UpdateResults 输入项来报告
diagnosticInfos []	DiagnosticInfo	在节点上执行的与操作对应的状态码列表(见 7.8 中的定义)。列表的大小和顺序是与任一个详情元素定义的列表的大小和顺序相匹配,该详情元素由这个 UpdateResults 输入项来报告。如果诊断信息未在请求首部中请求或者在请求处理过程中无诊断信息,那么该列表为空
diagnosticInfos []	DiagnosticInfo	历史更新详情对应的诊断信息的列表。列表的大小和顺序是与请求中规定的 HistoryUpdateDetails 参数的详情元素的大小和顺序相匹配。如果诊断信息未在请求头中请求或者在请求处理过程中无诊断信息,那么该列表为空

5.10.5.3 服务结果

表 59 定义了此服务特定的服务结果。公用的状态码在表 165 中有定义。

表 59 历史更新服务结果码

标识符	描述
Bad_NothingToDo	见表 165 中此结果的描述
Bad_TooManyOperations	见表 165 中此结果的描述

5.10.5.4 状态码

表 60 定义了服务特定的 StatusCode 参数和 OperationResult 参数的值。通用的状态码在表 166 中有定义。历史访问特定的状态码在 IEC 62541-11 中有定义。

表 60 HistoryUpdate 操作级结果码

标识符	描述
Bad_NotWritable	见表 166 中此结果的描述
Bad_HistoryOperationInvalid	见表 166 中此结果的描述
Bad_HistoryOperationUnsupported	见表 166 中此结果的描述

5.11 Method(方法)服务集

5.11.1 概述

Method(方法)标识对象的函数调用,在 IEC 62541 3 中有定义。Method 被调用并且只在完成(成

功或不成功)之后才返回。方法的执行时间可能会变化,这取决于执行的函数。

Method 服务集合定义了调用方法的手段。一个方法应该是一个对象中的一个部件。发现是通过 Browse 和 Query 服务来提供的。通过浏览用于识别它们所支持的方法的对象引用,客户端发现服务器支持的方法。

因为方法可能会控制设备操作的一些方面,因此方法调用可能取决于环境或其他条件。在方法已经完成执行后试图立即重新调用该方法时,尤其如此,因为调用方法的条件可能还没有返回到允许重新开始该方法的状态。

5.11.2 Call

5.11.2.1 描述

此服务用于调用方法的一个列表。每个方法调用是在现有会话的上下文中被调用。如果该会话终止,那么方法的执行结果不可能会被返回给客户端,且会被丢弃。方法的执行与服务器上实际执行的任务是不相关的。

此服务用于向方法传递输入/输出变量。这些变量由方法的特性定义。

5.11.2.2 参数

表 61 定义了服务对应的参数。

表 61 调用服务参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用的请求参数(见 7.26 中的定义)
methodsToCall []	CallMethodRequest	要调用的方法列表
objected	NodeId	NodeId 应该是对象或者对象类型的 NodeId,对象或对象类型是 HasComponent 引用的源(或 HasComponent 引用的子类型)。 见 IEC 62541 3 的对象描述和它们的方法
methodId	NodeId	用来调用方法的节点 Id
inputArguments	BaseDataType	输入自变量值的列表。一个空列表表明了无输入自变量。这个列表的大小和顺序与由方法的输入 inputArguments 属性定义的输入自变量的大小和顺序相匹配
响应(Response)		
responseHeader	ResponseHeader	通用的响应参数(见 7.27 的定义)
result[]	CallMethodResult	方法调用的结果
statusCode	StatusCode	在服务器里执行的方法的状态码。如果至少一个输入自变量不符合约束条件(如错误数据类型,范围外的值),那么这个状态码会被设置成 Bad_InvalidArgument StatusCode。 如果方法在服务器里执行失败,如基于一个异常或者一个 HRESULT,那么状态码会设置成 bad StatusCode
inputArgumentResults	StatusCode	每个 InputArgument 的状态码列表

表 61（续）

名称	类型	描述
inputArgument DiagnosticInfos	DiagnosticInfo	每个 InputArgument 的诊断信息列表。如果诊断信息未在请求首部中被请求或者如果在请求的处理过程中无诊断信息，那么该列表为空
outputArguments	BaseDataType	输出自变量值列表。一个空列表指示没有输出自变量。该列表的大小和顺序与由方法的 OutputArguments 属性定义的输出自变量的大小和顺序相匹配。 名称、描述和每个自变量的数据类型是由方法的 OutputArguments 属性的每个元素中的自变量结构定义
diagnosticInfos []	DiagnosticInfo	callResult 的状态码的诊断信息列表。如果诊断信息未在请求首部中被请求或者如果在请求的处理过程中无诊断信息，那么该列表为空

5.11.2.3 服务结果

表 62 定义了此服务特定的服务结果。通用的状态码在表 165 中的定义。

表 62 Call 服务结果码

标识符	描述
Bad_NothingToDo	见表 165 中此结果码的描述
Bad_TooManyOperations	见表 165 中此结果码的描述

5.11.2.4 状态码

表 63 定义了服务特定的 StatusCode 和 OperationResults 参数的值。通用状态码在表 166 中定义。

表 63 调用运算水平结果码

标识符	描述
Bad_NodeIdInvalid	见表 166 中此结果码的描述。用于指示特定对象是无效的
Bad_NodeIdUnknown	见表 166 中此结果码的描述。用于指示特定对象是无效的
Bad_ArgumentsMissing	客户端不能规定方法对应的所有的输入自变量
Bad_UserAccessDenied	见表 165 中此结果码的描述
Bad_MethodInvalid	方法 Id 与特定对象对应的方法无关
Bad_OutOfRange	见表 166 中此结果码的描述。 用于指示输入自变量在可接受范围之外
Bad_TypeMismatch	见表 166 中的此结果码的描述。 用于指示输入自变量无正确的数据类型

5.12 MonitoredItem(监视项)服务集

5.12.1 MonitoredItem 模型

5.12.1.1 概述

客户端定义 MonitorItem(监视项)来订阅数据和事件。每个 MonitorItem 识别受监视的项,以及用来发送通知的订阅。受到监视的项可能是任一个节点属性。

通知是描述数据变化和事件的数据结构。它们被打包到通知报文内以传送给客户端。订阅在一个特定用户发布间隔上定期地发送通知报文,发送这些报文的周期被称作发布周期。

为 MonitorItem 定义了四个主要参数,这些参数告诉服务器如何采样、评估与报告项目。这些参数是采样间隔、监视模型、过滤器以及队列参数。图 15 说明了这些概念。

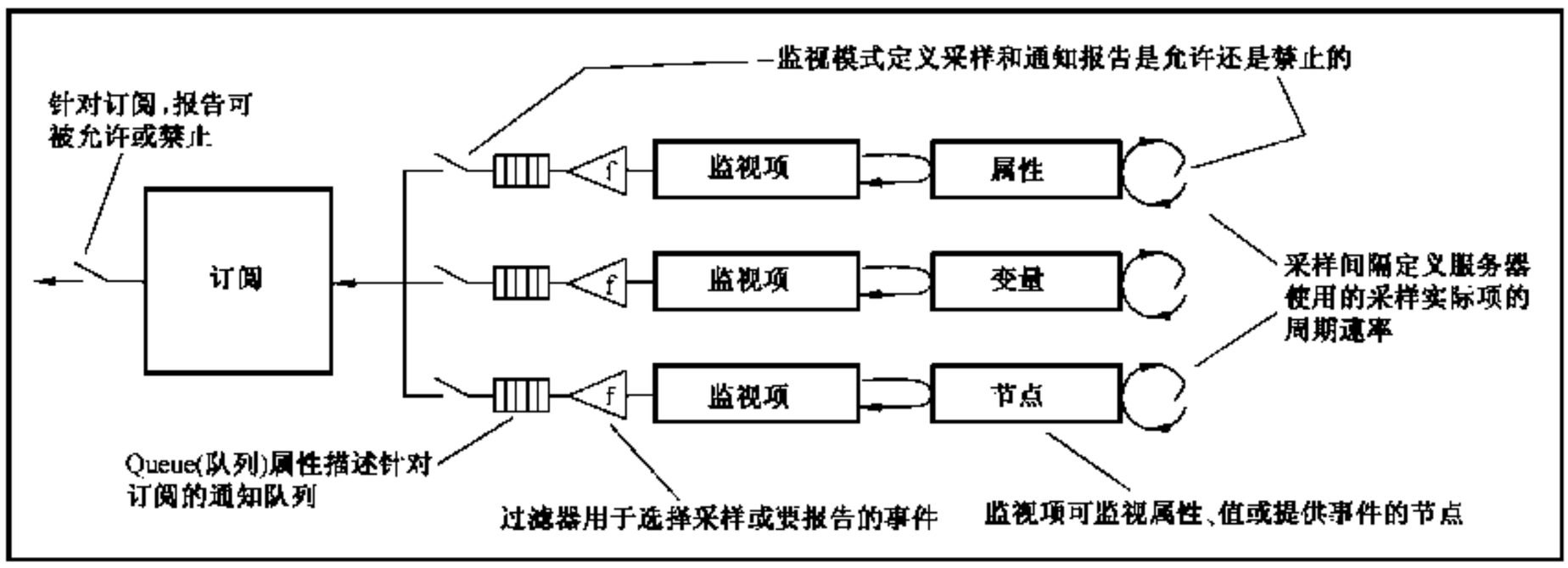

图 15 监视项模型

除了值属性,只对其他属性值的变化进行监视。过滤器不用于这些属性的。这些属性的任何值的变化会产生一个通知。

当监视变量时,值属性会被用到。对变量值的变化或变量状态改变进行监视。在本部分(见 7.16.2 节)和 IEC 62541-8 中定义的过滤器是用来确定:值变化是否足够大可产生针对变量的通知。

对象和视图可用来监视事件。只有来自那些 EventNotifier 属性的 SubscribeToEvents 比特被设置的节点的事件才是可用的。在本部分(见 7.16.3)中定义的过滤器用于确定:从节点收到的事件是否要发送给客户端。过滤器也允许选择 EventType 的字段,这些字段也将被包含在事件中,如同 EventId、EventType、SourceNode、Time 以及 Description。

IEC 62541-3 描述了事件模型和基本事件类型。

在地址空间中,基本事件类型的属性和基本事件类型的表示法在 IEC 62541-5 中规定。

5.12.1.2 采样间隔

为客户端创建的每个监视项(MonitoredItem)分配一个采样间隔,采样间隔或者是继承自订阅的发布间隔,或者是特殊定义来重载该速率。采样间隔表示服务器为数据交换对其底层数据源进行采样的最快速率。

如果客户端对事件进行订阅,那么它应定义一个 0 的采样间隔。

分配的采样间隔定义了一个"最理想的"循环速率,服务器按照该间隔采样来自它数据源的项。上下文中的"最理想"是指:服务器尽可能以这个速率采样。在比这个速率更快的采样是可接受的,但从满足客户端要求来看不是必须的。服务器如何处理采样速率以及隔多长时间对它的数据源进行内部轮询是服务器实现细节。但是,返回给客户端的值之间的时间间隔应大于或等于该采样间隔。

客户端也可指定采样间隔为0，这表示服务器应使用最快的实际速率。期望服务器仅支持有限的采样间隔集合来优化操作。如果服务器不支持客户端请求的间隔，那么服务器将其确定的最合适间隔分配给该监视项，并将该分配的间隔返回给客户端。在IEC 62541-5中定义的服务器能力对象识别服务器支持的采样间隔。

服务器可支持基于采样模型收集的或基于一种异常模型产生的数据。采样间隔所支持的最快速度可等于0，这表明数据项是基于异常而不是按周期进行采样。基于异常意味着底层系统不须采样以及报告数据的变化。

客户端可使用修改的采样间隔值作为对设置公共间隔的提示并保持订阅计数有效。例如：最小修改的监视项采样间隔是5s，那么在保持有效被发送之前的时间应该比5s长。

注意：在很多情况下，OPC UA服务器提供了对解耦系统的访问，因此并不清楚数据更新逻辑。在这种情况下，即使OPC UA服务器在协商的速率下采样，底层系统可能会以低得多的速率的更新数据。在这种情况下，变化只可能在这种低速率下检测。

如果底层系统更新项的行为是已知的，通过IEC 62541 3中定义的MinimumSamplingInterval属性是可用的。如果服务器给MinimumSamplingInterval属性指定一个值，且客户端订阅了Value属性，客户端应总是返回revisedSamplingInterval，该属性等于或高于MinimumSamplingInterval。

客户端也应意识到：OPCUA服务器采样和基础系统更新周期通常是不同步的。这可能引起在变化检测中额外的延时，见图16。

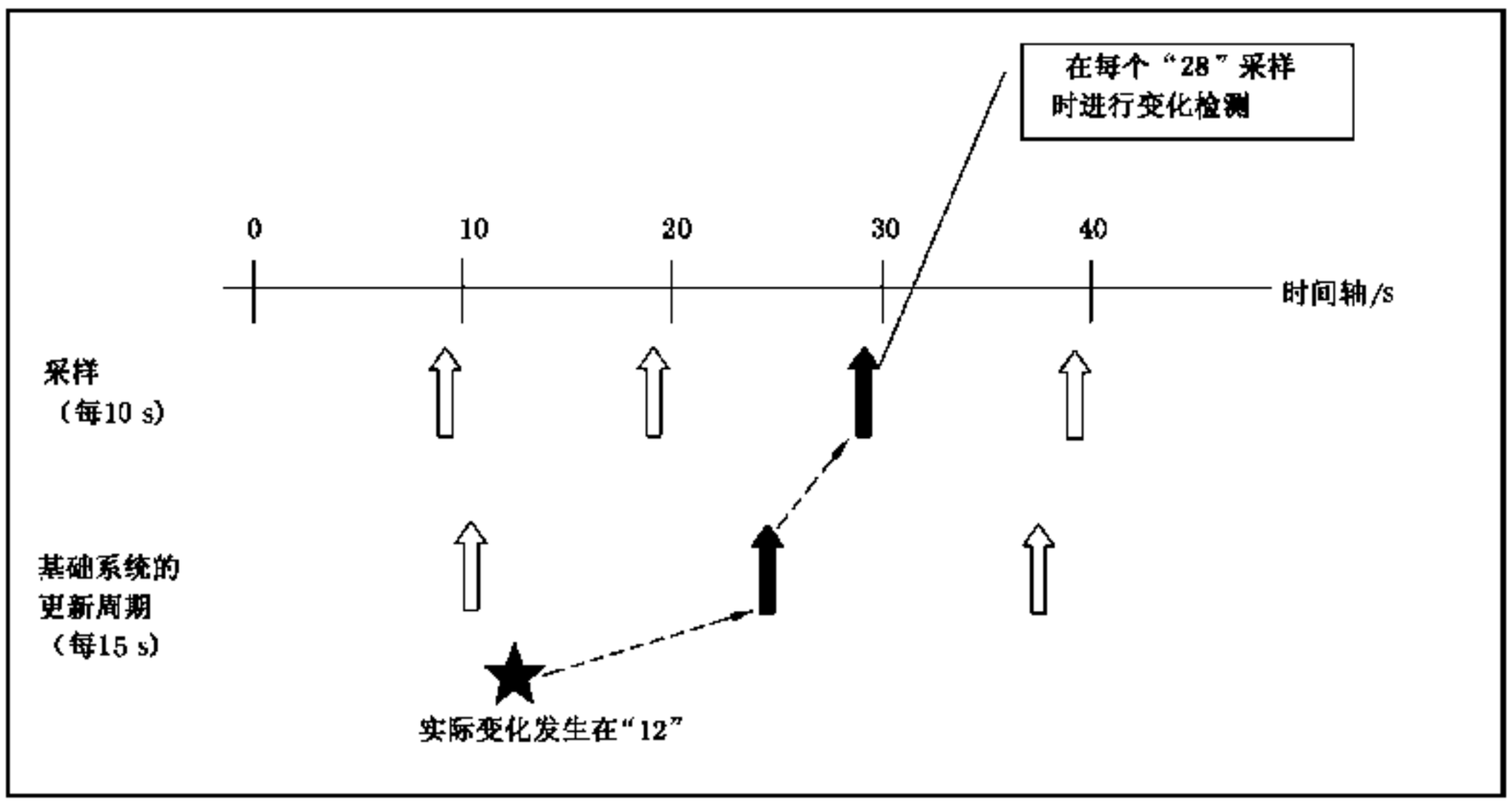

图16　变化检测中的典型延时

5.12.1.3　监视模式

监视模式参数是用来启用与禁止监视项的采样，也用来提供独立的启用与禁止通知(Notifications)的报告。这个功能允许配置监视项为采样，或采样与报告，或两者都不。禁止采样不会改变其他监视项参数的任何值，如采样间隔。

当监视项被启用(如当监视模型从DISABLED到SAMPLING或者PEPOTING变化)或者在启用状态下被创建，服务器该尽可能快地报告第一次的采样，且这个采样时间成为下个采样间隔的起点。

5.12.1.4　过滤器

每次监视项被采样，服务器使用为该监视项定义的过滤器来评估采样。过滤器参数定义了服务器

用来决定对采样是否产生通知的准则。过滤器类型依赖于被监视项类型。如,数据变化过滤器(DataChangeFilter)和聚合过滤器(AggregateFilter)是在监视变量值时使用的,事件过滤器(EventFilter)是在监视事件时使用的。采样和评估,以及过滤器的使用在本部分描述。额外的过滤器在本标准系列的其他部分定义。

5.12.1.5 队列参数

如果采样传送过滤器准则,那么为订阅传递产生一个通知,并排队等候。在创建监视项时定义队列的大小。当队列是满的且收到新通知,服务器要么放弃最老的通知,将新通知排队,要么它就直接放弃新通知。在创建监视项时就为它配置了这些丢弃策略中的一个。如果对于数据值的通知丢失,在数据值状态码的 InfoBits 部分中 Overflow 比特被置位。

如果队列大小是 1,且如果丢弃策略是丢弃最老的,那么该队列会成为一个总包含最新通知的缓冲器。在这种情况下,如果监视项的采样间隔比订阅的发布间隔要快,那么监视项将通过采样,且客户端将总收到最新的值。

另一方面,客户端可能打算订阅一个连续的无任何空隙的通知流,但不想它们在采样间隔上报告。在这种情况下,被创建的监视项应具有一个足够大的队列,以保持所有在两个连贯发布周期间产生的通知。然后,在每个发布周期,订阅向客户端发送监视项的所有通知队列。服务器应返回任何特殊项的通知,并在同样的通知所在的队列的顺序中。

服务器可能以比采样间隔更快的速度下采样,以此来支持其他客户端;客户端应该只期望在协商的采样间隔期间获取值。根据过滤器和实现约束,服务器可能会传送比采样间隔规定的更少的值。如果为某一个监视项配置了一个 DataChangeFilter(数据变化过滤器),那么相比当前采样,数据过滤器总是适用于该队列里最新的值。

例如:如果数据变化过滤器中的绝对死区是"10",那么队列中的值按下列顺序:

——100

——111

——101

89

——100

当使用死区过滤器且遇到变化的个数比能维护的值的个数要大时,数据队列可能会导致不期望的行为。由于丢弃策略"discardOldest=TRUE",实际中很有可能发生:发送到客户端的队列中新的第一个值可能不会超过先前值的死区限值。

当监视事件时,队列大小是服务器支持的最大值。在这种情况下,服务器负责事件的缓冲。如果事件丢失,那么 EventQueueOverflowEventType 类型的一个事件会产生并被放置。当第一个事件在监视项订阅事件上不得不被放弃时,这个事件就会产生。它被放入监视项的队列,增加为该监视项上定义的队列的大小,而不丢弃任何其他事件。如果 discardOldest 设置成 TRUE,那么它会被放置到序列的开始部分且不会被丢弃,否则将被放置在末尾。一个聚合服务器不该传递此类事件,应按其他连接错误情况处理。

5.12.1.6 触发模型

监视项(MonitoredItems)服务允许增加只在一些其他项(触发项)触发时报告的项。通过被触发项与要报告项间创建连接即可实现。要报告项的监视模式设置成只采样模式,那么它将采样以及排列通知,而无需报告它们。图 17 说明了这个概念。

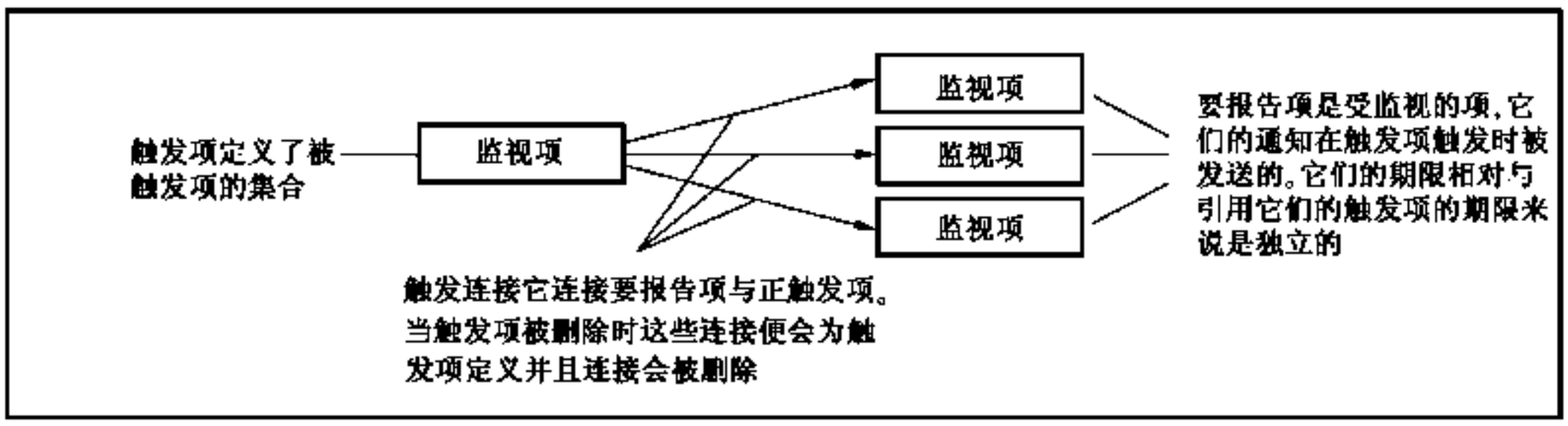

图 17　触发模型

触发机制是一个有用的特性。它允许客户端通过配置一些项为频繁采样但仅在一些其他事件发生时才报告的方法来减少线上传输的数据量。

规定了下述触发行为：

a) 触发项的监视模式指示报告被禁止。在这种情况下，触发项触发时不报告触发项。

b) 触发项的监视模式指示报告被启用。在这种情况下，当触发项触发时报告触发项。

c) 报告项的监视模式指示报告被禁止。在这种情况下，当触发项触发时报告项被报告。

d) 报告项的监视模式指示报告被启用。在这种情况下，要报告项只会报告一次(当报告项触发时)，触发项被忽略。

客户端在一个触发项和一组要报告项之间创建与删除触发连接。如果代表一个要报告项的监视项在与其相关触发连接被删除之前被删除，那么该触发连接也会被删除，但是触发项不受影响。

一个监视项的删除不应与它所监视的属性移除相混淆。如果包含正被监视的属性的节点被删除了，那么监视项会产生带有状态码 Bad_UnknownNodeId 的通知指示该删除，但监视项不会被删除。

5.12.2　CreateMonitoredItems

5.12.2.1　描述

该服务是用于向订阅创建和增加一个或多个监视项。当订阅被删除时，服务器自动删除监视项。删除监视项会引起它整个触发项链接集被删除，但对触发项引用的监视项不会受影响。

每次重复调用 CreateMonitoredItems 服务来增加小数量的监视项可能会对服务器的执行有不良影响。反之客户端要尽可能地向订阅增加一个完整的监视项集合。

当增加监视项时，服务器为其执行初始化过程。初始化过程是被正在监视项的通知类型定义的。通知类型是在本部分中以及在本系列标准中的访问类型规范部分规定，如 IEC 62541-8。见 IEC/TR 62541-1 第 4 章的访问类型部分的描述。

当用户增加一个监视项但对该项的读访问被拒绝时，那么对该项的增加操作应继续，且状态 Bad_NotReadable 或 Bad_UserAccessDenied 应该返回给发布响应。对访问权在调用 CreateMonitoredItem 之后发生变化的情况，上述行为同样适用。如果访问权改变为只读权，服务器该开始为监视项发送数据。

当在发布响应中发送通知时，CreateMonitoredItems 或上一个 ModifyMonitoredItems 服务的请求头中的返回诊断信息设置是用于被监视项，且用作诊断信息设置。

5.12.2.2　参数

表 64 定义该服务的参数。

表 64　创建监视项服务参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用的请求参数(见 7.26 中的定义)
subscriptionId	IntegerId	服务器分配的订阅的标识符,该订阅用于向该监视项报告通知(见 7.13 中的 IntegerId 定义)
timestampsToReturn	Enum TimestampsToReturn	一个枚举,规定了被传送给每个监视项的时间戳。TimestampsToReturn 枚举在 7.34 中定义。 当监视事件时,该参数仅用于 DataValue 类型的事件域
ItemsToCreate[]	MonitoredItemCreateRequest	要创建和分配给指定订阅的监视项列表
itemToMonitor	ReadValueId	在地址空间中识别要监视的项。为了对事件监视,ReadValue 结构的属性元素是 EventNotifier 属性的 Id。ReadValue 类型在 7.23 中定义
monitoringMode	Enum MonitoringMode	监视模式是为监视项而设置的。枚举类型 MonitoringMode 在 7.17 中定义
requestedParameters	Monitoring Parameters	请求的监视参数。服务器基于服务器的订阅和功能协商这些参数的值。MonitoringParameters 类型在 7.15 中定义的
响应(Reponse)		
responseHeader	Response Header	通用的响应参数(见 7.27 中的响应首部定义)
Results	MonitoredItemCreateResult	要创建的监视项的结果列表。列表的大小和顺序与 itemToCreate 请求参数的大小和顺序相匹配
statusCode	StatusCode	要创建的监视项的状态码(见 7.33 中的状态码定义)
monitoredItemId	IntegerId	服务器为监视项分配的 Id(见 7.13 中的 IntergerId 定义)。该 Id 在订阅中是惟一的,但可能在服务器或会话中不是惟一的。这个参数只有在 StatusCode 参数指示监视项成功创建的情况下才会出现
revisedSamplingInterval	Duration	服务器将要使用的实际采样间隔。 该值是基于一系列因数,包括底层系统的能力。服务器应总是返回 revisedSamplingInterval,该参数应大于或等于 requestedSamplingInterval。如果 requestedSamplingInterval 大于服务器所支持的最大采样间隔,那么返回最大采样间隔
revisedQueueSize	Counter	服务器将使用的实际队列长度
filterResult	Extensible Parameter MonitoringFilterResult	包含与请求中指定的 MonitoringFilter 相关的任何修改的参数值或错误结果。如果没有错误发生,则该参数可能会被省略。MonitoringFilterResult 参数类型是在 7.16 中定义的一个扩展参数类型
diagnosticInfos []	DiagnosticInfo	要创建的监视项的诊断信息列表(见 7.8 中的诊断信息定义)。列表的大小和顺序与 itemsToCreate 请求参数的大小和顺序相匹配。如果在请求首部中没有请求诊断信息,或如果在请求处理过程中没有产生诊断信息,那么该列表为空

5.12.2.3 服务结果

表 65 定义该服务特定的服务结果。通用的状态码在表 165 中定义。

表 65 创建监视项服务结果码

标识符	描述
Bad_NothingToDo	见表 165 中的该结果码描述
Bad_TooManyOperations	见表 165 中的该结果码描述
Bad_TimestampsToReturnInvalid	见表 165 中的该结果码描述
Bad_SubscriptionIdInvalid	见表 165 中的该结果码描述

5.12.2.4 状态码

表 66 定义了该服务特定的操作级状态码参数的值。通用状态码在表 166 中定义。

表 66 CreateMonitoredItems 操作级结果码

标识符	描述
Bad_MonitoringModeInvalid	见表 166 中该结果码的描述
Bad_NodeIdInvalid	见表 166 中该结果码的描述
Bad_NodeIdUnknown	见表 166 中该结果码的描述
Bad_AttributeIdInvalid	见表 166 中该结果码的描述
Bad_IndexRangeInvalid	见表 166 中该结果码的描述
Bad_IndexRangeNoData	见表 166 中该结果码的描述
Bad_DataEncodingInvalid	见表 166 中该结果码的描述
Bad_DataEncodingUnsupported	见表 166 中该结果码的描述
Bad_UserAccessDenied	见表 166 中该结果码的描述
Bad_MonitoredItemFilterInvalid	见表 166 中该结果码的描述
Bad_MonitoredItemFilterUnsupported	见表 166 中该结果码的描述
Bad_FilterNotAllowed	见表 166 中该结果码的描述

5.12.3 ModifyMonitoredItems

5.12.3.1 描述

该服务用于修改订阅的监视项。采样间隔和过滤器的变化会在下个采样间隔开始时生效(下次采样计时器期满)。

当在发布响应中发送通知时,在 CreateMonitoredItems 的请求首部中或在上一个 ModifyMonitoredItems 服务返回的诊断信息设置是提供给监视项的,且用作诊断信息设置。

5.12.3.2 参数

表 67 定义该服务的参数。

表 67　ModifyMonitoredItems 服务参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用请求参数(见 7.26 中的 RequestHeader 定义)
subscriptionId	IntegerId	服务器分配的订阅的标识,可用于验证 monitoredItemId(见 7.13 中的 IntegerId 定义)
timestampsToReturn	Enum TimestampsToReturn	为每个要修改的监视项规定了要传输的时间戳属性的枚举。TimestampsToReturn 枚举在 7.34 中定义。当监视事件时,该参数仅用于 DataValue 类型的事件域
itemsToModify	MonitoredItemModifyRequest	要修改的监视项列表
monitoredItemId	IntegerId	服务器为监视项分配的 Id
requestedParameters	MonitoringParameters	为监视参数的请求值。MonitoringParameter 类型在 7.14 中定义
响应(Response)		
responseHeader	ResponseHeader	通用响应参数(见 7.27 中的定义)
Results[]	MonitoredItemModifyResult	要修改的为监视项结果列表。列表的大小和顺序与请求参数 itemToModify 的大小和顺序相匹配
statusCode	StatusCode	要修改的监视项的状态码(见 7.33 中的状态码定义)
revisedSamplingInterval	Duration	服务器将要使用的实际采样间隔。服务器返回的值实际上用作采样间隔。该值是基于一些因数,包括底层系统的能力。服务器应总是返回一个大于或等于 requestedSamplingInterval 的 revisedSamplingInterval。如果 requestedSamplingInterval 大于服务器支持的最大采样间隔,那么最大的采样间隔会被返回
revisedQueueSize	Counter	服务器将要使用的实际队列的大小
filterResult	Extensible Parameter MonitoringFilterResult	包含与请求中指定的 MonitoringFilter 相关的任何修改参数值或错误结果。如果无错误发生,那么这个参数可被忽略。MonitoringFilterResult 参数类型是在 7.16 中定义的扩展参数类型
diagnosticInfos []	DiagnosticInfo	要修改的监视项的诊断信息列表(见 7.8 中的诊断信息定义)。列表的大小和顺序与请求参数 itemsToModify 的大小和顺序相匹配。如果诊断信息在请求首部中没有请求或如果在请求处理过程中无诊断信息,那么该列表为空

5.12.3.3　服务结果

表 68 定义了此服务特定的服务结果。通用的状态码在表 165 中有定义。

5.12.3.4　状态码

表 69 定义该服务特定的操作级状态码参数。通用的状态码在表 166 中有定义。

表 68　调试监视项服务结果码

标识符	描述
Bad_NothingToDo	见表 165 中的该结果码的描述
Bad_TooManyOperations	见表 165 中的该结果码的描述
Bad_TimestampsToReturnInvalid	见表 165 中的该结果码的描述
Bad_SubscriptionIdInvalid	见表 165 中的该结果码的描述

表 69　调试监视项操作水平结果码

标识符	描述
Bad_MonitoredItemIdInvalid	见表 166 中该结果码的描述
Bad_MonitoredItemFilterInvalid	见表 166 中该结果码的描述
Bad_MonitoredItemFilterUnsupported	见表 166 中该结果码的描述
Bad_FilterNotAllowed	见表 166 中该结果码的描述

5.12.4　SetMonitoringMode

5.12.4.1　描述

此服务用于为一个订阅的一个或多个监视项来设置监视模式。设置模式为 DISABLED 将导致所有已排队的序列删除。

5.12.4.2　参数

表 70 定义了该服务的参数。

表 70　SetMonitoringMode 服务参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用请求参数(见 7.26 中的 RequestHeader 定义)
subscriptionId	IntegerId	服务器分配的订阅的标识,可用于验证 monitoredItemId(见 7.13 中的 IntegerId 定义)
monitoringMode	Enum MonitoringMode	为监视项而设置监视模式,监视模式的枚举值在 7.17 中定义的
monitorItemIds []	IntegerId	服务器为监视项分配的 Id 列表
响应(Response)		
responseHeader	ResponseHeader	通用的响应参数(7.27 中的 responseHeader 定义)
results[]	StatusCode	要使能/禁止的监视项状态码列表(见 7.33 中的状态码定义)。列表的大小和顺序与请求参数 monitoredItemIds 的大小和顺序相匹配

表 70（续）

名称	类型	描述
diagnosticInfos[]	DiagnosticInfo	要使能/禁止的监视项的诊断信息列表（见 7.8 中的 DiagnosticInfo 定义）。列表的大小和顺序与请求参数 monitoredItemIds 的大小和顺序相匹配。如果诊断信息在请求首部中未被请求或在请求处理过程中无诊断信息，则列表为空

5.12.4.3 服务结果

表 71 定义该服务对应的服务结果。通用的状态码在表 165 中定义。

表 71 SetMonitoringMode 服务结果码

标识符	描述
Bad_NothingToDo	见表 165 该结果码描述
Bad_TooManyOperations	见表 165 该结果码描述
Bad_SubscriptionIdInvalid	见表 165 该结果码描述
Bad_MonitoringModeInvalid	见表 165 该结果码描述

5.12.4.4 状态码

表 72 为该服务特定的操作标准状态码参数定义了值。通用的状态码在表 166 中有定义。

表 72 设置监视模式操作标准结果码

值	描述
Bad_MonitoredItemIdInvalid	见表 166 中该结果码的描述

5.12.5 SetTriggering

5.12.5.1 描述

此服务用于为触发项目创建与删除触发链接。触发项目和要报告项目应属于相同的订阅。

每个触发链接将触发项与报告项连接起来。对于报告项目，每个链接由监视项 Id 表示。如果此 Id 无效，则返回错误码。

5.12.5.2 参数

表 73 定义了该服务的参数。

表 73 SetTriggering 服务参数

名称	类型	描述
请求(Request)		

表 73（续）

名称	类型	描述
requestHeader	RequestHeader	通用的请求参数（见 7.26 中的 requestHeader 定义）
subscriptionId	IntegerId	服务器为订阅分配的标识符，该订阅包含触发项和报告项（见 7.13 中 IntegerId 定义）
triggeringItemId	IntegerId	服务器为作为触发项的监视项分配的 Id
linksToAdd []	IntegerId	服务器为报告项分配的 Id 列表，该报告项将作为触发链接增加
linksToRemove []	IntegerId	服务器为报告项分配的 Id 列表，该报告项将作为要删除的触发链接
响应（Response）		
responseHeader	ResponseHeader	通用的响应参数（见 7.27 中的 ResponseHeader 定义）
AddResults[]	StatusCode	为项增加的状态码列表（见 7.33 中的状态码定义）。 列表的大小和顺序与请求中规定的参数 linksToAdd 的大小和顺序相匹配
AddDiagnosticInfos[]	DiagnosticInfo	为链接增加的诊断信息列表（见 7.8 中的诊断信息定义）。列表的大小和顺序与请求参数 linksToAdd 的大小和顺序相匹配。如果诊断信息在请求首部中未被请求或在请求处理过程中无诊断信息，则列表为空
RemoveResults[]	StatusCode	要删除项的状态码列表。列表的大小和规则与请求规定的参数 linksToDelete 的大小和顺序相匹配
RemoveDiagnosticInfos[]	DiagnosticInfo	要删除链接的诊断信息列表。列表的大小和顺序与请求参数 linksToDelete 的大小和顺序相匹配。如果诊断信息在请求首部中未被请求或在请求处理过程中无诊断信息，则列表为空

5.12.5.3 服务参数

表 74 定义了该服务特定的服务结果。通用状态码在 7.33 中定义。

表 74 SetTriggering 服务结果码

标识符	描述
Bad_NothingToDo	见表 165 中此结果码描述
Bad_TooManyOperations	见表 165 中此结果码描述
Bad_SubscriptionIdInvalid	见表 165 中此结果码描述
Bad_MonitoredItemIdInvalid	见表 165 中此结果码描述

5.12.5.4 状态码

表 75 定义了此服务特定的结果参数值。通用的状态码在表 166 中定义。

表 75　设置触发操作标准结果码

标识符	描述
Bad_MonitoredItemIdInvalid	见表 166 中此结果码描述

5.12.6　DeleteMonitoredItems

5.12.6.1　描述

此服务是用来删除一个订阅中的一个或多个监视项。当删除一个监视项时，它的触发项链接也会被删除。

然而，一个监视项的成功删除可能不会为订阅正发送过程中的监视项删除通知。因此，在客户端已经收到一个监视项删除的正响应后，客户端可能会收到针对监视项的通知。

5.12.6.2　参数

表 76 定义了针对服务的参数。

表 76　DeleteMonitoredItems 服务参数

名称	类型	描述
请求(Request)		
requestHeader	RequestHeader	通用请求参数(见 7.26 中的 RequestHeader 定义)
subscriptionId	integerId	服务器为包含要删除监视项的订阅分配的标识符(见 7.13 中 integerId 定义)
monitoredItemIds	integerId	服务器为要被删除的监视项分配 Id 的列表
响应(Response)		
responseHeader	ResponseHeader	通用的响应参数(见 7.27 中的 ResponseHeader 定义)
Results	StatusCode	要删除的监视项的状态码列表(见 7.33 中的状态码定义)。列表的大小和顺序与请求参数 monitoredItemIds 的大小和顺序相匹配
diagnosticInfos	DiagnosticInfo	要删除的监视项的诊断信息列表(见 7.8 中的 DiagnosticInfo 定义)。列表的大小和顺序与请求参数 monitoredItemIds 的大小和顺序相匹配。如果诊断信息在请求首部中未被请求或在请求处理过程中无诊断信息，则列表为空

5.12.6.3　服务结果

表 77 定义了此服务特定的服务结果。通用的状态码在表 165 中定义。

5.12.6.4　状态码

表 78 定义了针对此服务特定的结果参数的值。通用的状态码在表 166 中定义。

表 77　DeleteMonitoredItems 服务结果码

标识符	描述
Bad_NothingToDo	见表 165 中此结果码的描述
Bad_TooManyOperations	见表 165 中此结果码的描述
Bad_SubscriptionIdInvalid	见表 165 中此结果码的描述

表 78　删除监视项操作标准结果码

标识符	描述
Bad_MonitoredItemIdInvalid	见表 166 中此结果码的描述

5.13　Subscription(订阅)服务集

5.13.1　订阅模型

5.13.1.1　描述

订阅用于将通知报告给客户端，其通用行为概述如下，更详细行为在 5.13.1.2 中描述。

a)　订阅拥有由客户端分配给它的监视项的集合，监视项产生通知，通知由订阅报告给客户端(监视项的描述见 5.12.1)。

b)　订阅有一个发布间隔。订阅的发布间隔定义了订阅执行的周期速度。每次执行时，它总尝试发送一个通知消息(NotificationMessage)给客户端。通知消息包含了还没报告给客户端的通知。

c)　通知消息在发布(Publish)请求的响应中被发送给客户端。在接收到发布请求时通常将它们排队进入该会话，对于每个发布周期如果有通知要报告，则有一个发布请求被出列并由与此会话相关联的订阅来处理。当无通知报告时，发布请求不会从会话中出列，且服务器会一直等到下个周期，并再次检查通知。

d)　在一个周期的开始，如果有通知要发送，但没有发布请求在排队，那么服务器进入等待状态，等待接收一个发布请求。当收到请求时，不用等到下个发布周期该请求就会立即被处理。

e)　通知消息是惟一通过序列号标识的，序列号使能客户端检测丢失报文。发布间隔也为它的监视项定义缺省的采样间隔。

f)　订阅有个保持活动的计数器，该计数器对连续的已经没有通知要报告给客户端的发布周期进行计数。当达到最大活动数时，那么一个发布请求会出列且会被用来返回一个保持活动的消息。这个保持活动的消息通知客户端，订阅仍然是活动的。每个保持活动的消息是对一个发布请求作出的响应，在这个发布请求中，notificationMessage(通知消息)参数不包含任何通知，它包含了下一个将要发送的通知消息的序列号。在后续条中，通知消息作为对发布请求的响应，该发布请求中参数 notificationMessage 实际包含一个或多个通知。这与保持激活消息不同，在保持活动消息中该参数不包含通知。最大保持活动数在订阅创建期间由客户端设置，且可使用 ModifySubscription 服务进行后续修改。这与在上面 c)中所描述的通知处理相似，如果无发布请求在排队，则服务器会等待接收下一个发布请求，且无需等到下一个发布周期就立即发送保持活动。

g)　订阅的发布可由客户端在创建时或在后续使用 SetPublishingMode 服务时来启用或禁用。禁

用可使订阅停止发送通知报文给客户端。然而，订阅会继续周期性地执行且会继续发送保持活动的消息给客户端。

h) 订阅有个生命周期计数器，对连续发布的没有从客户端收到发布请求的周期进行计数。当这个计数器达到订阅生命周期的值时订阅会关闭。该值是基于 CreateSubscription 服务(5.13.2)中的 MaxKeepAliveCount 计算得到。关闭订阅可使它的监视项被删除。此外，服务器发布带有状态码 Bad_Timeout 的通知消息 StatusChangeNotification。通知消息 StatusChangeNotification 的通知消息类型在 7.19.4 定义。

i) 订阅维护一个发送通知报文的重发队列。通知报文被保持在这个序列，直到它们被确认或它们已经在处于最小活动间隔的队列中为止。要求客户端收到通知消息时，对其进行确认。

序列号是一个无符号 32 位的整数，对于每个发送的通知报文加 1。值 0 不会用作序列号。在订阅中发送的第一个通知报文具有序列号 1。如果序列号运转完毕，那么该值为 1。

当创建了一个订阅时，在第一个消息会在第一个发布周期结尾发送，来通知客户端订阅是可操作的。如果有通知准备好要报告，那么会发送通知消息。如果没有通知，则替代发送一个保持活动的报文，该报文序列号为 1，指示第一个通知报文还没有被发送。这是惟一一次无需达到最大保持活动计数就发送保持活动报文，如在上述 f)中描述。

序列号的值在订阅的生命周期期间是不会被重置的。因此，同样的序列号不该在一个订阅上重复使用，直到已经发送了超过 40 亿个通知为止。在一个给定的订阅上，以每秒一千个的速度发送通知消息，则将花费大约 50 天时间才能重复使用相同序列号。所以客户端将序列号作为惟一来考虑是安全的。

序列号也被客户端用来确认已收到通知消息。发布请求允许客户端根据特定序列号确认所有通知号，并确认上一个收到的通知消息的序列号。可能存在一个或多个序列号间隔。确认允许服务器从它的重发队列中删除通知消息。

客户端可使用 Republish 服务要求重新发送所选择的通知消息。此服务返回所请求的报文。

5.13.1.2 状态表

状态表形式上描述了订阅的操作。下述的操作模型通过状态表来描述。当发布对订阅被启用或禁用时，此描述适用。

在创建订阅后，服务器启动发布的定时器并在溢出时重启定时器。如果当定时器值超过了为订阅生命周期定义的时间计数限值时，还没有从客户端收到一个 Subscription 服务请求，那么订阅假设该客户端不再存在，并终止该服务。

客户端向服务器发送发布请求来接收通知。发布请求不是直接指向任何一个订阅，但可被任一个订阅所使用。每个发布都包含一个或多个订阅的确认。当发布请求收到后处理这些确认。然后服务器将所有订阅共享的队列中的请求进行排队，除了以下几种情况外：

a) 先前的发布响应指示仍然有更多通知准备被发送并且没有更多发布请求排队等待发送；

b) 订阅的发布定时器溢出，且有要发送的订阅或要发送的保持活动消息。

在情况 a)或 b)下，新收到的发布请求立即被第一个订阅进行处理。

每次发布定时器溢出时，就对定时器立即重置。如果有要发送的通知或保持活动消息，那么订阅会出列且处理一个发布请求。当订阅处理发布请求时，它访问监视项的队列并从队列中取出它的通知(如果有的话)。如果它不能返回在响应中所有可用的通知，那么它返回在响应中的这些通知，并设置 moreNotifications 标志。

如果有通知或保持活动消息要被发送但没有排队的发布请求，那么订阅假设发布请求过期并等待下一个要接收的发布请求，如情况 b)中描述。

当在保持活动状态下，在每次发布计时器溢出时对通知进行检查。如果已经产生一个或多个通知，

那么一个发布请求会出列，并且在响应中返回一个通知消息。然而，如果发布定时器在没有通知成为可用的情况下溢出，那么发布请求会出列并且一个保持活动消息在响应中返回。然后订阅返回到再次等待发布定时器溢出的正常状态。如果在这些情况中的任意一种，没有排队的发布请求，那么订阅会等待收到的下一个发布请求，如在情况 b)中描述。

订阅状态在表 79 中定义。

表 79　订阅状态

状态	描述
CLOSED	订阅还没创建或已终止
CREATING	订阅正被创建
NORMAL	订阅周期性检查来自监视项的通知。保持活动计数器在此状态下不使用
LATE	当发布定时器器已溢出，并且有可用通知或保持活动的报文准备发送，但还没有排队的发布请求。在这种状态下，当下个发布请求收到时它会被处理。保持活动计数器在这种状态下不使用
KEEPALIVE	订阅周期性检查来自其监视项的通知或检查保持活动计数器从最大值变为 0

状态表在表 80 中描述，以下规则和约定适用：

a) 事件表示接收到服务请求及发生内部事件，如定时器溢出；

b) 服务请求事件可能伴随着测试服务参数值条件。参数名以小写字符开始；

c) 内部事件可能伴随着测试状态变量值条件。状态变量在 5.13.1.3 中定义，以大写字符开始；

d) 服务请求和内部事件可能伴随着由函数表示的条件，该函数的返回值被测试。函数表示为在名称之后加“()”，见 5.13.1.4 中描述；

e) 当收到一个事件时，当前状态的第一次转换被定位，针对第一次状态转化顺序搜索满足事件和条件规则的后续转换。如果一个都没找到，则忽略该事件；

f) 动作是由函数和状态变量操作描述；

g) 当对应的计数器到 0 时触发 LifetimeTimerExpires 事件。

表 80　订阅声明表

#	当前状态	事件/条件	动作	下个状态
1	CLOSED	接收创建订阅请求	CreateSubscription()	CREATING
2	CREATING	创建订阅失败	ReturnNegativeResponse()	CLOSED
3	CREATING	创建订阅成功	InitializeSubscription() MessageSent = FALSE ReturnResponse()	NORMAL
4	NORMAL	接收发布请求 && (PublishingEnabled == FALSE || (PublishingEnabled == TRUE && MoreNotifications == FALSE))	ResetLifetimeCounter() DeleteAckedNotificationMsgs() EnqueuePublishingReq()	NORMAL

表 80（续）

#	当前状态	事件/条件	动作	下个状态
5	NORMAL	接收发布请求 &&. PublishingEnabled —— TRUE &&. MoreNotifications == TRUE	ResetLifetimeCounter() DeleteAckedNotificationMsgs() ReturnNotifications() MessageSent = TRUE	NORMAL
6	NORMAL	发布定时器溢出 &&. PublishingReqQueued TRUE &&. PublishingEnabled == TRUE &&. NotificationsAvailable TRUE	StartPublishingTimer() DequeuePublishReq() ReturnNotifications() MessageSent TRUE	NORMAL
7	NORMAL	发布定时器溢出 &&. PublishingReqQueued TRUE &&. MessageSent == FALSE &&. (PublishingEnabled FALSE || (PublishingEnabled == TRUE &&. NotificationsAvailable —— FALSE))	StartPublishingTimer() DequeuePublishReq() ReturnKeepAlive() MessageSent == TRUE	NORMAL
8	NORMAL	发布定时器溢出 &&. PublishingReqQueued == FALSE &&. (MessageSent == FALSE || (PublishingEnabled == TRUE &&. NotificationsAvailable TRUE))	StartPublishingTimer()	LATE
9	NORMAL	发布定时器溢出 &&. MessageSent TRUE &&. (PublishingEnabled FALSE || (PublishingEnabled == TRUE &&. NotificationsAvailable FALSE))	StartPublishingTimer() ResetKeepAliveCounter()	KEEPALIVE
10	LATE	接收发布请求 &&. PublishingEnabled TRUE &&. (NotificationsAvailable == TRUE || MoreNotifications —— TRUE)	ResetLifetimeCounter() DeleteAckedNotificationMsgs() ReturnNotifications() MessageSent — TRUE	NORMAL

表 80（续）

#	当前状态	事件/条件	动作	下个状态
11	LATE	接收发布请求 && (PublishingEnabled == FALSE || (PublishingEnabled == TRUE && NotificationsAvailable == FALSE && MoreNotifications == FALSE))	ResetLifetimeCounter() DeleteAckedNotificationMsgs() ReturnKeepAlive() MessageSent = TRUE	KEEPALIVE
12	LATE	发布定时器溢出	StartPublishingTimer()	LATE
13	KEEPALIVE	接收发布请求	ResetLifetimeCounter() DeleteAckedNotificationMsgs() EnqueuePublishingReq()	KEEPALIVE
14	KEEPALIVE	发布定时器溢出 && PublishingEnabled == TRUE && NotificationsAvailable TRUE && PublishingReqQueued == TRUE	StartPublishingTimer() DequeuePublishReq() ReturnNotifications() MessageSent == TRUE	NORMAL
15	KEEPALIVE	发布定时器溢出 && PublishingReqQueued TRUE && KeepAliveCounter == 1 && (PublishingEnabled FALSE || (PublishingEnabled == TRUE && NotificationsAvailable == FALSE)	StartPublishingTimer() DequeuePublishReq() ReturnKeepAlive() ResetKeepAliveCounter()	KEEPALIVE
16	KEEPALIVE	发布定时器溢出 && KeepAliveCounter > 1 && (PublishingEnabled == FALSE || (PublishingEnabled == TRUE && NotificationsAvailable == FALSE))	StartPublishingTimer() KeepAliveCounter--	KEEPALIVE
17	KEEPALIVE	发布定时器溢出 && PublishingReqQueued == FALSE && (KeepAliveCounter == 1 || (KeepAliveCounter > 1 && PublishingEnabled == TRUE && NotificationsAvailable TRUE))	StartPublishingTimer()	LATE

表 80（续）

卡	当前状态	事件/条件	动作	下个状态
18	NORMAL ||LATE ||KEEPALIVE	接收修改订阅请求	ResetLifetimeCounter() UpdateSubscriptionParams() ReturnResponse()	SAME
19	NORMAL ||LATE ||KEEPALIVE	接收设置发布模式请求	ResetLifetimeCounter() SetPublishingEnabled() MoreNotifications FALSE ReturnResponse()	SAME
20	NORMAL ||LATE ||KEEPALIVE	接收重发布请求 && RequestedMessageFound = = TRUE	ResetLifetimeCounter() ReturnResponse()	SAME
21	NORMAL ||LATE ||KEEPALIVE	接收重发布请求 && RequestedMessageFound = = FALSE	ResetLifetimeCounter() ReturnNegativeResponse()	SAME
22	NORMAL ||LATE ||KEEPALIVE	接收发送订阅请求 && SessionChanged() == FALSE	ResetLifetimeCounter() ReturnNegativeResponse ()	SAME
23	NORMAL ||LATE ||KEEPALIVE	接收发送订阅请求 && SessionChanged() == TRUE && ClientValidated() ==TRUE	SetSession() ResetLifetimeCounter() DeleteAckedNotificationMsgs() ReturnResponse() IssueStatusChangeNotification()	SAME
24	NORMAL ||LATE ||KEEPALIVE	接收发送订阅请求 && SessionChanged() == TRUE && ClientValidated() == FALSE	ReturnNegativeResponse()	SAME
25	NORMAL ||LATE ||KEEPALIVE	接收删除订阅请求 && SubscriptionAssignedToClient = =TRUE	DeleteMonitoredItems() DeleteClientPublReqQueue()	CLOSED
26	NORMAL ||LATE ||KEEPALIVE	接收删除订阅请求 && SubscriptionAssignedToClient = =FALSE	ResetLifetimeCounter() ReturnNegativeResponse()	SAME
27	NORMAL ||LATE ||KEEPALIVE	LifetimeCounter == 1 如果发布定时器溢出，且 PublishingReqQueued == FALSE，则 LifetimeCounter 递减； 如果 PublishingReqQueued == TRUE，则 LifetimeCounter 复位	DeleteMonitoredItems() IssueStatusChangeNotification()	CLOSED

5.13.1.3 状态变量和参数

状态变量在表 81 中按字母次序定义。

表 81　声明变量和参数

状态变量	描述
MoreNotifications	布尔值，当对一个通知消息有太多的通知时，仅通过 CreateNotificationMsg()将该值设成 TRUE
LatePublishRequest	布尔值，该值被设置成 TRUE 来反应最后一次发布定时器溢出，无排队的发布请求
LifetimeCounter	在订阅终止前无客户端动作情况下，连续的发布定时器溢出次数的值
MessageSent	布尔值，该值被设置成 TRUE 来表示，通知消息或是保持活动消息已发送到订阅。它是一个标志，用来确保在第一次发布定时器溢出时通知消息或保持活动消息报文被发送出去
NotificationsAvailable	布尔值，只有当至少有一个在报告模式及有在排队的通知的监视项，或者当至少有一个报告项，且该项的触发项已经触发并有在排队的通知时，该布尔值会被设置为 TRUE。该状态变量从 FALSE 转换到 TRUE 在状态表中产生“排队的新通知”事件
PublishingEnabled	请求发布被许可或禁止的参数
PublishingReqQueued	仅当发布请求消息入列到订阅时被设置成 TRUE 的布尔值
RequestedMessageFound	仅当被请求重传的消息在重传队列被找到时被设置成 TRUE 的布尔值
SeqNum	记录了在通知消息中使用的序列号的值
SubscriptionAssignedToClient	仅当请求删除的订阅被分配给发布请求的客户端时被设置为 TRUE 的布尔值。订阅被分配给创建它的客户端。只有通过成功完成 TransferSubscriptions 服务才能改变分配

5.13.1.4　函数

行为函数在表 82 中按字母顺序定义。

表 82　函数

声明	描述
BindSession()	将与订阅有关的客户端会话与用来发送正被处理的服务的客户端会话绑定一起。如果这是与先前客户端绑定的最后一个订阅，则消除被先前客户端发送的所有发布请求的发布请求队列，并对每个请求返回否定响应
ClientValidated()	布尔函数，仅当正在递交发送订阅请求的客户端代表相同用户正进行操作，且支持与之前会话的客户端相同的行规，则返回 TRUE
CreateNotificationMsg()	增加序列号并为分配给订阅的监视项中创建通知消息。 将新创建的通知消息保存在重发队列中。 如果所有可用通知能在发布响应中发送，则更多通知状态变量会被设置成 FALSE，否则，设置成 TRUE
CreateSubscription()	尝试创建订阅
DeleteAckedNotificationMsgs()	从被请求确认的重发队列中删除通知消息

表 82（续）

声明	描述
DeleteClientPublReqQueue()	如果不再有分配给客户端的订阅，则对正发送删除订阅请求的客户端的发布请求队列进行清除
DeleteMonitoredItems()	删除所有分配给订阅的监视项
DequeuePublishReq()	按先进先出的顺序从队列中取出发布请求。 通过在请求首部中检查 timeoutHint 确认发布请求是否仍然是有效。 如果请求超时，那么给请求发送一个 Bad_Timeout 服务结果，并从队列中取出另一个发布请求
EnqueuePublishingReq()	将发布请求加入队列
InitializeSubscription()	ResetLifetimeCounter() MoreNotifications = FALSE PublishRateChange = FALSE PublishingEnabled = 在 CreateSubscription 请求中参数 publishingEnabled 的值 PublishingReqQueued = FALSE SeqNum = 0 SetSession() StartPublishingTimer()
IssueStatusChangeNotification()	为订阅的状态改变，发布带有状态码的 StatusChangeNotification 通知消息。StatusChangeNotification 的通知消息类型在 7.19.4 中有定义。生命周期超时则使用 Bad_Timeout 状态码。如果订阅被传递给另一个会话，则使用 Good_SubcriptionTransferred
ResetKeepAliveCounter()	将保持获得计数器复位为订阅的最大保持活动计数。最大保持活动数目是在订阅被创建时由客户端设置，并可使用 ModifySubscription 服务进行修改
ResetLifetimeCounter()	将 LifetimeCounter 变量复位为在 CreateSubscription 服务（5.13.2）中订阅生命周期规定的值
ReturnKeepAlive()	CreateKeepAliveMsg() ReturnResponse()
ReturnNegativeResponse ()	返回一个指示适当服务级错误的服务响应。只有包含服务级状态码的 ResponseHeader 参数被返回

ReturnNotifications()	CreateNotificationMsg()
	ReturnResponse()
	If (MoreNotifications == TRUE) && (PublishingReqQueued == TRUE)
	{
	DequeuePublishReq()
	Loop through this function again
	}

ReturnResponse()	返回合适的响应，设置合适的参数值和为服务定义的状态码

表 82（续）

声明	描述
SessionChanged()	布尔函数.仅当用来发送订阅请求传输的会话不同于与目前订阅相关的客户端会话时，该布尔函数返回 TRUE
SetPublishingEnabled ()	将 PublishingEnabled 状态变量设置为请求中收到的 PublishingEnabled 参数值
SetSession	对订阅的会话信息进行设置.使其与发布发送订阅请求的会话相匹配
StartPublishingTimer()	开始或重新开始发布定时器并递减 LifetimeCounter 变量
UpdateSubscriptionParams()	协商和更新订阅参数。如果新的保持活动间隔小于保持活动计数器的当前值.则执行 ResetKeepAliveCounter() 和 ResetLifetimeCounter()

5.13.2 CreateSubscription

5.13.2.1 描述

此服务用于创建订阅。订阅对通知的监视项集进行监视，并向客户端返回监视项作为对发布请求的响应。

5.13.2.2 参数

表 83 定义了服务的参数。

表 83 CreateSubscription 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用的请求参数(见 7.26 中的 RequestHeader 定义)
requestedPublishing Interval	Duration	此间隔定义了订阅正被请求并返回通知给客户端的周期速度。此间隔以毫秒来表示。在订阅状态表(见 5.13.1.2)中.此间隔是由发布定时器表示。 在响应中返回的该参数的协商值作为分配给此订阅的监视项的缺省采样间隔。该值 0 无效
requestedLifetimeCount	Counter	请求的生命时间计数(见 7.5 中的计数器定义)。生命时间计数该是三次保持活动计数中的最小值。 当发布定时器已经超过此时间数，且没有可用的发布请求用于发送通知消息，则服务器应删除该订阅
requestedMaxKeep Alive Count	Counter	请求的最大保持活动计数(见 7.5 中的计数器定义)。当发布定时器已经超过此时间数，且没要求发送任何通知消息，则订阅会给客户端发送一个保持活动消息。 值 0 是无效的
maxNotifications PerPublish	Counter	客户端在一个发布响应中希望收到的最大通知个数。0 值指示无限值

表 83（续）

名称	类型	描述
publishingEnabled	Boolean	具有以下值的布尔型参数： TRUE　订阅的发布被允许 FALSE　订阅的发布被禁止 此参数值不影响监视项的监视模式属性
Priority	Byte	指示订阅的相对优先权。当多个订阅需要发送通知时，则服务器应从队列中取出发布请求给拥有最高优先权的订阅。对有相同优先权的订阅，服务器应在一个循环模式中从队列取出发布请求。当对于一个订阅的保持活动周期超时时，则该订阅应优先，而不管它的实际优先权，以避免订阅超时。 没要求特殊优先权设置的客户端该设置值为 0
响应		
responseHeader	ResponseHeader	通用的响应参数（见 7.27 中的 ResponseHeader 定义）
subscriptionId	IntegerId	服务器为订阅分配的标识符（见 7.13 中的 IntegerId 定义）。此标识符对于整个服务器是惟一的，不仅仅对于会话，目的是允许订阅发送给使用 TransferSubscriptions 服务的另一个会话
revisedPublishingInterval	Duration	服务器将使用的实际发布间隔，以毫秒来表述。服务器应尝试为该参数向客户端请求授权，但可将此值向上或向下进行协商以满足它自身的约束
revisedLifetimeCount	Counter	订阅的生命时间应是服务器协商的三次保持活动间隔中最小值
revisedMaxKeepAliveCount	Counter	实际最大的保持活动计数（见 7.5 中计数器定义）。服务器应该尝试为该参数给客户端请求授权，但可将此值向上或向下协商以满足它自身的约束

5.13.2.3　服务结果

表 84 定义此服务特定的服务结果。通用的状态码见表 165 中定义。

表 84　CreateSubscription 服务结果码

符号 Id	描述
Bad_TooManySubscriptions	服务器已经达到订阅的最大数量

5.13.3　ModifySubscription

5.13.3.1　描述

此服务用于修改订阅。

5.13.3.2　参数

表 85 定义了服务的参数。发布间隔的变化在下次发布定时器溢出的时候会生效。

表 85　ModifySubscription 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用的请求参数(见 7.26 中的 RequestHeader 定义)
subscriptionId	IntegerId	服务器为订阅分配的标识符(见 7.13 中的 IntegerId 定义)
requestedPublishingInterval	Duration	此间隔定义了订阅正被请求并返回通知给客户端的周期速度。此间隔以毫秒来表示。在订阅状态表(见 5.13.1.2)中,此间隔是由发布定时器表示。 在响应中返回的该参数的协商值作为分配给此订阅的监视项的缺省采样间隔。 该值 0 无效
requestedLifetimeCount	Counter	请求的生命时间计数(见 7.5 中的计数器定义)。生命时间计数该是三次保持活动计数中的最小值。 当发布定时器已经超过此时间数,且没有可用的发布请求用于发送通知消息,则服务器应删除该订阅
requestedMaxKeepAliveCount	Counter	请求的最大保持活动计数(见 7.5 中的计数器定义)。当发布定时器已经超过此时间数,且没要求发送任何通知消息,则订阅会给客户端发送一个保持活动消息。 值 0 是无效的
maxNotificationsPerPublish	Counter	客户端在一个发布响应中希望收到的最大通知个数。0 值指示无限值
priority	Byte	指示订阅的相对优先权。当多个订阅需要发送通知时,则服务器应从队列中取出发布请求给拥有最高优先权的订阅。对有相同优先权的订阅,服务器应在一个循环模式中从队列取出发布请求。当对于一个订阅的保持活动周期超时时,则该订阅应优先,而不管它的实际优先权,以避免订阅超时。 没要求特殊优先权设置的客户端该设置值为 0
响应		
responseHeader	ResponseHeader	通用的响应参数(见 7.27 中的 ResponseHeader 定义)
revisedPublishingInterval	Duration	服务器将使用的实际发布间隔,以毫秒来表述。服务器应尝试为该参数向客户端请求授权,但可将此值向上或向下进行协商以满足它自身的约束
revisedLifetimeCount	Counter	订阅的生命时间应是服务器协商的三次保持活动间隔中最小值
请求		
revisedMaxKeepAliveCount	Counter	实际最大的保持活动计数(见 7.5 中计数器定义)。服务器应该尝试为该参数给客户端请求授权,但可将此值向上或向下协商以满足它自身的约束

5.13.3.3 服务结果

表 86 定义了此服务特定的服务结果。通用的状态码在表 165 中定义。

表 86 调试订阅服务结果码

符号 Id	描述
Bad_SubscriptionIdInvalid	见表 165 中此结果码的描述

5.13.4 SetPublishingMode

5.13.4.1 描述

该服务用于允许在一个或多个订阅上发送通知。

5.13.4.2 参数

表 87 定义了该服务的参数。

表 87 设置发布模式服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(见 7.26 的 RequestHeader 定义)
publishingEnabled	Boolean	有以下值的布尔值参数: TRUE 允许用于订阅的通知报文发布。 FALSE 禁止用于订阅的通知报文发布。 此参数值不会影响监视项的监视模式(monitoring mode)属性值。设置该值为 FALSE 不会继续发送保持活动报文
SubscriptionIds[]	IntegerId	服务器分配的允许或禁止订阅的标识符列表(见 7.13 中的 IntegerId 定义)
响应		
responseHeader	ResponseHeader	通用响应参数(见 7.27 中的 ResponseHeader 定义)
results[]	StatusCode	允许/禁止订阅的状态码列表(见 7.33 中的状态码定义)。列表的大小和顺序与订阅请求参数的大小和顺序相匹配
diagnosticInfos[]	DiagnosticInfo	允许/禁止订阅的诊断信息列表(见 7.8 中 DiagnosticInfo 定义)。列表的大小和顺序与订阅请求参数的大小和顺序相匹配。如果诊断信息在请求头中没有被请求或如果在请求的处理过程中没有诊断信息,那么此列表为空

5.13.4.3 服务结果

表 88 定义了此服务特定的服务结果。通用状态码在表 165 中定义。

表 88　SetPublishingMode 服务结果码

符号 Id	描述
Bad_NothingToDo	见表 165 中的此结果码描述
Bad_TooManyOperations	见表 165 中的此结果码描述

5.13.4.4　状态码

表 89 定义了针对此服务特定的结果参数值。通用的状态码在表 166 中定义。

表 89　SetPublishingMode 操作级结果码

符号 Id	描述
Bad_SubscriptionIdInvalid	见表 165 中此结果码描述

5.13.5　Publish

5.13.5.1　描述

此服务的使用有两个目的。第一，它用于确认针对一个或更多订阅的通知报文的接收。第二，它用于请求服务器返回通知消息或保持活动消息。鉴于发布请求不是直接指向特定的订阅，它们可被任何订阅所使用。5.13.1.2 描述 Publish 服务的使用。

客户端用于发出发布请求的策略可依据客户端和服务器间的网络延时来调整。在很多情况下，客户端可希望在创建一个订阅后以及在收到发布响应后立即发出一个发布请求。

在另一种情况下，尤其是在高时延网络，客户端可能希望通过管道传输(pipeline)发布请求以确保得到来自服务器的循环报告。建立管道牵涉到在收到响应之前针对每个订阅发送多个发布请求。例如：如果网络在客户端与服务器间引入一个 5 s 的延迟，订阅的发布间隔是 1 s，那么客户端将不得不每秒发出发布请求，而不是在发送下个请求前等待收到一个响应。

鉴于期望发布请求在服务器中排队，服务器应该限制活动的发布请求的个数以避免一个极大的数。但是服务器应接受比创建的订阅更多的排队中的发布请求。期望服务器支持每个订阅有多个发布请求。当服务器收到一个超过其自身限制的新发布请求时，则它应从队列中取出最老的发布请求并返回一个带有设置为 Bad_TooManyPublishRequests 结果的响应。如果客户端收到针对发布请求的服务结果，则它在未完成的发布请求从服务器返回之前不应发出另一个发布请求。

客户端能用 maxNotificationsPerPublish 参数来限制传递给 CreateSubscription 服务的发布响应的大小。然而，这可能仍然造成一个对客户端或服务器太大以至无法处理的消息。在这种情形下，客户端将发现安全通道进入缺省状态并需重新建立，或发布响应返回一个错误并调用 Republish 服务，也返回一个错误。只要其中一种情况发生，则客户端将不得不调整消息处理限值或调整订阅与/或监视项的参数。

在 CreateMonitoredItems 服务或上一个 ModifyMonitoredItems 服务的请求首部中返回诊断信息设置是用于监视项，几当在发布响应中发送通知时用作诊断信息设置。

5.13.5.2　参数

表 90 定义了该服务的参数。

表 90　Publish 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(见 7.26 中 RequestHeader 定义)
subscriptionAcknowledgements[]	SubscriptionAcknowledgement	针对一个或多个订阅的确认列表。该列表可包含相同订阅的多个确认(有相同 SubscriptionId 的多个条目)
subscriptionId	IntegerId	服务器为订阅分配的标识符(见 7.13 中 IntegerId 定义)
sequenceNumber	Counter	被确认的序列号(见 7.5 中计数器定义)。服务器可删除重发对垒中带有该序列号的消息
响应		
responseHeader	ResponseHeader	通用响应参数(见 7.27 中 ResponseHeader 定义)
subscriptionId	IntegerId	服务器为通知正在被返回的订阅分配的标识符(见 7.13 中的 IntegerId 定义)。0 值用于指示没有要发送响应而定义的订阅
availableSequence Numbers []	Counter	序列号范围列表,该列表标识未确认的在订阅重传队列中可用的通知消息。在处理请求确认后而准备该列表
moreNotifications	Boolean	有以下值的布尔参数: TRUE　准备发送的通知号不能在一个响应中发送。 FALSE　准备发送的所有通知都包含在响应中
notificationMessage	NotificationMessage	包含通知列表的通知消息。通知消息参数类型在 7.20 中规定
results[]	StatusCode	确认的结果列表(见 7.33 中状态码定义)。列表的大小和顺序与 subscriptionAcknowledgements 请求参数的大小和顺序相匹配
DiagnosticInfos[]	DiagnosticInfo	确认的诊断信息列表(见 7.8 中的 DiagnosticInfo 定义)。列表的大小和顺序与 subscriptionAcknowledgements 请求参数的大小和顺序相匹配。如果诊断信息在请求头中未被请求或如果在请求处理过程中无诊断信息,则该列表为空

5.13.5.3　服务结果

表 91 定义了此服务特定的服务结果。通用状态码在表 165 中定义。

表 91　Publish 服务结果码

符号 Id	描述
Bad_TooManyPublishRequests	服务器已经达到排队的发布请求的最大数
Bad_NoSubscription	此会话无可用的订阅

5.13.5.4　状态码

表 92 定义了此服务特定的 acknowledgeResults 参数值。通用状态码在表 166 中定义。

表 92　Publish 操作级结果码

标识符	描述
Bad_SubscriptionIdInvalid	见表 165 中该结果码描述
Bad_SequenceNumberUnknown	服务器对序列号是未知的

5.13.6　Republish

5.13.6.1　描述

此服务请求订阅从它的重发序列中重新发布一个通知消息。如果服务器在它的重发序列中没有请求的消息，则返回一个错误的响应。

见 5.13.1.2 中该服务行为的详细描述。

5.13.6.2　参数

表 93 定义了该服务的参数。

表 93　Republish 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(见 7.26 中 RequestHeader 定义)
subscriptionId	IntegerId	服务器为要重新发布的订阅分配的标识符(见 7.13 中 IntegerId 定义)
retransmitSequence Number	Counter	要被重新发布的特定通知消息的序列号(见 7.5 中计数器定义)
响应		
responseHeader	ResponseHeader	通用响应参数(见 7.27 中的 ResponseHeader 定义)
notificationMessage	NotificationMessage	请求的通知消息。通知消息参数类型在 7.18 中规定

5.13.6.3　服务结果

表 94 定义该服务特定的服务结果。通用状态码在表 165 中定义。

表 94 Republish 服务结果码

标识符	描述
Bad_SubscriptionIdInvalid	见表 165 中该结果码描述
Bad_MessageNotAvailable	请求的消息不再可用

5.13.7 TransferSubscriptions

5.13.7.1 描述

此服务用于将订阅和它的监视项从一个会话传输到另一个会话。例如:客户端可能需要重开一个会话,然后向该会话传输其订阅。也可以由一个客户端使用该服务通过向其会话传输订阅以接管来自另一个客户端的订阅。

在请求首部中包含的鉴别令牌识别被传输订阅和监视项的会话。服务器应验证会话的客户端是代表相同用户操作的,且潜在新客户端支持订阅需要的行规。如果服务器传输订阅,则返回重新发送可用的通知消息的序列号。客户端该确认列表中的所有消息,且客户端不会请求重新发送给列表。

如果服务器将订阅传递给新的会话,则服务器应发出一个带有状态码 Good_SubscriptionTransferred 的通知消息 StatusChangeNotification。StatusChangeNotification 的通知消息类型在 7.19.4 中定义。

5.13.7.2 参数

表 95 定义了该服务的参数。

表 95 TransferSubscriptions 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(见 7.26 中 RequestHeader 定义)
subscriptionId[ ]	IntegerId	要发送给新客户端的订阅的标识符列表(见 7.13 中 IntegerId 定义)
sendInitialValues	Boolean	有以下值的布尔参数: TRUE 在 TransferSubscriptions 调用之后的第一次发布响应应包含订阅的所有监视项的当前值,该订阅的在监视模式设置成 Reporting。 FALSE 在 TransferSubscriptions 调用之后的第一次发布响应应仅包含从发送上一次发布响应后的变化值。 此参数仅适用监视属性变化使用的监视项
响应		
responseHeader	ResponseHeader	通用响应参数(见 7.27 中的 ResponseHeader 定义)
results [ ]	TransferResult	要传输的订阅结果列表。列表的大小和顺序与订阅请求参数的大小和顺序相匹配
statusCode	StatusCode	要传输的每个订阅的状态码(见 7.33 中的状态码定义)

表 95（续）

名称	类型	描述
availableSequence NumbersRanges []	Counter	标识订阅中重发队列的通知消息的序列号范围列表。如果订阅传输失败，则此参数为空。NumericRange 类型在 7.21 中定义
diagnosticInfos []	DiagnosticInfo	要传递订阅的诊断信息列表（见 7.8 中 DiagnosticInfo 定义）。列表的大小和顺序与 SubscriptionIds 请求参数的大小和顺序相匹配。如果诊断信息在请求首部中未被请求或如果在请求处理过程中无诊断信息，则列表为空

5.13.7.3 服务结果

表 96 定义了该服务特定的服务结果。通用的状态码在表 165 中定义。

表 96 TransferSubscriptions 服务结果码

标识符	描述
Bad_NothingToDo	见表 165 中的此结果描述
Bad_TooManyOperations	见表 165 中的此结果描述
Bad_UserAccessDenied	见表 165 中的此结果描述。 当前会话的客户端不代表与初有订阅的会话相同的用户运行
Bad_InsufficientClientProfile	当前会话的客户端不支持订阅需要的一个或多个行规

5.13.7.4 状态码

表 97 定义了该服务特定的操作级状态码参数值，通用状态码在表 166 中定义。

表 97 TransferSubscriptions 操作级结果码

标识符	描述
Bad_SubscriptionIdInvalid	见表 165 中的此结果码描述

5.13.8 DeleteSubscription

5.13.8.1 描述

该服务是由客户端调用用于删除一个或多个已创建但还没有被传输到另一个客户端的订阅或已经传输到客户端的订阅。

此服务的成功完成会造成所有使用该订阅的监视项被删除。如果这是最后一个分配给客户端来发出请求的订阅，则所有由该客户端排队的发布请求被取出队列，且对每个请求返回一个负响应。

5.13.8.2 参数

表 98 定义了该服务的参数。

表 98 DeleteSubscriptions 服务参数

名称	类型	描述
请求		
requestHeader	RequestHeader	通用请求参数(见 7.26 中 RequestHeader 定义)
subscriptionId[]	IntegerId	服务器分配的订阅的标识符(见 7.13 中的 IntegerId 定义)
响应		
responseHeader	ResponseHeader	通用响应参数(见 7.27 中的 ResponseHeader 定义)
results []	StatusCode	要删除的订阅的状态码列表(见 7.33 中状态码定义)。列表的大小和顺序与 SubscriptionId 请求参数的大小和顺序相匹配
diagnosticInfos []	DiagnosticInfo	要删除的订阅的诊断信息列表(见 7.8 中 DiagnosticInfo 定义)。列表的大小和顺序与 SubscriptionIds 请求参数的大小和顺序相匹配。如果诊断信息在请求首部中未被请求或在请求处理过程中无诊断信息,则该列表为空

5.13.8.3 服务结果

表 99 定义了该服务特定的服务结果。通用状态码在表 165 中有定义。

表 99 DeleteSubscriptions 服务结果码

标识符	描述
Bad_NothingToDo	见表 165 中此结果码描述
Bad_TooManyOperations	见表 165 中此结果码描述

5.13.8.4 状态码

表 100 定义了此服务特定的 results 参数值。通用的状态码在表 166 中定义。

表 100 DeleteSubscriptions 操作级结果码

标识符	描述
Bad_SubscriptionIdInvalid	见表 165 中此结果码描述

6 服务行为

6.1 安全

6.1.1 概述

OPC UA 服务定义了一些机制以满足 IEC/TR 62541 2 中的安全要求。本条描述了 OPC UA 应用应遵循的一些重要的安全相关规程。

6.1.2 获得并安装一个应用实例证书

所有 OPC UA 应用要求应用实例证书，该证书应包含以下信息：

——运行应用的计算机的网络名称或地址；

——管理或拥有应用的组织机构名称；

——应用的名称；

——应用实例的 uri(统一资源标识符)；

——发布证书的证书授权机构名称；

——证书的发布和终止日期；

证书授权机构(CA)发布给应用的公共密钥；

证书授权机构(CA)创建的数字签名。

另外，每个应用实例证书都有一个应该存储在本地且仅能由该应用访问的私有密匙。如果该私有密钥被泄露，管理员应给应用分配一个新的应用实例证书和私有密钥。

证书可能在安装应用程序时自动生成。在这种情况下，分配给证书的私有密钥将用于创建证书签名。这种方式创建的证书被称为自签名证书。

如果负责这个应用的管理员认为自签名证书不满足该机构的安全需要，则管理员可以安装由证书授权机构发布的证书。从证书授权机构申请一个应用实例证书所包含的相关步骤见图 18。

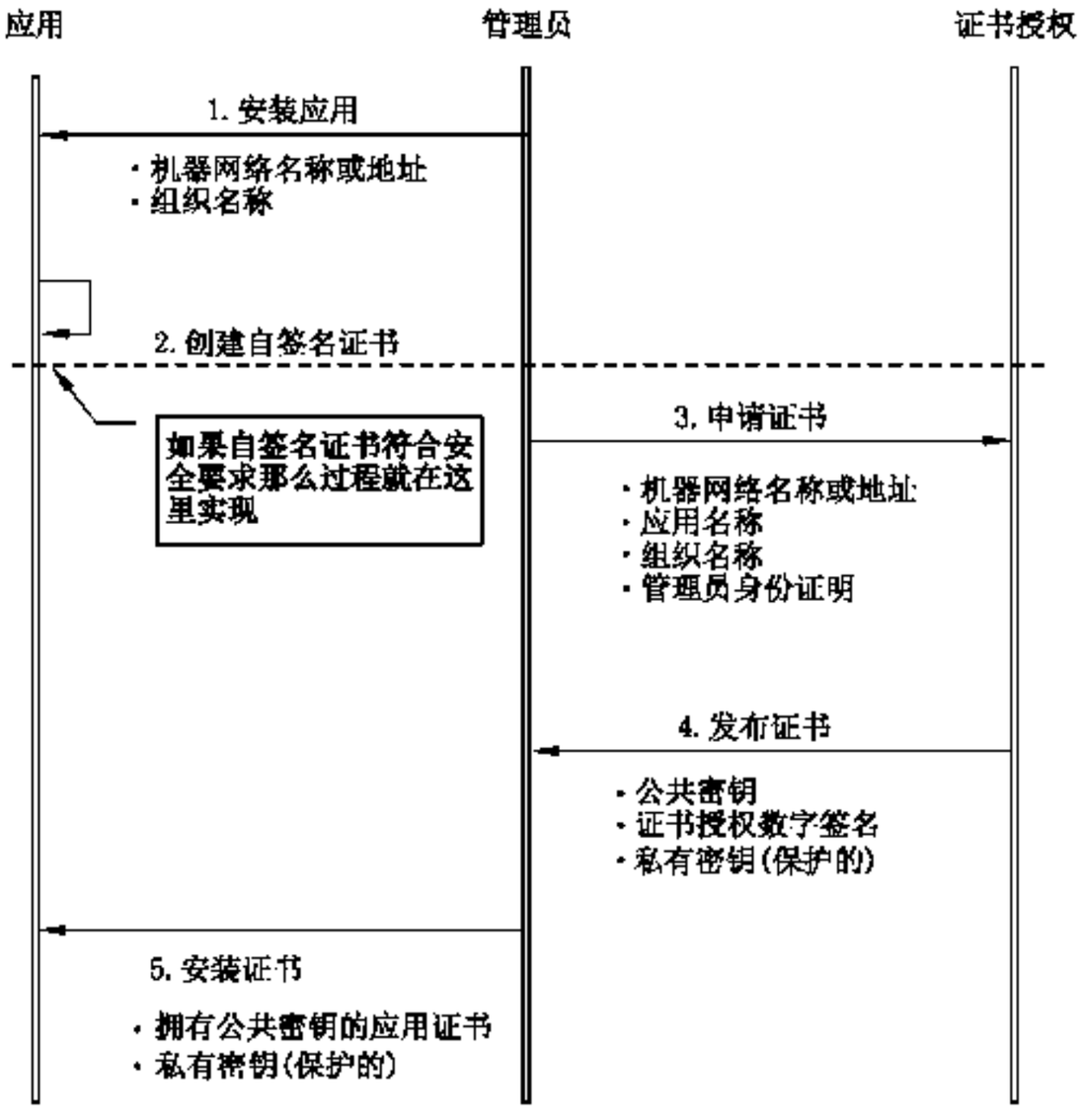

图 18 获取并安装应用实例证书

图 18 阐明了应用、管理员和证书授权机构间的相互关系。应用是安装在单台计算机上的 OPC UA 的应用。管理员是负责管理计算机和 OPC UA 应用的人员。证书授权是为满足使用 OPC UA 应用的组织要求而能发布数字证书的实体。

如果管理员认为自签名证书满足组织安全要求，则可以跳过步骤 3 至步骤 5。应用供应商应总是在安装过程中创建一个缺省的自签名证书。每个 OPC UA 应用应允许管理员用符合自己要求的证书来替换应用实例证书。

当管理员从证书授权机构申请一个新证书时，证书授权机构可能要求管理员提供为拥有该证书的

组织申请证书的授权证明。用于提供该证明的具体机制取决于证书授权机构。

供应商可能选择从授权机构获取证书的过程实现自动化。如果是这样的话，管理员就需要执行图18中的步骤，但应用安装程序将自动完成这些操作，并且仅提示管理员提供正安装的应用实例的信息。

6.1.3 获得并安装软件证书

所有OPC UA应用都可能有一个或多个由授权机构发布的软件证书，并规定应用所支持的行规。这些软件证书包含以下信息：

——产品的供应商名称；

产品的名称；

产品的uri；

软件版本和构造编号；

供应商产品证书；

应用所支持的行规列表；

——每个受支持行规的证书测试状况；

——发布该证书的测试机构的名称；

——证书发布和终止的日期；

证书授权机构发布给应用程序的公开密钥；

——证书授权机构创建的数字签名。

产品供应商负责实现认证过程和从授权机构申请软件证书。当供应商的软件程序在计算机上安装时也应该安装软件证书。分发这些证书时，应用供应商应该做好防范措施，以防止那些未被授权的用户获取软件证书，并将这些证书用于不是由供应商自己开发的应用上。滥用软件证书并不冒安全风险，但是供应商将会由于那些未被授权使用证书的应用所带来的互操作性问题而饱受批评。

供应商会选择给应用程序分配他们认可的证书(称作供应商产品证书)。该应用证书将会与软件证书合并。供应商产品证书的私有密钥将由供应商管理。

从授权机构获取和安装软件证书的步骤描述见图19。

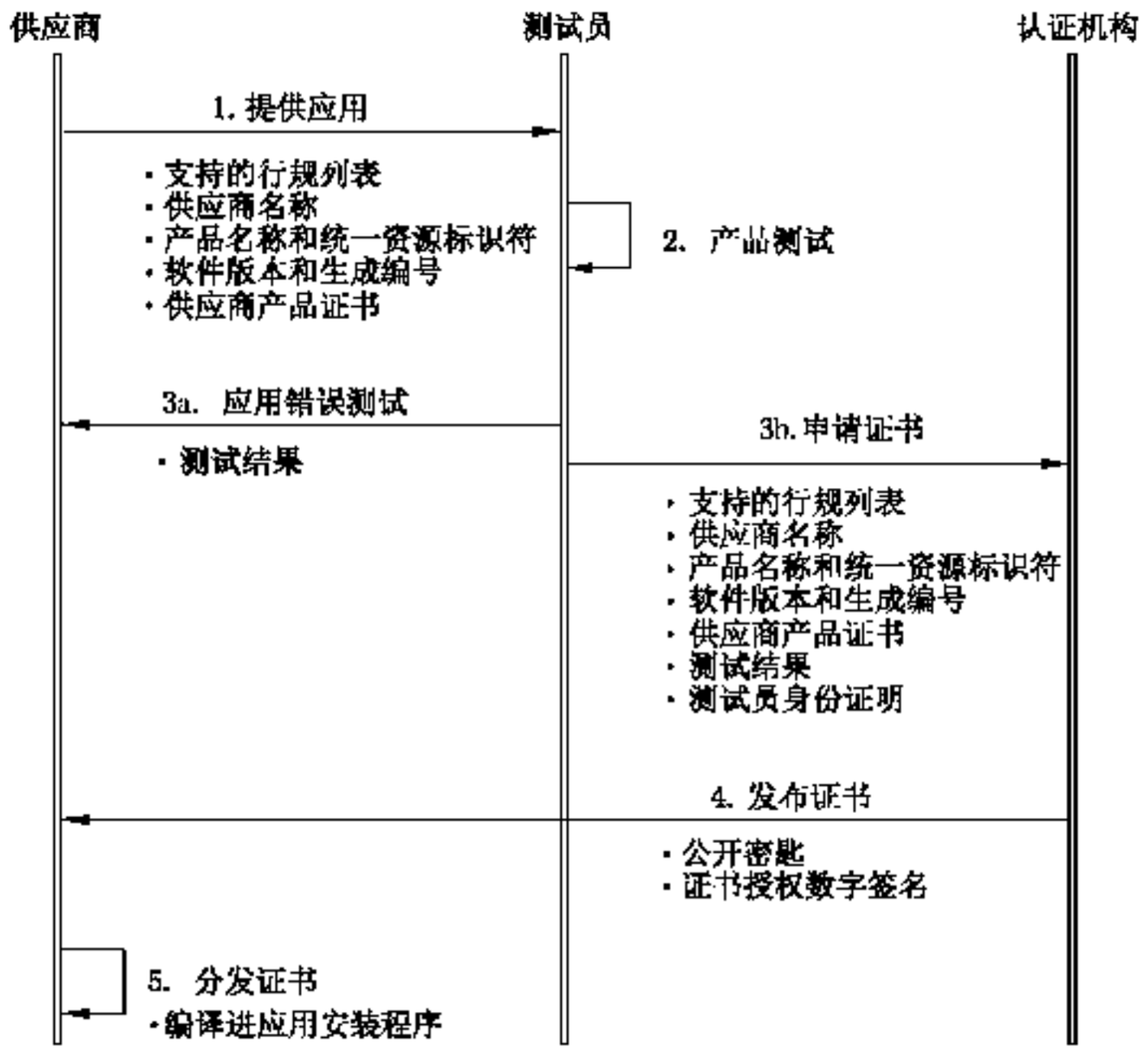

图19 获得和安装软件证书

图 19 阐明了供应商、测试员和认证机构之间的相互作用关系。供应商是对 OPC UA 应用负责的机构。测试员可以是供应商(自认证方式)或者第三方的测试机构。认证机构是由创建 OPC UA 应用程序行规和认证程序的机构所管理的公开密匙结构(PKI)证书授权机构。

认证机构仅对信任的测试对象发布证书。基于这个原因,测试员在获取证书前,应该向认证机构提供身份证明。

6.1.4 判断证书是否可信

应用绝不会与其他不被信任的应用进行通信。一个应用通过检查其他应用的应用实例证书是否可信,来确定该应用是否可信任。应用依靠管理员提供的可信证书列表确定其是否可信。这里有两种分开的列表:可信的应用列表和可信的证书授权机构(CAs)列表。如果一个应用不是直接可信的(即它的证书未列入可信的应用列表),则应用应构建可朔源到可信任 CA 的证书链。

在构建证书链时,链表中的每个证书都应被验证。任何一个校验错误都将导致可信检查失败。一些验证错误是非关键性的,这意味着这些验证可以被有适当权限的用户取消使用。取消的验证错误总是通过审核进行报告(即出现一个对应的审核事件)。

构建可信链需要查找访问链表中的所有证书。这些证书可能存储在本地,也可能由应用的证书提供。如果不能发现 CA 证书或者证书已经存在于可信的证书列表中,则查找过程结束。

表 101 规定了验证证书所执行的步骤,目的是这些步骤可被效仿。除非找到一个可信的证书,那么链表中的每一个证书都将重复执行这些步骤。如果检查失败,那么每个验证步骤具有惟一错误状态和审核事件类型都将被报告。此审核事件是除那些准备给被调用的特殊服务而产生的审核事件以外的事件。消息文本中的服务审核事件包含了 AuditCertifcateEvent 的审核 eventID(更多细节见 6.2)。除非没有关键性错误或者服务已被取消使用,一旦出现错误验证就会立即中止过程。

除非应用实例证书已经被评估和标记为可信的,否则不会被用于客户端或者服务器上。这种情况发生在证书存在于 PKI 信任的链表中或者在离线的方式下被管理员评估后标记为可信的时候。

表 101 证书验证步骤

步骤	错误/审核事件	描述
证书结构	Bad_SecurityChecksFailed AuditCertificateInvalidEventType	证书结构已被验证。 错误可能不会被取消
有效周期	Bad_CertificateTimeInvalid Bad_CertificateIssuerTimeInvalid AuditCertificateExpiredEventType	当前时间应在有效的开始时间和结束时间之间。 错误可以被取消
主机名称	Bad_CertificateHostNameInvalid AuditCertificateDataMismatchEventType	URL 中用于连接服务器的主机名应与证书中规定的主机名相同。 错误可以被取消
统一资源标识符(URI)	Bad_CertificateUriInvalid AuditCertificateDataMismatchEventType	应用和软件证书包含应用或产品 URI 应与证书中提供的应用描述中规定的 URI 相同。 CA 证书的检查可以忽略。 错误不可被取消。 网关服务器 Uri 用于当连接网关服务器时校验应用证书(见 7.1)
证书用法	Bad_CertificateUseNotAllowed Bad_CertificateIssuerUseNotAllowed AuditCertificateMismatchEventType	每个证书都有一套用法。 用法应匹配所申请的对象(比如应用、软件或 CA)。 除非证书显著标明用法受限否则错误可以被取消

表 101（续）

步骤	错误/审核事件	描述
可信列表检查	None	如果证书存在于可信列表中就不需要进一步检查。管理员在将每个证书列入可信列表前应对证书进行验证
查找发布的证书	Bad_CertificateUntrusted AuditCertificateUntrustedEventType	如果发布者证书未知则证书就是不可信的。自签名证书是自己的发行者
签名	Bad_SecurityChecksFailed AuditCertificateInvalidEventType	拥有无效签名的证书会一直被拒绝
查找撤消列表	Bad_CertificateRevocationUnknown Bad_CertificateIssuerRevocationUnknown AuditCertificateRevokedEventType	每个 CA 证书都有撤消列表。如果列表无效则导致检查失败（比如网络中断使得应用访问列表失败）。如果管理员禁止对 CA 发布的证书做撤销检查就不报告错误。错误可以被取消
撤销检查	Bad_CertificateRevoked Bad_CertificateIssuerRevoked AuditCertificateRevokedEventType	证书被撤销且不再使用。 错误不可被取消

证书通常放在称为证书商店的地方。图 20 给出了应用、管理员和证书商店之间的相互关系。证书商店可以在本机或者某些中央服务器上。用于访问证书商店的具体机制取决于管理员为应用和 PKI 设置的环境。

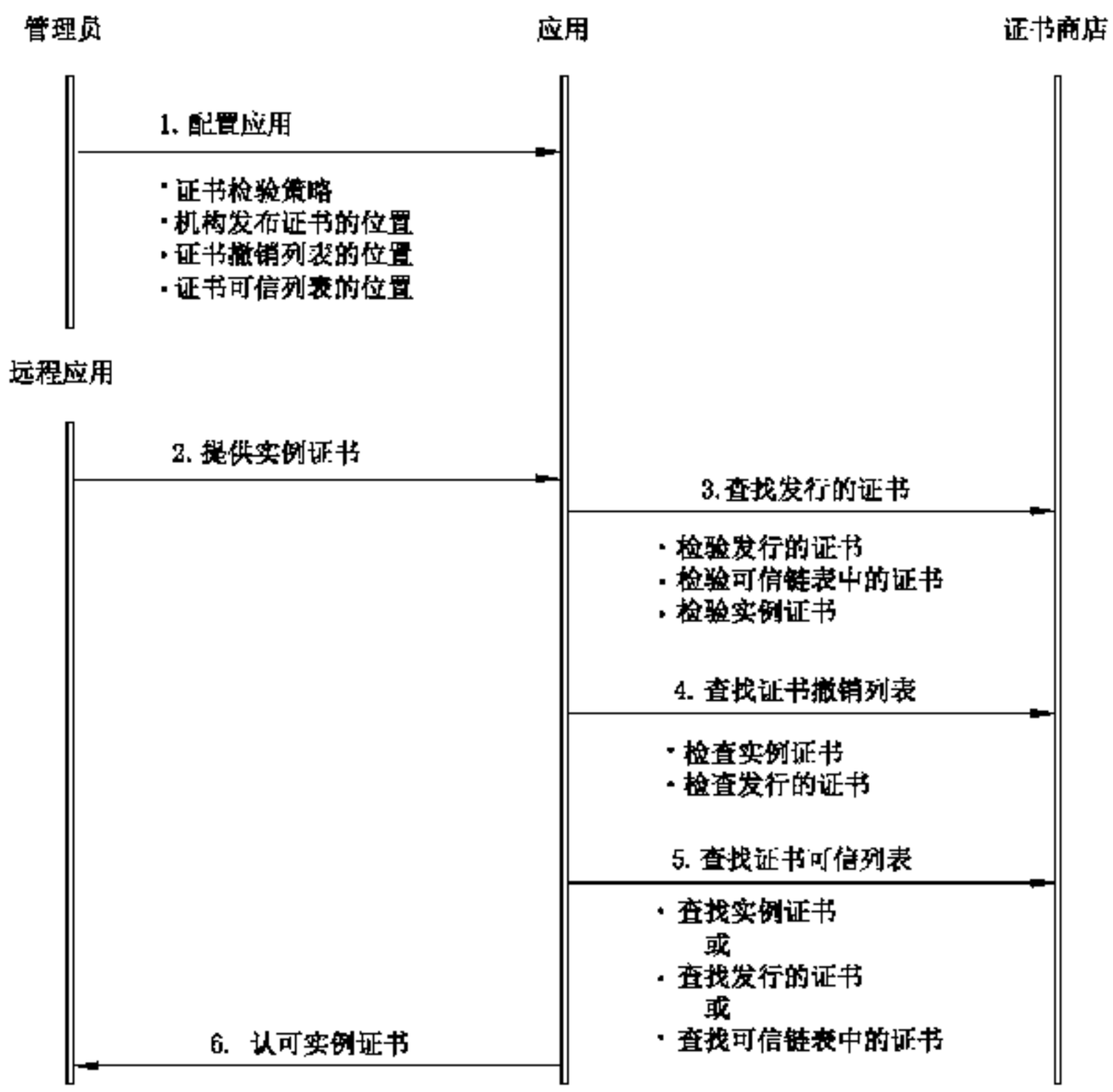

图 20　判断应用实例证书是否可信

6.1.5 验证软件证书

OPC UA 应用程序应验证与之通信的应用所提供的软件证书。

以下情况的软件证书是有效的：

——软件证书上的有效签名；

——软件证书正处于有效期内；

——软件证书未被发行者撤销；

——发行的证书有效并且证书未被发行的 CA 撤销。

用于验证软件证书有效性的步骤与 6.1.4 中描述的应用实例证书基本相同，惟一区别是软件证书不检查应用所依赖的可信列表。

软件证书总会包含一个应用证书，该应用证书由分配该应用的供应商所拥有。存在于证书链表的应用证书可被用于发布应用实例证书。基于这个原因，如果应用证书不是应用实例证书所对应的证书链表的一部分，那么 OPC UA 应用程序可能会拒绝这些应用提供的软件证书。OPC UA 应用会允许管理员要求该行为。在 IEC 62541-7 中定义的行规可详细说明预期的证书处理行为。

6.1.6 创建安全通道

所有 OPC UA 应用在创建会话前都会建立安全通道。安全通道要求应用双方已访问证书，且该证书用于对消息交换进行加密和签名。6.1.2 中描述的按以下过程安装的应用实例证书就用于此目的。

建立安全通道的相关步骤描述见图 21。

图 21 建立安全通道

图 21 假定客户端和服务器都在线访问证书授权机构(CA)。如果在线访问不可用并且管理员在本机上安装了证书授权公匙，那么客户端和服务器都将使用这个公匙进行验证。上图只给出一个 CA，但

是,并没有规定客户端和服务器的证书必须由同一个授权机构发布。自签名应用实例证书不需要 CA 检验。任何一个未被管理员列入可信列表的证书都将被遗弃。

客户端和服务器都有一个被配置为可信的证书列表(有时称为证书可信表或称为 CTL)。这些可信证书可能是授权给证书认证机构的证书或者是 OPC UA 应用实例的证书。OPC UA 应用会被配置成拒绝与没有可信证书的应用进行连接的模式。

证书可能会泄露,意味着它们不再可信。管理员可以通过从所有应用的可信列表中删除它来表示废除,或者 CA 将它加入证书发布者的证书撤销列表(CRL)中。当在线访问不可用时,管理员可以为每个发布的证书保存一个 CRL 的本地副本。

客户端不需要在每次连接服务器时调用 GetEndpoints 服务。终端的信息很少改变,所以客户端可以在本机缓存起来。如果服务器驳回了 OpenSecureChannel 请求,那客户端可以调用 GetEndpoints 服务来确定服务器配置是否改变。

客户端应该意识到在使用 GetEndpoints 服务时存在的两个安全风险。第一个来自伪装的发现服务器将客户端指向伪装的服务器。这样的话,客户端在调用创建会话服务前应该校验终端描述信息中的服务器证书是否为可信证书。

第二个风险来自第三方在通过以太网向客户端转发的过程中修改了终端描述信息的内容。客户端通过比较 GetEndpoints 服务返回的终端描述列表和响应创建会话返回的列表来保护自己免受安全风险。

关于安全通道中使用安全令牌来签名和加密消息交换的确切机制在 IEC 62541-6 中已做了描述。更新令牌的过程也在 IEC 62541-6 中有详细的介绍。

在许多情况下,用于建立安全通道的证书通常就是应用实例证书。然而,一些通信协议栈不支持专门针对单个应用的证书。相反,它们希望所有通信程序都有一个针对用户或者整个机器都是安全的证书。这样的话,OPC UA 应用程序就需要在创建会话时交换它们的应用实例证书。

6.1.7 创建会话

OPC UA 客户端一旦与服务器建立了安全通道,就能创建 OPC UA 会话。

图 22 描述了建立会话的相关步骤。

图 22 阐明了客户端、服务器、证书授权(CA)和认证服务之间的相互关系。CA 负责发布应用证书。如果客户端和服务器都不能在线访问 CA,那么它们应使用管理员安装在本机的证书授权公匙来验证应用证书。

认证服务是中央数据库中能验证客户端提供的用户令牌的一种服务程序。认证服务同时也告诉服务器,用户所具有的访问权限。认证服务依靠用户身份令牌。可能是证书授权机构,票证许可服务认证协议,WS-Trust 规范服务或者某种专用的数据库。

客户端和服务器通过附加随机数签名证书的方式来验证它们拥有的应用证书。5.6.2 描述了用于创建验证签名的验证机制。同样的,客户端也会通过创建与安全令牌关联的签名来验证自身的某些用户身份类型的令牌。

6.1.8 假冒用户

OPC UA 客户端一旦与服务器建立了会话,那它就能通过调用 ActivateSession 服务来改变与会话有联系的用户身份。

图 23 描述了模仿用户的相关步骤。

图 22　建立会话

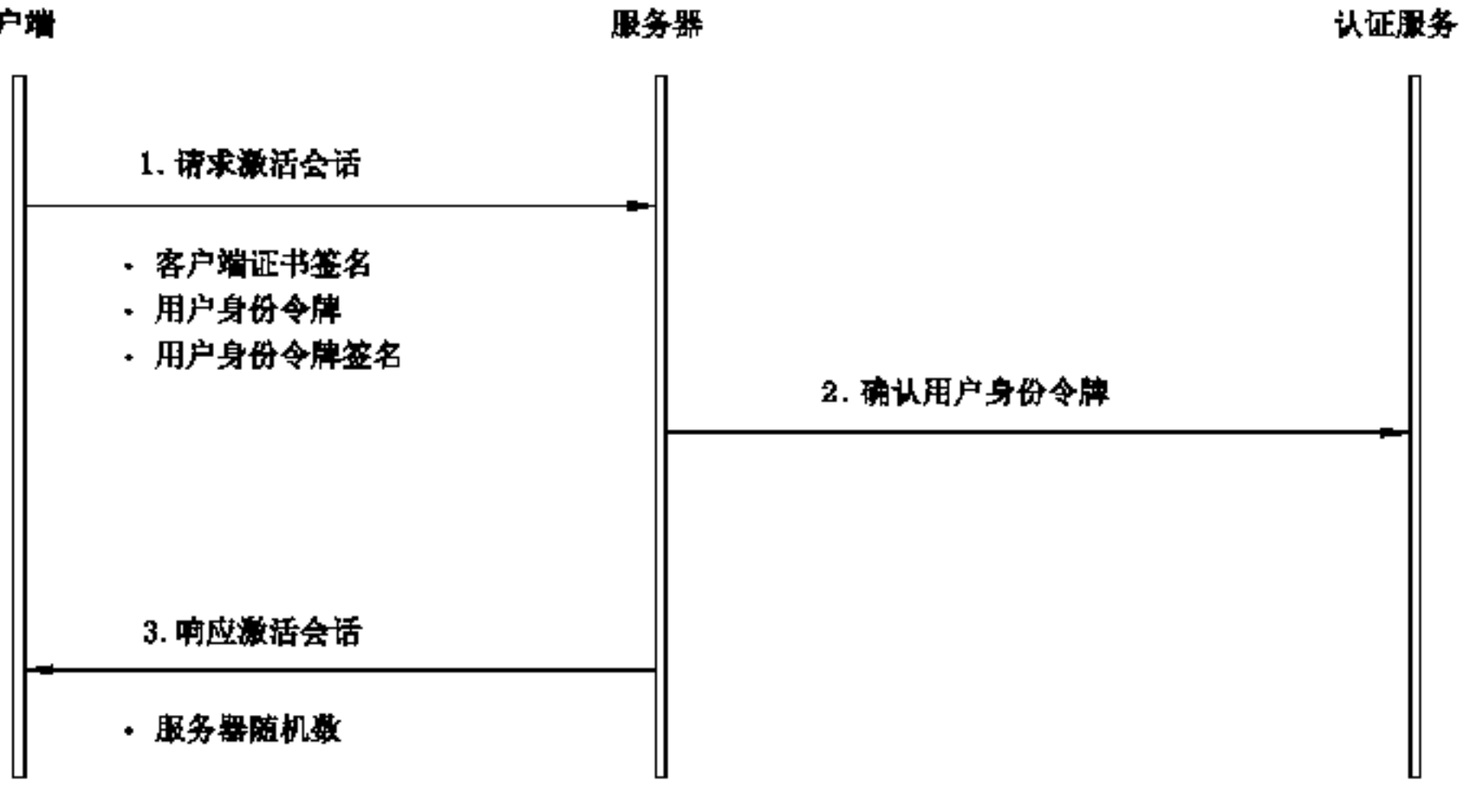

图 23　模仿用户

6.2　审核

6.2.1　概述

在许多系统中审核都是必要的。它提供了跟踪系统正常操作部分的活动和异常行为的手段。从安全的角度讲也是必要的。更多安全方面的审核信息，见 IEC/TR 62541-2。本条描述了 OPC UA 服务器和客户端关于审核的预期目的，以及审核要求的每个服务集的细节。审核可以通过下面一个或多个方式来实现。

a)　生成审核事件的 OPC UA 应用能在日志文件或其他存档路径中记录审核输入项。

b)　生成审核事件的 OPC UA 应用能通过 OPC UA 事件机制发布审核事件。这样就允许一个外部的 OPC UA 客户端向日志文件或其他存档路径中订阅和记录审核输入项。

6.2.2 通用审核日志

每个 OPC UA 服务请求都包含一个用于传递审核记录号的字符串。客户端或任何以客户端方式工作的服务器,比如聚合服务器,都能为它处理的请求创建本地审核日志输入项。这个参数允许客户端在发送请求的同时将输入项的标识符传递过去。如果服务器也保存着一个审核日志,就应该在它写入的审核日志中加入这个标识符。当日志被校验并且发现了这个入口,校验者就能直接关联到由客户端创建的审核日志输入项。这种能力允许在系统中追溯审核日志。

6.2.3 生成审核事件

保存着审核日志的服务器会通过事件消息提供审核日志入口。AuditEventType 及其子类型都定义在 IEC 62541-3 中。一个审核事件消息同时也包含审核记录 Id。AuditEventType 及其子类型的细节都定义在 IEC 62541-5 中。一个支持审核,同时也支持对所有服务器审核事件的订阅的聚合服务器才是真正的聚合服务器。来自聚合服务器的组合信息流应该是可用的。

6.2.4 审核发现服务集

发现服务集可以分为两个部分:由 OPC UA 客户端访问的服务和 OPC UA 服务器调用的服务。由 OPC UA 客户端访问的 FindServers 和 GetEndpoints 服务可以为不成功的服务请求提供审核输入项。由 OPC UA 服务器调用的 RegisterServer 服务可以为所有新的注册和不成功的服务请求提供审核输入项。这些审核条目包含服务器 URI,服务器名称,发现 URIs 和 isOnline 状态等等。不应该为注册服务器(RegisterServer)产生审核输入项,因为对注册服务器的调用不会造成注册服务器的改变。

6.2.5 审核安全通道服务集

在此服务集中用于支持审核服务器的所有服务集,都会生成审核输入项,并应该为 OpenSecureChannel 和 CloseSecureChannel 服务的不成功和成功调用生成审核事件。客户端会在实际调用前提供审核输入项,并提供正确的审核记录 ID。OpenSecureChannel 服务会生成关于类型 AuditOpenSecureChannelEventType 或其子类型的审核事件。CloseSecureChannel 服务会生成关于类型 AuditCloseSecureChannelEventType 或其子类型的审核事件。所有这些事件类型都是 AuditChannelEventType 的子类型。源节点、源名称和附加参数的详细赋值见 IEC 62541-5。对于失败的情况,此类型事件消息文本中会包含服务失败原因的描述。这个描述会比返回给客户端的信息更加详细。从安全角度看,客户端只需要知道服务是否失败了,但从审核角度看,就必须知道更多失败的详细情况。在证书验证错误的情况下,描述会包含特定的用于提供报告证书错误的 AuditCertificateEvent 的 AuditEventId。AuditCertificateEvent 会包含详细的证书验证错误信息。附加参数会包含有关请求的详细信息。多数时候这些事件是由下层的通信协议栈生成的,不过它们也会被服务器使用或者由服务器报告。

6.2.6 审核会话服务集

在此服务集中用于支持审核服务器的所有服务,都会生成审核条目,以及为所有成功和不成功的服务调用生成审核事件。这些服务会生成 AuditSessionEventType 或其子类型的审核事件。特别的,依赖调用的不同服务,它们会生成基本的事件类型或者适当的其他子类型。CreateSession 服务会提供 AuditCreateSessionEventType 或其子类型的事件。ActivateSession 服务会提供 AuditActivateSessionType 或其子类型的事件。当 ActivateSession 服务被用于改变用户身份时,服务器就会提供 AuditImpersonateUserEventType 或其子类型的事件。CloseSession 服务会生成关于 AuditSessionEventType 或其子类型的基本的事件类型。源节点、源名称和附加参数的详细赋值见 IEC 62541-5。对于失败的情况,此类型事件消息文本中会包含服务失败原因的描述。附加参数将包含有关请求的更多细节。

当发生证书认证错误时，这个服务集会生成除 AuditSessionEvents 以外的额外的审核事件。

对于支持审核的客户端，访问会话服务集中的服务都会提供所有成功和不成功服务调用的审核输入项。这些审核输入项会在实际服务调用前设置，并允许调用包含正确审核记录 ID 的服务。

6.2.7 审核节点管理服务集

在此服务集中针对支持审核的服务器的所有服务，都会提供审核输入项，并应该为所有成功和不成功的服务调用生成审核事件。这些服务会生成 AuditNodeManagementEventType 或其子类型的审核事件。源节点、源名称和附加参数的详细赋值见 IEC 62541-5。此类型事件失败的消息文本中会包含服务失败原因的描述。附加参数将包含有关请求的更多细节。

对于支持审核的客户端，访问节点管理服务集中的服务都会提供所有成功和不成功服务调用的审核输入项。这些审核输入项会在实际服务调用前设置，并允许调用包含正确审核记录 ID 的服务。

6.2.8 审核属性服务集

在此服务集中用于支持审核的服务器的 Write(写入)或 HistoryUpdate(历史更新服务)，都会生成审核输入项，以及为所有成功和不成功的服务调用生成审核事件。这些服务会生成 AuditUpdateEventType 或其子类型的审核事件。特别的，写入服务会生成 AuditWriteUpdateEventType 或其子类型的审核输入项。历史更新服务会生成 AuditHistoryUpdateEventType 或其子类型的审核条目。AuditHistoryUpdateEvent 的三个子类型定义为 AuditHistoryEventUpdateEventType、AuditHistoryValueUpdateEventType 和 AuditHistoryDeleteUpdateEventType。究竟是何种子类型是由它们执行的任务类型来决定，历史事件更新、历史数据值更新或是历史删除。源节点、源名称和附加参数的详细赋值见 IEC 62541-5。此类型事件失败的消息文本中会包含服务失败原因的描述。附加参数将包含有关请求的更多细节。

读取和历史读取服务会为不成功的服务调用生成审核输入项和审核事件。这些服务会生成 AuditEventType 或其子类型的审核事件。源节点、源名称和附加参数的详细赋值见 IEC 62541-5。这类型的事件消息文本包含服务失败的描述。

对于支持审核的客户端，访问属性服务集中的写入或历史更新服务都会提供所有成功和不成功服务调用生成审核条目。调用这个服务集中的其他服务也会产生审核条目。这些审核条目会在实际服务调用前设置，并允许调用包含正确审核记录 ID 的服务。

6.2.9 审核方法服务集

在此服务集中针对支持审核的服务器的所有服务，都会提供审核输入项，并且如果服务调用修改地址空间、写入值或修改系统的状态(报警确认、批处理或其他系统变化)，应该为所有成功和不成功的服务调用提供审核事件。这些方法调用服务会生成 AuditUpdateMethodEventType 或其子类型的审核事件。没有修改地址空间、写入值或修改系统状态的方法调用可能会生成事件。源节点、源名称和附加参数的详细赋值请参见 IEC 62541-5。

对于支持审核的客户端，假如调用修改了地址空间、写入值或者修改了系统状态(报警确认、批处理或其他系统变化)，访问方法服务集的服务都会为所有成功和不成功服务调用生成审核输入项。调用这个服务集中的其他服务也会提供审核输入项。这些审核输入项会在实际服务调用前设置，并允许调用包含正确审核记录 ID 的服务。

6.2.10 审核视图、查询、监视项和订阅服务集

在这四个服务集中的所有服务，除了订阅服务集中的 *TransferSubscription* 服务以外，都只向客户端提供信息。通常，这些服务都不生成审核输入项或审核事件消息。*TransferSubscription* 服务会为

所有成功和不成功的调用服务提供类型为AuditSessionEventType 或其子类型的审核事件。源节点、源名称和附加参数的详细赋值请参见 IEC 62541-5。此类型事件失败的消息文本中会包含服务失败原因的描述。

对于支持审核的客户端，访问订阅服务集中的 *TransferSubscription* 服务应生成所有成功和不成功服务调用的审核输入项。调用服务集中的其他服务不需要提供审核输入项。这些审核输入项应在实际服务调用前进行设置，并允许调用包含正确审核记录 ID 的服务。

6.3 冗余

6.3.1 冗余概述

OPC UA 中的冗余确保了客户端和服务器的冗余。OPC UA 没有提供冗余，它只规定了通过标准方式来实现冗余的数据结构和服务。

6.3.2 服务器冗余概述

6.3.2.1 简介

服务器冗余可分为两种模式，透明模式和非透明模式。由定义可知，透明冗余是指故障的服务器将其职责切换给其他服务器的这一过程对客户端是透明的：客户端并不关心甚至不知道服务器发生故障了；客户端根本不需要做任何事情去保持数据流通。相反，非透明冗余要求在服务器故障时，客户端应执行一些动作。

冗余的特殊需求在于两方面，一是保持服务器和客户端的信息通过服务器同步，二是控制数据流从一个服务器到其他服务器的故障切换。

6.3.2.2 透明冗余

对透明冗余而言，OPC UA 规定的仅是数据结构，来帮助客户端识别冗余环境中的哪个服务器是可用的、每个服务器的服务等级和哪个服务器目前支持特定会话。应有一个服务器支持给定会话中的所有 OPC UA 通信，并且客户端能识别出它是哪个服务器，允许完全的数据审核追踪。服务器要负责保证服务器间的信息是同步的，以及在发生故障时其他服务器可以接管服务器与关联客户端的会话和订阅地址。故障时会要求传输层重新连接客户端，但在服务器的终端 URL 不应改变。

图 24 描述了一个透明冗余的典型设置。

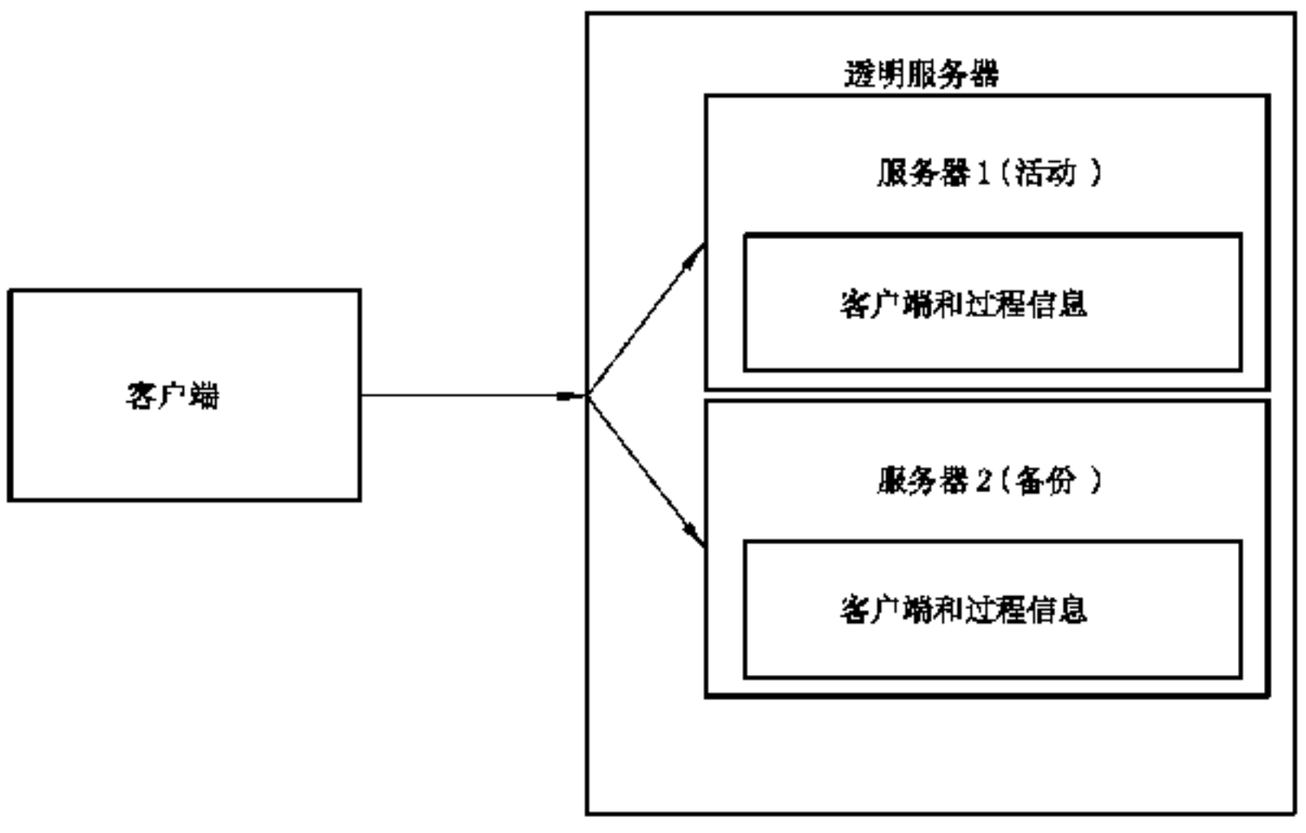

图 24 透明冗余配置

6.3.2.3 非透明冗余

对非透明冗余而言，OPC UA 规定了相同的数据结构和服务器信息，该信息告诉客户端该服务器支持哪些故障模式。这些信息可以让客户端决定应该采取哪些动作来恢复故障。

图 25 描述了一个非透明冗余的典型配置。

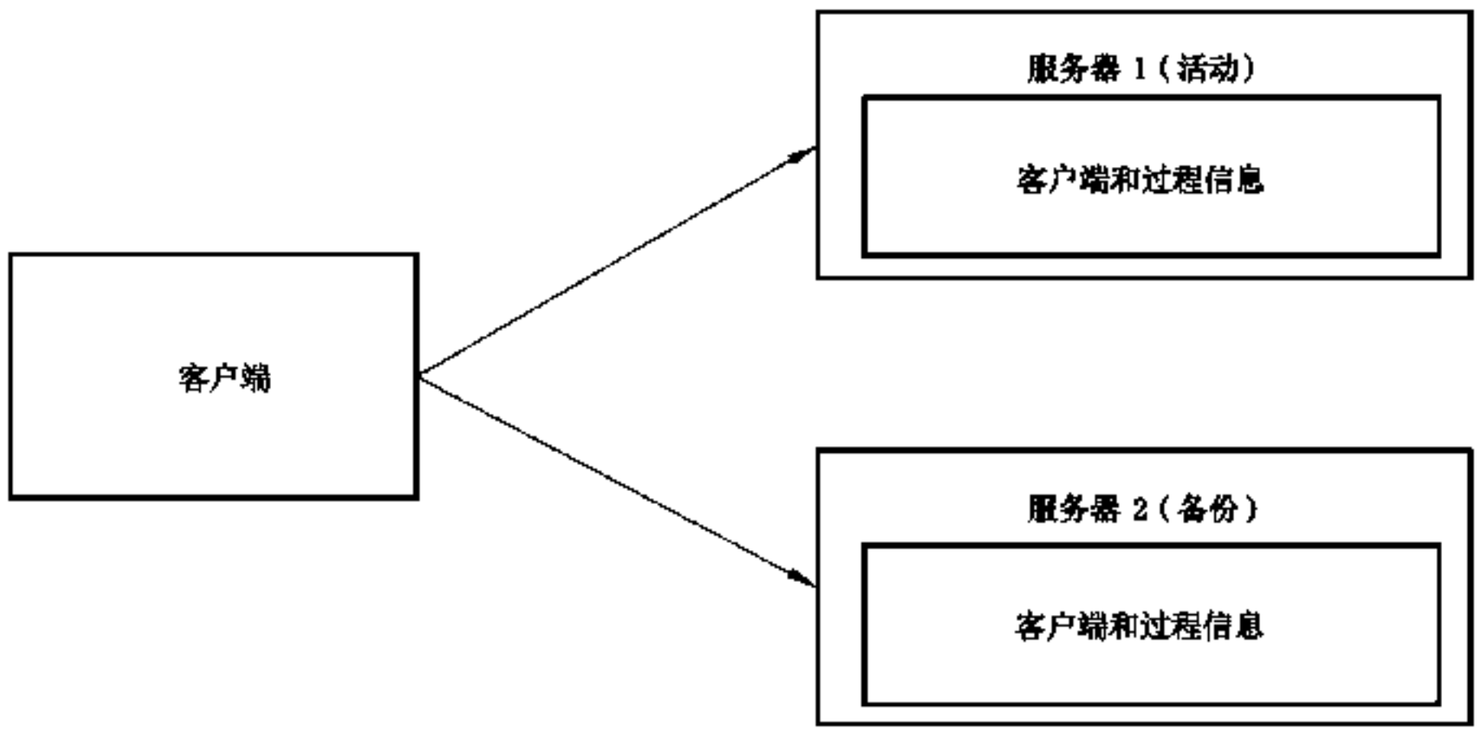

图 25 非透明冗余配置

对非透明冗余，服务器还有冷、暖和热故障的额外概念。冷故障是针对同一时间只有一个活动的服务器。暖故障是针对备份服务器可以处于活动状态，但是不能连接实际数据点的服务器（典型情况是单通道连接的系统）。热故障是针对包含不只一个可以处于活动和完全操作状态的服务器。

表 102 定义了故障行为列表。

表 102 冗余故障行为

故障模式	冷	暖	热
初始化连接			
连接多个 OPC UA 服务器		×	×
创建订阅和添加监视项		×	×
激活订阅取样			×
故障动作			
连接备份 OPC UA 服务器	×		
创建订阅和添加监视项	×		
激活订阅取样	×	×	
激活发布	×	×	×

部分或全部活动会被推进客户端计算机的服务代理程序中，用于减少设计在客户端中功能的数量，更便于客户端利用非透明冗余的好处。通过使用 *TransferSubscription* 服务，允许客户端请求将订阅服务集从一个会话转移到另一个会话，服务供应商能有效地将透明故障生成客户端上的代理存根。有两种方法来实现，一种是要求服务器上的代码支持，另一种是客户端代理程序全权负责。

当使用客户端代理程序时，代理通过向所有服务器发出调用，来简单的复制订阅和修改，但是只能在一个服务器上发布或取样。当代理检测到一个故障时，它能在备份服务器上进行发布和/或取样，就

好像冗余客户端要做到的一样。

其他的方式也要求客户端存根，但在这些方式中存根是个轻量级的程序。在这种模式下，服务器在其他服务器中保持所有订阅的镜像，但在这些订阅中，客户端终端是有效的服务器。当存根检测到有效的服务器发生故障了，它就向备份服务器发布一个 *TransferSubscription* 的命令，从故障服务器持有的会话中将订阅服务转移到自己的会话中，然后激活这个发布。

图 26 描述了客户端代理和服务器代理冗余之间的不同。

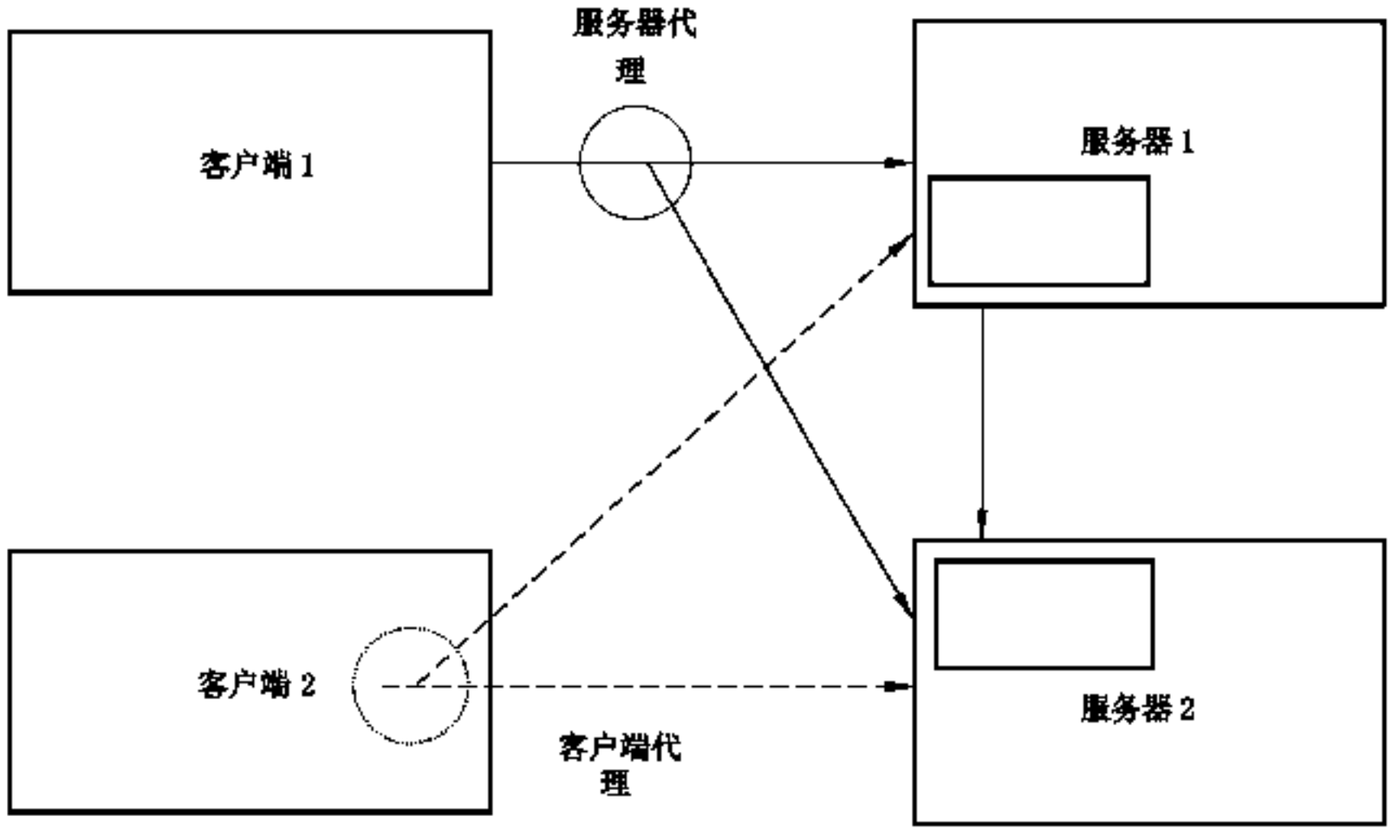

图 26 冗余模式

6.3.3 客户端冗余

OPC UA 中通过调用 *TransferSubscription* 命令和在服务器信息结构中显示客户端信息的方式，来支持客户端冗余。当订阅生命周期与创建它的会话无关后，备份客户端就能监视活动的客户端与服务器的会话，就像它监视其他数据变量一样。如果活动的客户端不再活动了，服务器会向每个有变量监视的客户端发送一个数据更新通知。一旦收到这个通知，备份客户端就会通知服务器将订阅传输到它自己的会话中。如果订阅处理的足够仔细，同时有足够的资源缓存交换的数据，那么客户端故障的时候就能保证数据不丢失。

OPC UA 没有提供标准的机制用于在活动和备份的客户端之间传输 SessionId 和 SubscriptionIds，但只要备份客户端知道活动客户端的名称，这个信息就可以利用 SubscriptionDiagnostics 和 ServerDiagnostics 的 SessionDiagnostics 部分，很容易地得到。

7 通用参数类型定义

7.1 ApplicationDescription(应用描述)

该参数的组成见表 103。

表 103 ApplicationDescription

名称	类型	描述
ApplicationDescription	structure	指定一个可用的应用
ApplicationUri	String	应用实例的全球惟一标识符

表 103（续）

名称	类型	描述
productUri	String	产品的全球惟一标识符
applicationName	LocalizedText	应用的本地描述名
applicationType	Enum ApplicationType	应用的类型。 该值为以下值构成的枚举之一： SERVER_0 应用是服务器 CLIENT_1 应用是客户端 CLIENTANDSERVER_2 应用是客户端和服务器 DISCOVERYSERVER_3 应用是发现服务器
gatewayServerUri	String	标识与 discoveryUrls 有关的网关服务器的 URI。 如果服务器可以直接访问就不需要指定该值。 如果 applicationType 是 CLIENT_1 就不使用该字段
discoveryProfileUri	String	标识被 URLs 支持的发现行规的 URI。 如果 applicationType 是 CLIENT_1 就不使用该字段。 如果没有指定该值，那么终端就应支持 5.4 中定义的发现服务。 交替发现行规定义在 IEC 62541-7 小节中
discoveryUrls[]	String	应用提供的发现终端 URL 列表。 如果 applicationType 是 CLIENT_1，则该字段应是空列表

7.2 ApplicationInstanceCertificate(应用实例证书)

ApplicationInstanceCertificate 是包含编码证书的 ByteString。ApplicationInstanceCertificate 的编码技术取决于安全技术映射，并在 IEC 62541-6 中有完整的定义。表 104 描述了应包含在 ApplicationInstanceCertificate 中的信息。

表 104 ApplicationInstanceCertificate

名称	类型	描述
ApplicationInstanceCertificate	structure	由证书授权机构创建了签名的 ApplicationInstanceCertificate
Version	String	标识证书编码的版本的标识符
serialNumber	ByteString	发布者分配的证书惟一标识符
signatureAlgorithm	String	用于签署证书的算法。 该字段的语法取决于证书编码技术
Signature	ByteString	发布者创建的签名
Issuer	structure	标识创建签名的发布者证书的名称
validFrom	UtcTime	证书开始有效的时间
validTo	UtcTime	证书开始无效的时间
Subject	structure	标识证书所描述的应用实例的名称。 包含产品名称和负责应用实例的机构的名称

表 104（续）

名称	类型	描述
applicationUri	String	由 ApplicationDescription 指定的 applicationUri。 ApplicationDescription 参见 7.1
Hostnames[]	String	运行应用实例的机器名称。 如果主机可访问多个网络那么它可能会有多个名称。 主机名可以是数值化的网络地址或者描述名称。 服务器证书至少应该定义了一个主机名
publicKey	ByteString	与证书关联的公共密钥
keyUsage[]	String	指定证书密匙的用法。 ApplicationInstanceCertificate 应支持数字签名、防抵赖密匙加密、数据加密和客户端/服务器授权。 该字段的内容取决于证书编码技术

7.3 BrowseResult(浏览结果)

该参数的组成见表 105。

表 105 BrowseResult

名称	类型	描述
BrowseResult	structure	浏览操作的结果
statusCode	StatusCode	浏览描述的状态。 如果还指向浏览描述返回的引用，就应该置为 Good_MoreReferences-Exist
continuationPoint	ContinuationPoint	服务器定义来标识延长点的不透明值。 ContinuationPoint 类型在 7.6 中定义
References[]	ReferenceDescription	符合浏览描述中指定准则的引用集。 如果没有符合的准则为空。 引用描述类型在 7.24 中定义

7.4 ContentFilter(内容过滤器)

7.4.1 ContentFilter 结构

内容过滤器结构定义了一个由过滤标准组成的元素集合。集合中的每个元素都描述了一个运算符和用于运算符的一组运算对象。表 110 描述了内容过滤器中可能使用的运算符。通过评价以运算对象组中的第一个运算对象为开始的元素组的第一项来评价过滤器。每个元素的运算对象都可能包含了元素的引用，这就需要继续计算那些被引用的元素。如果它不能追溯到开头那个被略过的元素，那对每个运算符的其他操作对象就会导致一个错误。附录 B 提供了使用内容过滤器结构的范例。

表 106 定义内容过滤器的结构。

表 106　ContentFilter 结构

名称	类型	描述
ContentFilter	Structure	
Elements[]	ContentFilterElement	组成过滤器条件的运算符和运算对象列表。过滤器从第一个输入项开始进行评价
filterOperator	enum FilterOperator	待评价的过滤器运算符。 过滤器运算符枚举在表 110 中定义
filterOperands[]	Extensible Parameter FilterOperand	选中的运算符使用的运算对象。表数量和用法取决于表 110 定义的运算对象。该数组至少要有一个输入项。 该可扩展参数类型是 7.4.4 中规定的 FilterOperand 参数类型。它规定了有效 FilterOperand 值的列表

7.4.2　ContentFilterResult(内容过滤器结果)

该数据类型的组成见表 107。

表 107　ContentFilterResult 结构

名称	类型	描述
ContentFilterResult	structure	包含过滤器有关的每个错误信息的结构
elementResults	ContentFilterElementResult	过滤器中单个元素的结果列表。列表的大小和顺序匹配 *ContentFilter* 参数中元素的大小和顺序
statusCode	StatusCode	单个元素的状态码
operandStatusCodes[]	StatusCode	一个元素中运算对象的状态码列表。列表的大小和顺序与 ContentFilterElement 中运算对象的大小和顺序相匹配。如果没发生运算对象错误,列表就为空
operandDiagnosticInfos[]	DiagnosticInfo	一个元素中运算对象的诊断信息列表。列表的大小和顺序与 ContentFilterElement 中运算对象的大小和顺序相匹配。如果在请求报文中没有请求诊断信息,或者处理运算对象过程中没有诊断信息,则列表为空
elementDiagnosticInfos[]	DiagnosticInfo	过滤器中单个元素的诊断信息列表。列表的大小和顺序匹配过滤器请求参数中元素的大小和顺序。如果在请求报文中没有请求诊断信息或者处理过程中没有诊断信息,则列表为空

表 108 定义了该结构特定的状态码参数值。通用状态码的定义见表 166。

表 108　ContentFilterResult 返回代码

符号 Id	描述
Bad_FilterOperandCountMismatch	提供给过滤器运算符的运算对象数量小于期望的数量
Bad_FilterOperatorInvalid	过滤器中提供了不识别的运算符
Bad_FilterOperatorUnsupported	运算符有效,但是服务器不支持这个过滤器运算符

表 109 定义了该结构特定的 operandStatusCode 参数的值。通用状态码的定义见表 166。

表 109 ContentFilterResult 运算对象返回码

符号 Id	描述
Bad_FilterOperandInvalid	此结果码的描述见表 166
Bad_FilterElementInvalid	被引用的元素在内容过滤器中不是一个有效元素
Bad_FilterLiteralInvalid	被引用的文字不是一个有效的基本数据类型
Bad_AttributeIdInvalid	属性 Id 不是系统中的有效属性 Id
Bad_IndexRangeInvalid	此返回代码的描述见表 166
Bad_NodeIdInvalid	此返回代码的描述见表 166
Bad_NodeIdUnknown	此返回代码的描述见表 166
Bad_NotTypeDefinition	提供的节点 Id 不是一个类型定义节点 Id

7.4.3 FilterOperator(过滤器运算符)

表 110 定义了可用在 ContentFilter 中的基本运算符。高级运算符的描述见表 111。7.4.4 定义了运算对象。

表 110 基本 FilterOperator 定义

运算符	运算对象数量	描述
Equals 0	2	如果运算对象的[0]等于[1]则为 TRUE。 如果运算对象为不同数据类型,那么系统内部会进行通用类型的转换。 如果运算对象是不同类型并且内部无法进行自动转换,那这个操作结果就为 FALSE。关于如何转换不同类型运算对象的更多信息请参见表 114 中关于数据类型优先级的讨论
IsNull_1	1	如果运算对象的[0]是 NULL 则为 TRUE
GreaterThan_2	2	如果运算对象的[0]大于[1]则为 TRUE。 以下是对运算对象的约束规则: [0]: 能分解成有序值的运算对象。 [1]: 能分解成有序值的运算对象。 转换规则的定义同 Equals
LessThan 3	2	如果运算对象[0]小于运算对象[1]则为 TRUE。 转换规则和约束的定义同 GreaterThan
GreaterThanOrEqual_4	2	如果运算对象[0]大于或等于运算对象[1]则为 TRUE。 转换规则和约束的定义同 GreaterThan
LessThanOrEqual_5	2	如果运算对象[0]小于或等于运算对象[1]则为 TRUE。 转换规则和约束的定义同 GreaterThan

表 110（续）

运算符	运算对象数量	描述
Like_6	2	如果运算对象[0]与运算对象[1]的模式结构匹配则为 TRUE。模式语法定义见表 112。 以下重构方法用于运算对象： [0]： 能转换成字符串的运算对象。 [1]： 能转换成字符串的运算对象。 False，如果运算对象不能转换成字符串
Not_7	1	如果运算对象的[0]是 False 则为 FALSE。 以下约束方法用于运算对象： [0]： 能转换成布尔型的运算对象。 如果运算对象不能转换成布尔型值，那么结果就为 NULL。参见下面关于 NULL 处理的讨论
Between_8	3	如果运算对象的[0]大于或等于[1]并且小于或等于[2]则为 TRUE。 以下约束方法用于运算对象： [0]： 能分解成有序值的运算对象。 [1]： 能分解成有序值的运算对象。 [2]： 能分解成有序值的运算对象。 如果运算对象类型都不相同，那么系统内部会进行通用类型的转换。如果运算对象是不同类型并且内部无法进行自动转换，那这个操作结果就为 FALSE。关于如何转换不同类型运算对象的更多信息请参见表 114 中关于数据类型优先级的讨论
InList_9	2..n	如果运算对象[0]等于剩余的一个或多个对象则为 TRUE。Equals 运算符用于比较运算对象[0]和列表中剩余的每个运算对象。如果每个 Equals 比较结果都为 True，则 InList 返回 True
And_10	2	如果运算对象的[0]和[1]都是 True 则为 TRUE。 以下约束方法用于运算对象： [0]： 能转换成布尔型的运算对象。 [1]： 能转换成布尔型的运算对象。 如果任一个运算对象不能转换成布尔型值，那么就视为 NULL。参见下面关于 NULL 处理的讨论
Or_11	2	如果运算对象[0]或者[1]是 True 则为 TRUE。 以下约束方法用于运算对象： [0]： 能转换成布尔型的运算对象。 [1]： 能转换成布尔型的运算对象。 如果任一个运算对象不能转换成布尔型值，那么就视为 NULL。参见下面关于 NULL 处理的讨论

表 110（续）

运算符	运算对象数量	描述
Cast_12	2	将运算对象[0]转换成具有节点 Id 标识数据类型的值，标识为运算对象[1]。 以下约束方法用于运算对象： [0]： 任一对象。 [1]： 能转换成 NodeId 或 ExpandedNodeId 的运算对象，其中节点类型为 *NodeClass*。 如果任一参数或转换出现错误，那么 Cast 结果就视为 NULL。参见下面关于 NULL 处理的讨论
BitwiseAnd_16	2	运算结果是与最大运算对象相同大小的整数，它包含了将两个对象转换成两者最大长度后的按位与操作。 以下约束方法用于运算对象： [0]： 能转换成整型的运算对象。 [1]： 能转换成整型的运算对象。 如果任一运算对象不能转换成整型，那么就视为 NULL。参加下面关于 NULL 处理的讨论
BitwiseOr_17	2	运算结果是与最大运算对象相同大小的整数，它包含了将两个对象转换成两者最大长度后的按位或操作。 以下约束方法用于运算对象： [0]： 能转换成整型的运算对象。 [1]： 能转换成整型的运算对象。 如果任一运算对象不能转换成整型，那么就视为 NULL。参加下面关于 NULL 处理的讨论

许多运算对象的类型都有约束限制。这就要求运算对象的类型是可确定的。有时候，运算对象指定了特定类型（比如，Literal Operand）。其他时候，这个类型要求属性的值是可读的。ElementOperands 是布尔型值，除非运算符是 Cast 或者嵌套 *RelatedTo* 运算符。

表 111 定义了复杂的且要求计算目标节点的运算符。这些运算符应对结果集中每个可能的目标节点进行重新计算。

表 111　复杂 FilterOperator 定义

运算符	运算对象数量	描述
InView_13	1	如果目标节点包含在运算对象[0]定义的视图中则为 TRUE。 以下约束规则用于运算对象： [0]：能转换成视图节点的 NodeId 的运算对象。 如果不能转换成视图节点的节点 Id，那么结果就为 False

表 111（续）

运算符	运算对象数量	描述
OfType_14	1	如果目标节点是运算对象[0]的类型或者子类型则为 TRUE。 以下约束规则用于运算对象： [0]：能转换成标识对象类型或变量类型节点的 NodeId 的运算对象。 如果运算对象[0]不能转换成标识对象类型或者变量类型节点的节点 Id，那么结果就为 FALSE
RelatedTo 15	6	如果目标节点是运算对象[0]的类型，同时与 NodeId 相关（该 NodeId 的类型在运算对象[1]定义，引用类型在运算对象[2]定义），则为 TRUE。 运算对象[0]或运算对象[1]可以指向涉及其他 relatedTo 运算符成员的引用。这里允许关系链表（比如，A 与 B 相关，B 与 C 相关）。这时，引用成员返回 NodeId 列表而不是 TRUE 或 FALSE。如果出现任何错误或者任一运算对象不能转换成适当的值，那么链表关系结果就是一个空节点列表。 运算对象[3]定义了可以级联的数量。如果运算对象[3]等于 1，那么对象被直接相关。如果它大于 1，那么类型为运算对象[1]所述的 NodeId 将进行级联数指定深度的校验。这时，中间的节点类型是不确定的，同时只有用于到达最末节点的引用类型是确定的。如果要求的深度不能实现，那返回 FALSE，比如，空节点列表。如果运算对象[3]等于 0，那么该关系前向检索到逻辑末端，并且检查各节点是否为运算对象[1]指定的类型。如果每个节点都满足这个标准，那返回 TRUE，比如，NodeId 会包含在子列表中。 运算对象[4]定义了在运算对象[0]和[1]中是否包含由它们定义的子类型。TRUE 表示支持子类型。 运算对象[5]定义了运算对象[3]中是否包含由它定义的引用子类型。TRUE 表示支持子类型。 以下约束规则用于运算对象： [0]：能转换成 NodeId 或 ExpandedNodeId 的运算对象，该 NodeId 或 ExpandedNodeId 标识对象类型、变量类型节点或对 RelatedTo 运算符引用。 [1]：能转换成 NodeId 或 ExpandedNodeId 的运算对象，该 NodeId 或 ExpandedNodeId 标识了对象类型、变量类型节点或对 RelatedTo 运算符的引用。 [2]：能转换成标识引用类型节点的 NodeId 的运算对象。 [3]：能隐式转换成 32 位整型值的运算对象。 [4]：能隐式转换成布尔型值的运算对象。如果不能成功转换，那值就为 FALSE。 [5]：能隐式转换成布尔型值的运算对象。如果不能成功转换，那值就为 FALSE。 如果运算对象[0]、[1]、[2]、[3]都没能成功转换成指定类型值，那么该操作结果应为 FALSE（当设为内嵌 relatedTo 运算对象时为空）。 RelatedTo 的示例见 7.4.4

通过将运算对象[2]设为 HasSubtype 引用类型并将运算对象[3]设为 0，RelatedTo 运算符可用于标识运算对象[1]设的指定类型是否为运算对象[0]设的另一个类型的子类型。

Like 运算符可用于通配符比较。特殊的几个字符可以包含在 Like 运算符的第 2 个运算对象中。有效字符定义在表 112 中。通配符可以组合在单个字符串中（比如，"Th[ia_[ts]%"可以匹配"That is

fine"、"This is fine"、"That as one"、"This it is"、"Then at any"等等)。

表 112 通配符

特殊字符	描述
%	匹配任意零字符串或多个字符(比如,"main%"可以匹配任意以"main"开头的字符串、"%en%"可以匹配任意包含字母'en'的字符串,像"entail"、"green"、"content")
_	匹配任意单个字符(比如,"_ould"可以匹配"would"、"could")
\	转义字符用于字面的解释(比如,\\is\、\%is%、_is_)
[ ]	匹配列表中的任意单个字符(比如,"abc[13 68]"可以匹配"abc1"、"abc3"、"abc4"、"abc5"、"abc6"、"abc8"。"xyz[c-f]"可以匹配"xyzc"、"xyzd"、"xyze"、"xyzf")
[^]	不匹配列表中的任意单个字符。^应该是[ ]中的第一个字符。(比如,"ABC[^13 5]"不匹配"ABC1"、"ABC3"、"ABC4"、"ABC5"、"xyz[^dgh]"不匹配"xyzd"、"xyzg"、"xyzh")

表 113 定义了运算对象值的转换规则。如果存在隐式转换(I),那类型将会自动转换。如果存在显式转换(E),那类型将会使用 cast 运算符进行转换。如果没有转换可能(X),那对象就不可转换,然而,有些服务器支持运用特殊的显式转换。表中使用的类型定义在 IEC 62541-3 中。不在表中的数据类型就不存在任何定义了的转换。

表 113 转换规则

源类型(From)	目标类型(To)																			
	Boolean	Byte	ByteString	DateTime	Double	ExpandedNodeId	Float	Guid	Int16	Int32	Int64	NodeId	SByte	StatusCode	String	LocalizedText	QualifiedName	UInt16	UInt32	UInt64	XmlElement
Boolean	—	I	X	X	I	X	I	X	I	I	I	X	I	X	E	X	X	I	I	I	X
Byte	E		X	X	I	X	I	X	I	I	I	X	I	X	E	X	X	I	I	I	X
ByteString	X	X		X	X	X	X	E	X	X	X	X	X	X	X	X	X	X	X	X	X
DateTime	X	X	X	—	X	X	X	X	X	X	X	X	X	X	E	X	X	X	X	X	X
Double	E	E	X	X	—	X	E	X	E	E	E	X	E	X	E	X	X	E	E	E	X
ExpandedNodeId	X	X	X	X	X		X	X	X	X	X	E	X	X	I	X	X	X	X	X	X
Float	E	E	X	X	I	X		X	E	E	E	X	E	X	E	X	X	E	E	E	X
Guid	X	X	E	X	X	X	X		X	X	X	X	X	X	E	X	X	X	X	X	X
Int16	E	E	X	X	I	X	I	X	—	I	I	X	E	X	E	X	X	E	I	I	X
Int32	E	E	X	X	I	X	I	X	E	—	I	X	E	E	E	X	X	E	E	I	X
Int64	E	E	X	X	I	X	I	X	E	E	—	X	E	E	E	X	X	E	E	E	X

表 113（续）

源类型(From)	目标类型(To)																			
	Boolean	Byte	ByteString	DateTime	Double	ExpandedNodeId	Float	Guid	Int16	Int32	Int64	NodeId	SByte	StatusCode	String	LocalizedText	QualifiedName	UInt16	UInt32	UInt64	XmlElement
NodeId	X	X	X	X	X	I	X	X	X	X	X	—	X	X	I	X	X	X	X	X	X
SByte	E	E	X	X	I	X	I	X	I	I	I	X		X	E	X	X	I	I	I	X
StatusCode	X	X	X	X	X	X	X	X	X	I	I	X	X	—	X	X	X	E	I	I	X
String	I	I	X	E	I	E	I	I	I	I	I	E	I	X	—	E	E	I	I	I	X
LocalizedText	X	X	X	X	X	X	X	X	X	X	X	X	X	X	I	—	X	X	X	X	X
QualifiedName	X	X	X	X	X	X	X	X	X	X	X	X	X	X	I	I		X	X	X	X
UInt16	E	E	X	X	I	X	I	X	I	I	I	X	E	I	E	X	X		I	I	X
UInt32	E	E	X	X	I	X	I	X	E	I	I	X	E	E	E	X	X	E	—	I	X
UInt64	E	E	X	X	I	X	I	X	E	E	I	X	E	E	E	X	X	E	E	—	X
XmlElement	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	—

通过每个成员的转换可以将源类型的数组转换成目标类型的数组。任一转换错误都将导致整个转换失败。

长度为 1 的数组可隐式转换成相同类型的标量值。

利用 IEC 62541-6 定义的语法，Guid、NodeId 和 ExpandedNodeId 都可以转成 String 类型，或从 String 类型转换而来。

Floating 点值要先加上 0.5 然后再截取转换成整型值。

ByteString 通过格式化为有序的十六进制数字来转换成 String。

LocalizedText 通过去掉 Locale 转换为 String。String 通过将 Locale 设为""来转换为 LocalizedText。

QualifiedName 通过去掉 NamespaceIndex 来转换为 String。String 通过将 NamespaceIndex 设为 0 来转换为 QualifiedName。

StatusCode 可以通过拷贝每个位，转换为 UInt32 和 Int32 或从 UInt32 和 Int32 转换而来。当 StatusCode 转换为 UInt16 或者 Int16 或从 UInt16 或者 Int16 转换而来时，仅仅是高 16 位会被拷贝。

Boolean 值为真时被转换为"1"，为假时被转换为"0"。非零数值被转换为 Boolean 值 TRUE。零值被转换为 Boolean 值 FALSE。包含"TRUE"、"FALSE"、"1"、"0"的字符串可以转换为 Boolean 值。其他的字符串转换会引发错误。这时，字符串是大小写敏感的。

当使用不同数据类型的运算对象时，就很有可能使用隐式转换。这种情况下，定义在表 114 中的优先规则用于确定使用哪种隐式转换。列表中的第一个数据（从上往下）类型拥有最高优先级。如果某种数据类型没有在表中定义，那当评价运算时它就不能用于隐式转换。

例如,假定 A = 1.1(Float) and B = 1(Int32),并且这些值使用了 Equal 运算符。这个运算就是将 Int32 型值转换成 Float 型,因为 Float 型具有更高优先级。

表 114 数据优先级规则

等级	数据类型
1	Double
2	Float
3	Int64
4	UInt64
5	Int32
6	UInt32
7	StatusCode
8	Int16
9	UInt16
10	SByte
11	Byte
12	Boolean
13	Guid
14	String
15	ExpandedNodeId
16	NodeId
17	LocalizedText
18	QualifiedName

运算对象也可能包含空值(比如,不存在的值)。出现这种情况时会将该元素视为 NULL(除非指定了 IsNull 运算符)。表 115 定义了在逻辑 AND 运算中如何组合 NULL 和其他元素。

表 115 逻辑 AND 真值表

	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

表 116 定义了在逻辑 OR 运算中如何组合 NULL 和其他元素。

表 116 逻辑 OR 真值表

	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

如果运算对象为 NULL，NOT 运算符的结果为 NULL。

当评价完所有元素都为 NULL 后，ContentFilter 被评为 FALSE。

7.4.4 FilterOperand(过滤器对象)参数

7.4.4.1 概述

7.4 描述的 ContentFilter 结构定义了一组过滤器标准，并包含不同 FilterOperand 类型。FilterOperand 是一个可扩展参数。这个参数定义在表 117 中。ExtensibleParamter 类型定义在 7.11。

表 117 FilterOperand 参数类型 Id

符号 Id	描述
Element	指定成员数组中的索引。通过链接元素的运算对象到子元素，使用该类型来创建子元素的逻辑树
Literal	指定文字值
Attribute	指定对象、或类型系统使用的变量节点的任何属性，以及从引用类型和浏览名称构造的相关路径
SimpleAttribute	指定使用类型定义的对象或变量节点的任何属性，以及从浏览名称构造的相关路径

7.4.4.2 ElementOperand

ElementOperand 提供了在 ContentFilter 中指向子成员的链接。链接以整数的形式用于索引包含在 ContentFilter 中的成员数组。如果索引值不大于它包含的成员索引且未引用不存在的成员，那就认为该索引有效。客户端应用此方法来构造过滤器，以避免循环和无效引用。服务器宜在使用前检查索引，从而防止无效索引。

表 118 定义了 ElementOperand 类型。

表 118 ElementOperand

名称	类型	描述
ElementOperand	structure	ElementOperand 值
Index	UInt32	成员数组中的索引

7.4.4.3 LiteralOperand

表 119 定义了 LiteralOperand 类型。

表 119 LiteralOperand

名称	类型	描述
LiteralOperand	structure	LiteralOperand 值
Value	BaseDataType	文字值

7.4.4.4 AttributeOperand

表 120 定义了 AttributeOperand 类型。

表 120 AttributeOperand

名称	类型	描述
AttributeOperand	structure	地址空间中节点的属性
NodeId	NodeId	类型系统中节点的 NodeId
Alias	String	用于标识或涉及别名的可选参数。别名是可用于描述运算对象和用在过滤器结构其他地方使用的一个象征性的名称
browsePath	RelativePath	nodeId 参数标识的节点的相关浏览路径。参见 7.25 中 RelativePath 的定义
AttributeId	IntegerId	属性的 Id。它应是一个有效的属性 Id。IntegerId 定义在 7.13 中。属性的 IntegerId 定义在 IEC 62541 6 中
IndexRange	NumericRange	这个参数用于标识数组中的单个成员或者数组索引中的一段范围。第一个成员由索引 0 标识。NumericRange 类型定义在 7.21 中。 如果指定的属性对象不是数组的话，就不使用这个参数。然而，如果指定的属性对象是数组并且这个参数未使用的话，那么所有成员都将被包含在这个范围中。如果这个参数不使用就置为 null

7.4.4.5 SimpleAttributeOperand

SimpleAttributeOperand 是 AttributeOperand 的简单表现形态，并且适用于 AttributeOperand 的所有规则也都适用于它。B.1 中提供的示例虽然只使用了 AttributeOperand，但是只要 RelativePath 中的 ReferenceType 都是 HierarchialReferences 的子类型，并且目标是对象或者变量节点时，AttributeOperand 都可被 SimpleAttributeOperand 替换，而且不需要一定有别名。

表 121 定义了 SimpleAttributeOperand 类型。

表 121 SimpleAttributeOperand

名称	类型	描述
SimpleAttributeOperand	Structure	地址空间中节点的属性
typeId	NodeId	TypeDefinitionNode 的 NodeId。 这个参数限制了 TypeDefinitionNode 或其子类型的运算对象实例
BrowsePath[]	QualifiedName	节点的相对路径。 这个参数指定了相对路径，它使用 BrowseNames 列表代替用在 AttributeOperand 中的 RelativePath 结构体。浏览名称列表等同于指定前向引用的 RelativePath，该引用是 HierarchicalReferances 引用类型的子类型。 后面有 browsePath 的节点都应是 NodeClass 对象或变量。 如果此列表为空，那节点就是类型定义的实例
AttributeId	IntegerId	属性的 Id。IntegerId 定义在 7.13 中。 所有服务器都应支持该值属性。其他属性是否支持取决于行规或本标准其他部分所设的要求

表 121（续）

名称	类型	描述
IndexRange	NumericRange	这个参数用于标识数组中的单个成员或者数组索引中的一段范围。第一个成员由索引 0 标识。 如果所选节点不是变量或变量的值不是数组，则忽略该参数。 如果不指定，这个参数就为 null。 如果这个参数未被指定，那数组中的所有值都可使用。 NumericRange 类型的定义在 7.21 中

7.5 Counter(计数器)

用一个 UInt32 类型数据来表示计数器的值。根据计数器的用途来定义它的初始值。系数算法适用于所有的计算，其中该系数为最大值加 1。所以有：

$x+y=(x+y)\bmod(\text{max value}+1)$

比如：

Max value + 1 − 0

Max value + 2 = 1

7.6 ContinuationPoint(延长点)

ContinuationPoint 用于暂停一个浏览操作或优先查询操作，同时允许稍后通过调用 BrowseNext 或 QueryNext 来重新启动。当结果数超越了客户端或服务器的限制时，操作将被暂停。

客户端规定了每次请求消息操作中结果的最大个数。服务器不应返回超过这个最大数的结果，而只会比它少。如果这里有多个需要返回的结果，那服务器就会分配 ContinuationPoint。如果返回一个 ContinuationPoint，那服务器应返回至少一个结果。

服务器在每个会话中至少支持一个 ContinuationPoint。服务器在 IEC 62541-5 定义的 ServerCapabilities 对象中为每个会话指定 ContinuationPoint 的最大个数。ContinuationPoint 保持激活除非客户端重新返回剩余的结果、客户端释放了 ContinuationPoint 或者会话被关闭。如果需要处理一个新的请求，那么服务器会自动释放先前请求的 ContinuationPoint。如果客户端尝试使用一个被释放了的 ContinuationPoint，那服务器将返回 Bad_ContinuationPointvalid 错误。

请求常常指定多个可能需要或不需要 ContinuationPoint 的操作。服务器应处理这些操作，直到它们运行在 ContinuationPoint 以外。一旦发生这种情况，服务器将为每个未完的操作返回 Bad_NoContinuationPoints 错误。

客户端通过将 ContinuationPoint 发送给服务器来继续暂停的操作。当继续一个先前被挂起的操作时，服务器总能重用预备好了的 ContinuationPoint，而不会返回 Bad_NoContinuationPoints 错误。

ContinuationPoint 是 ByteString 数据类型的子类型。

7.7 DataValue(数值)

7.7.1 概述

这个参数的组成部分定义在表 122 中。

表 122 DataValue

名称	类型	描述
DataValue	structure	值和关联信息
Value	BaseDataType	数据的值
statusCode	StatusCode	状态码定义了服务器访问/提供值的能力。状态码的定义在 7.33 中
sourceTimestamp	UtcTime	值的源时间戳
sourcePicoSeconds	UInteger	指定附加在源时间戳上的 10 ps(10×10^{-12} s)间隔数
serveTimestamp	UtcTime	值的服务器时间戳
serverPicoSeconds	UInteger	指定附加在服务器时间戳上的 10 ps(10×10^{-12} s)间隔数

7.7.2 PicoSeconds(皮秒)

一些应用要求高分辨率的时间戳。PicoSeconds 字段允许应用指定 10 ps 分辨率的时间戳。PicoSeconds 字段的实际大小取决于 UtcTime 数据类型的分辨率。比如,如果 UtcTime 数据类型具有 100 ns 的分辨率,那么 PicoSeconds 属性只能存储最大 10 000 的值来满足 10 ps 的分辨率。UtcTime 数据类型的分辨率取决于 IEC 62541-6 中定义的映射。

7.7.3 SourceTimestamp(源时间戳)

sourceTimestamp 用来反映数据源中用于变量值的时间戳。一个值一旦与源时间戳关联了,那值的实例的时间戳就不会改变。这里,"值的实例"是指接收到的值,不依赖于它实际的值。

SourceTimestamp 应是 UTC 时间,并且表示了值或状态码最后变化的时间。

SourceTimestamp 的产生应该尽可能的贴近值的源,不过时间戳必须要按相同的物理时钟来设定。在冗余源的情况下,冗余数据源的时钟也应该是同步的。

如果 OPC UA 服务器从其他 OPC UA 服务器接收到变量值,那它会无改变的将源时间戳传递过来。如果申请时间戳的数据源不可用,那源时间戳要置为 Null。比如,如果一个值因为某些诸如请求中的无效参数之类的错误原因不能读,那么源时间戳就应该置为 Null。

如果 SourceTimestamp 处于不良状态或者不确定状态下,源时间戳被用于反映认识到 SourceTimestamp 处于不良状态或从不确定状态恢复认证的时间。

源时间戳只返回一个值属性。返回的其他属性都被源时间戳置为 null。

7.7.4 ServerTimestamp(服务器时间戳)

ServerTimestamp 用于反映服务器接收到变量值的时间或者知道它是准确数值的时间。

在 ServerTimestamp 处于不良或者不确定的状态下,ServerTimestamp 用于反映服务器接收到该状态的时间或者上一次从不良状态或不确定状态中恢复的时间。

当 OPC UA 服务器向其他 OPC UA 服务器订阅一个数值时,每个服务器都应用它自己的 ServerTimestamp。它不同于 sourceTimestamp,那就是只能由数据源使用 sourceTimestamp。

如果一个服务器每隔十秒钟就向其他服务器订阅一个变化的数值,那么 ServerTimestamp 会在每收到一个新数值时更新。如果该值未改变,那在订阅中将不会接收新数值。然而,在没有错误的情况下,接收服务器每十秒钟应用一个新的 ServerTimestamp,因为没收到值意味着值没有改变。如此,ServerTimestamp 反映了服务器获知准确数值的时间。

这个概念也应用于从基于异常的数据源接收数值的 OPC UA 服务器。比如,假定服务器从基于异

常的数据源接收数值，并且

a) 设备每隔 0.5 s 检查一次数值；

b) 设备连接正常；

c) 设备 3 分钟前发送了数值 1234。

这时，服务器端的值就是 1234，同时在收到数值之后，ServerTimestamp 也会每隔 0.5 s 刷新。

7.7.5 分配给数值的 StatusCode(状态码)

状态码用于表示变量值生成的条件，同时也可用于数值可用性的一个标记。状态码定义在 7.33 中。

整体环境(严格性)：

具有严格性为好的状态码意味着数值的质量是好的。

具有严格性为不确定的状态码意味着数值的质量是不确定的，其原因由子状态标记。

具有严格性为坏的状态码意味着数值是不可用的，其原因由子状态标记。

规则：

——状态码表示了数值是否可用。因此，它就要求客户端在访问和使用一个数值前，就算不检查其他字段，至少要对每个结果检查其状态码严格性。

不支持状态信息的服务器应返回严格性为好的状态码。也接受服务器简单返回一个严格性和非特定(0)子状态。

如果服务器不知道数值，特别是当严格性为坏时，返回的数值将为 NULL。

7.8 DiagnosticInfo(诊断信息)

该参数的组成部分定义在表 123 中。

表 123 DiagnosticInfo

名称	类型	描述
DiagnosticInfo	structure	供应商特定的诊断信息
Identifier	structure	供应商特定的一个错误或条件的标识符
namespaceUri	Int32	由 symbolicIdIndex 定义的符号 Id 是由名称空间中的文本所定义。名称空间由字符串组成并由定义在 7.27 中的 ResponseHeader 参数的 stringTable 参数传递给客户端。名称空间索引参数包含了这个字符串在 stringTable 中的索引。−1 表示未指定字符串
symbolicId	Int32	供应商特定的表示一个错误或条件的标识字符串。串最大长度为 32 个字符。希望返回一个数字代码的服务器应将返回码转化为字符串，同时在标识符中返回该字符串。 由定义在 7.27 中的 ResponseHeader 参数的 stringTable 参数将标识字符串传递给客户端。symbolicIdIndex 参数包含这个字符串在 stringTable 的索引。−1 表示未指定字符串
Locale	Int32	供应商特定的区域化文本中用于描述符号 Id 区域文本的区域部分。 由定义在 7.27 中的 ResponseHeader 参数的 stringTable 参数将区域化文本传递给客户端。localizedTextIndex 参数包含这个字符串在 stringTable 的索引。−1 表示未指定字符串

表 123（续）

名称	类型	描述
localizedText	Int32	供应商特定的用于描述符号 Id 的区域文本字符串。该文本字符串的最大长度是 256 个字符。 由定义在 7.27 中的 ResponseHeader 参数的 stringTable 参数将区域文本传递给客户端。localizedTextIndex 参数包含这个字符串在 stringTable 的索引。−1 表示未指定字符串
additionalInfo	String	供应商特定的诊断信息
innerStatusCode	StatusCode	内部运算产生的状态码。 在 OPC UA 请求处理的过程中，许多应用都会在系统内部进行程序调用。OPC UA 服务器可以选择在诊断信息系统中报告系统内部的状态
innerDianosticInfo	DiagnosticInfo	内部状态码相关的诊断信息

7.9 EndpointDescription（终端描述）

该参数各个部分的定义在表 124 中。

表 124 EndpointDescription

名称	类型	描述
EndpointDescription	Structure	描述一个服务器的终端
endpointUrl	String	所描述终端的 URL
Server	ApplicationDescription	描述终端归属的服务器。 ApplicationDescription 类型定义在 7.1 中
serverCertificate	ApplicationInstanceCertificate	发布给服务器的应用实例证书。 ApplicationInstanceCertificate 定义在 7.2 中
securityMode	Enum MessageSecurityMode	消息所用的安全类型。 MessageSecurityMode 类型定义在 7.14 中。 即使没有配置安全模式也必须要创建安全通道。确切的行为取决于所用的映射，并在 IEC 62541-6 描述
securityPolicyUri	String	保护消息时使用的安全策略的 URI。 已知的 URI 集合和对应的安全策略定义在 IEC 62541-7 中
UserIdentityTokens[]	UserTokenPolicy	服务器能够接收的用户身份令牌。 客户端在 ActiveSession 请求中会发送一个 UserIdentityToken。 UserTokenPolicy 类型描述在 7.36 中
TransportProfileUri	String	终端支持的传输行规的 URI。 IEC 62541-7 中定义了传输行规的 URIs
securityLevel	Byte	对同一服务器而言，表示一个终端描述相对比其他终端描述的安全程度的一个数字。 0 表示不推荐的终端描述，同时这个参数仅能向后兼容

7.10 ExpandedNodeId(扩展节点 ID)

该参数各个部分的定义在表 125 中。ExpandedNodeId 允许用字符串或者服务器名称空间表索引来明确指定名称空间。

表 125 ExpandedNodeId

名称	类型	描述
ExpandedNodeId	structure	将名称空间的节点 Id 扩展为字符串表示
serverIndex	Index	表示包含目标节点的服务器的索引。他可以是本地或远程的服务器。 表示服务器在本地服务器列表中的索引号。本地服务器在服务器列表中的索引通常就是 0。所有远程服务器索引都大于 0。服务器列表包含在地址空间的服务器对象中(参见 IEC 62541-3 和 IEC 62541-5)。 客户端会读取服务器列表变量来访问目标服务器的描述
namespaceUri	String	名称空间的 URI。如果指定了这个参数,那名称空间索引就被忽略。 5.4 和 IEC 62541-12 描述了将 URIs 分解为 URLs 的发现机制
namespaceIndex	Index	服务器名称空间列表的索引。 如果指定了名称空间 URI,此参数就要置 0,同时被服务器忽略掉
identifierType	IdType	节点 Id 的标识符成员类型
Identifier	*	OPC UA 服务器地址空间中节点的标识符(参见 IEC 62541-3 中节点 Id 的定义)

7.11 ExtensibleParameter(可扩展参数)

可扩展参数的类型只能按以下本系列标准的附加部分进行扩展。

ExtensibleParameter 定义了拥有两个成员的数据结构。parameterTypeId 指定了第二个成员的数据类型编码。因此,第二个成员指定为“—”。ExtensibleParameter 基本类型定义在表 126 中。

OPC UA 通用的可扩展参数具体定义在第 7 章中。本系列标准的附加部分中定义了额外的参数类型。

表 126 ExtensibleParameter 基本类型

名称	类型	描述
ExtensibleParameter	structure	指定了可扩展参数类型的细节
ParameterTypeId	NodeId	表示跟随在后的参数的数据类型
parameterData		可扩展参数类型的细节

7.12 Index(索引)

此基本数据类型为 UInt32,表示数组的元素。

7.13 IntegerId(整型 ID)

此基本数据类型为 UInt32,像句柄一样用作标识符。除 0 之外的所有值都是有效的。

7.14 MessageSecurityMode(消息安全模式)

MessageSecurityMode 为枚举型,规定了在会话中消息交互使用何种信息安全模式。可能的值见表 127。

表 127 MessageSecurityMode 值

值	描述
INVALID_0	MessageSecurityMode 无效。 该值为缺省值,用于避免没有指定信息安全模式的偶然情况。它应总是被拒绝
SIGN_1	所有信息都是有符号数,但未被加密
SIGNANDENCRYPT_2	所有信息都是有符号数,且被加密

7.15 MonitoringParameter(监控参数)

组成该参数的各个部分定义在表 128 中。

表 128 MonitoringParameters

名称	类型	描述
MonitoringParameters	structure	定义监视项的监视特性的参数
clientHandle	IntegerId	客户端提供的监视项 Id。用于为列表节点产生的通知。IntegerId 类型定义在 7.13 中
samplingInterval	Duration	定义监视项被访问和评价的最快速率。单位为毫秒。 0 表示服务器将使用最快速率。 -1 表示使用订阅发布速率中定义的缺省采样间隔。 服务器用这个参数来表示监视项所支持的采样间隔
Filter	Extensible Parameter MonitoringFilter	被服务器用于确定是否监视项产生一个通知的过滤器。若不使用,参数就为 NULL。MonitoringFilter 参数类型是 7.16 中指定的可扩展参数类型。它指定了可以使用的过滤器类型
queueSize	Counter	被请求的监视项队列大小。如果丢失了事件,EventQueueOverflow 事件就会产生。 如下值有的特殊含义: 值 含义 1 队列有一个条目,可有效禁用排队。 >1 使用先入先出队列。 Max Value 服务器支持的最大值。用于事件通知。在此情况下,服务器要负责管理事件缓冲器。如果客户端送出 0,服务器返回 1 个大小的缺省队列,同时返回的数据监视项的 revisedQueueSize 的大小也是 1。 对于事件监视项,该值会被忽略,同时服务器 应将支持的最大队列大小作为 reviseQueueSize。 如果丢失了事件,就会产生 EventQueueOverflow 类型的事件

表 128（续）

名称	类型	描述
DiscardOldest	Boolean	当队列满时有一个新的通知进入队列时，用于指定丢弃策略的布尔参数。它有下面两个值： TRUE　最先进入队列的通知被丢弃。新通知加入队列末尾。 FALSE　新通知被丢弃。队列保持不变

7.16　MonitoringFilter（监控过滤器）参数

7.16.1　概述

CreateMonitoredItem 服务允许为每个监视项指定一个过滤器。MonitoringFilter 是一个可扩展参数，其结构取决于被监视项的类型。该参数的类型 Id 定义在表 129 中。其他类型通过本系列标准的附加部分或基于 OPC UA 的其他规范定义。ExtensibleParameter 类型定义在 7.11 中。

每个 MonitoringFilter 都有一个关联的 MonitoringFilterResult 结构，他向客户端响应中返回修订的参数和/或错误信息。在定义 MonitoringFilter 的部分对结果结构体（如有）进行了描述。

表 129　MonitoringFilter 参数类型 Id

符号 Id	描述
DataChangeFilter	引发通知产生的数据值的改变
EventFilter	如果通知符合事件过滤器，就将通知送到客户端
AggregateFilter	当需要进行计算和产生通知时的集合和间隔

7.16.2　DataChangeFilter（数据变化过滤器）

DataChangeFilter 定义了数据变化引发通知产生的条件，以及不产生通知的数据变化的范围和区域。这个范围称为死区。DataChangeFilter 定义在表 130 中。

表 130　DataChangeFilter

名称	类型	描述
DataChangeFilter	structure	
Trigger	Enum DataChangeTrigger	指定引发通知的数据变化的条件。有以下几个值： STATUS_0　仅在状态码与变化值关联的情况下引发通知。参见表 166 中状态码定义的标准。IEC 62541-8 指定了针对特殊设备数据有效的附加状态码。 STATUS_VALUE_1　如果状态码或值有变化就引发通知。死区过滤器可用作值变化的附加过滤器。**如未设置，此参数就作为缺省值。** STATUS_VALUE_TIMESTAMP_2　如果状态码、值或源时间戳有变化就引发通知。死区过滤器可用作值变化的附加过滤器。 如果 DataChangeFilter 未应用于监视项，那 STATUS_VALUE_1 就是缺省的通知行为

表 130（续）

名称	类型	描述
DeadbandType	UInt32	定义死区类型和行为的值。 值　　死区类型 None_0　　不宜使用死区计算。 Absolute_1　绝对死区(见下面)。 Precent_2　百分数死区(类型定义在 IEC 62541-8)
DeadbandValue	Double	仅当以下条件时才应用死区： * 触发器包含了值变化，同时 * 设置了适当的死区类型。 如果数据项状态变化，则忽略死区。 **死区类型 = 绝对死区：** 对于这一类型，死区值包含引发通知的数据值中的绝对变化。这个参数仅应用于数字数据类型的变量。 引发基于绝对死区的数据变化通知的异常的定义如下： **如果(上一次缓存值－当前值)的绝对值 > 绝对死区**，就引发异常。 上一次缓存值是先前最近一次送到通知通道的值。 如果该项是一个值的数组，那当任一数组成员超过绝对死区时就会返回整个数组。 **死区类型 = 百分数死区：** 这个类型在 IEC 62541-8 中指定

DataChangeFilter 没有关联的结果结构体。

7.16.3 EventFilter(事件过滤器)

EventFilter 提供了事件订阅的过滤和内容选择。

如果事件通知符合 EventFilter 参数的定义的过滤器，那通知就会送往客户端。

每个事件通知都应包含由 EventFilter 的 selectClause 参数定义的字段。事件类型定义描述在 IEC 62541-5 中。

SimpleAttributeOperand 结构体中指定了 selectClause 和 whereClause 参数(参见 7.4.4.5)。这个结构体要求服务器支持的事件类型的节点 Id 和实例声明的路径。实例声明是一个能从完全继承的事件类型中前向层次引用发现的节点，其中该节点也是 HasModellingRule 引用的源节点。事件类型、实例声明和建模规则的描述在 IEC 62541-3 中。

一些时候，相同的浏览路径会应用于多个事件类型。如果客户端在 SimpleAttributeOperand 中指定了基本事件类型，那么服务器将评估浏览路径而不考虑其类型。

路径中的每个实例声明都应是对象或变量节点。路径中最后一个节点可以是一个对象节点，然而，对象节点仅对服务器地址空间中可见的事件有效。

SimpleAttributeOperand 结构体允许客户端指定任意一个属性，然而，服务器只要求支持变量节点的值属性和对象节点的 NodeId 属性。那就是说，定义在 IEC 62541-7 中的行规会支持附加的强制属性。

如果事件满足了 whereClause 规定的标准，那么 selectClause 中的 SimpleAttributeOperand 结构体就被用于选择一个返回值。如果所选的字段不是事件的一部分或者事件过滤结果的 selectClauseResults 中返回一个错误，那么对应的事件字段在发布响应中将返回一个空值。如果所选字段可用但是不能返回给客户端，那么服务器将返回标识错误原因的状态码。比如，如果数值对会话相关的用户来说不

可访问，那么服务器返回 Bad_UserAccessDenied 错误。如果一个值属性的状态码是不确定或不良状态，那么服务器将返回状态码而不是这个值。

当客户的创建或更新事件过滤器时，服务器应确认 selectClauses 的有效性。事件集中的任何错误都会包含在表 132 定义的 selectClausesResults 参数中被返回。服务器不会报告可能依赖于服务器状态或事件类型引发的错误。比如，如果属性参数的数据类型字段是一个标量，那么请求数组中单个成员的 selectClause 服务就会引发一个错误。然而，如果针对指定事件的请求索引不存在，即使数据类型字段是一个数组都会引发一个错误。如果存在后面的情况，服务器就不会在 selectClauseResults 参数中报告错误。

whereClause 字段中的 SimpleAttributeOperand 是用于选择组成逻辑表达式的一个值。这些逻辑表达式用于确定是否向客户端报告指定的事件。当给指定的事件应用 whereClause 时，任一错误的发生都会导致服务器使用空值。如果关联的值属性是不确定或不良状态码时，那么服务器将使用空值来代替该值。

表 110 中的任一基本过滤器操作符都可以用在 whereClause 中，然而，仅有表 111 中 OfType 过滤器操作符可以使用。

当客户端创建或更新事件过滤器时，服务器应确认 whereClause 的有效性。构造过滤器时的任何结构错误和对于任何可能的事件都存在的错误都在表 132 描述的 whereClauseResult 参数中被返回。依赖于服务器状态或事件状态引发的错误不会被报告。

订阅控制事件是专门用于向客户端提供控制信息的特殊事件。这些事件仅向那些产生订阅控制事件的订阅中的监视项发布。这些事件都绕过 whereClause。

表 131 定义了 EventFilter 结构体。

表 131 EventFilter 结构体

名称	类型	描述
EventFilter	structure	
SelectClauses[]	SimpleAttributeOperand	返回的通知器中每个事件的值列表。至少指定一个有效条款。*SimpleAttributeOperand* 定义在 7.4.4.5 中
whereClause	ContentFilter	限制匹配内容过滤器定义的事件的通知。内容过滤器结构体的描述在 7.4 中。*AttributeOperand* 结构体不用在事件过滤器中

表 132 定义了 EventFilterResult 结构体。

表 132 EventFilterResult 结构体

名称	类型	描述
EventFilterResult	structure	
SelectClausesResults[]	StatusCode	所选条款中成员的状态码列表。列表的大小和顺序与 selectClauses 请求参数中成员的大小和顺序相一致。服务器对无效或拒绝的事件字段返回空值
SelectClausesDianosticInfos[]	DiagnosticInfo	所选条款中单个成员的诊断信息列表。列表的大小和顺序与 selectClauses 请求参数中成员的大小和顺序相一致。如果在请求报文首部中未请求诊断信息或者在处理所选条款过程中没有诊断信息，那么列表就为空
WhereClauseResult	ContentFilterResult	关联 whereClause 请求参数的结果。 ContentFilterResult 类型定义在 7.4.2 中

表 133 定义了 selectClausesResults 参数的值。常见状态码定义在表 166 中。

表 133 EventFilterResult 代码

符号 Id	描述
Bad_TypeDefinitionInvalid	关于该结果代码的描述参见表 166。 该类型 Id 不是基础事件类型的节点 Id 或者它的子类型
Bad_NodeIdUnknown	关于该结果代码的描述参见表 166。 指定了浏览路径但是它不存在于任何事件中
Bad_BrowseNameInvalid	关于该结果代码的描述参见表 166。 指定了浏览路径，同时它还包含一个空成员
Bad_AttributeIdInvalid	关于结果代码的描述参见表 166。 浏览路径指定的节点不允许返回给定的属性 Id
Bad_IndexRangeInvalid	关于该结果代码的描述参见表 166
Bad_TypeMismatch	关于该结果代码的描述参见表 166。 索引范围有效但是属性的值并不是一个数组

7.16.4 AggregateFilter(聚合过滤器)

AggregateFilter 定义了可用于计算返回值的集合函数。参见 IEC 62541-13 关于可能的集合函数的细节描述。它指定了第一个被计算的集合的开始时间。监视属性(参见 7.15)的采样间隔定义了服务器从数据源采样的内部时间间隔。处理间隔指定了计算集合的周期大小。监控属性的队列尺寸指定了被保留处理的值的数量。

AggregateFilter 的目的不是为了读取历史数据，历史读服务才会用于这个目的。然而，它允许将开始时间设置为过去从服务器接收的时间。先前被计算的集合数量不应超过监控属性中定义的队列尺寸，因为超出队列尺寸的值会直接被丢失并且不会返回到客户端。

开始时间和处理间隔可以被服务器修改，但是开始时间应保持同样的约束(开始时间 — 修改后的处理间隔 * n = 修改后开始时间)。通过调用历史读服务来访问集合历史数据的单一行为使用相同的约束。可扩展参数 AggregateFilterResult 用于返回 AggregateFilter 修改了的值。

一些系统可能用相同的值来轮询数据和产生多重抽样。另一些系统可能仅仅报告值的变化。每个集合类型的定义解释了如何处理两个不同的方案。

监视项仅仅报告发布定时器终止前已经完成的值。未使用的数据被转移，并用来计算下一次发布返回的值。

每个间隔的 *ServerTimestamp* 都应是处理过程间隔的结束时间。

AggregateFilter 定义在表 134 中。

表 134 AggregateFilter 结构体

名称	类型	描述
AggregateFilter	structure	
startTime	UtcTime	第一次计算集合的开始时间。用于计算集合的每个周期的尺寸通过监控属性(见 7.15)的采样间隔来定义
aggregateType	NodeId	历史集合对象的节点 Id 表示了重新找回处理数据时使用的集合列表。更多细节见 IEC 62541-13

表 134（续）

名称	类型	描述
processingInterval	Duration	用于计算集合的时间周期
aggregateConfiguration	AggregateConfiguration	这个参数允许客户端重载由每个监视项上的 AggregateConfiguration 对象提供的集合配置设置。集合配置的更多信息参见 IEC 62541-13。如果服务器不支持重载集合配置的设置，那么它将返回 Bad_AggregateListMismatch 状态码
useServerCapabilities Defaults	Boolean	如果值为 TRUE，就使用 AggregateConfiguration 对象所设的集合配置设置。 如果值为 FALSE，就使用下面所设的设置。 缺省值为 TRUE
TreatUncertainAsBad	Boolean	描述在 IEC 62541 13
percentDataBad	Byte	描述在 IEC 62541 13
percentDataGood	Byte	描述在 IEC 62541-13
steppedSloped Extrapolation	Boolean	描述在 IEC 62541-13

当一个 AggregateFilter 定义给 CreateMonitoredItems 或 ModifyMonitoredItems 中的监视项时，AggregateFilterResult 就定义了服务器返回的修改后的 AggregateFilter。AggregateFilterResult 定义在表 135 中。

表 135　AggregateFilterResult 结构体

名称	类型	描述
AggregateFilterResult	structure	
revisedStartTime	UtcTime	服务器实际使用的开始时间。 这个值基于若干因子，包括服务器访问历史数据的能力。该参数和开始时间有着一致的边界（开始时间 − 采样间隔 × n = 修订的开始时间）
revisedProcessingInterval	Duration	服务器实际使用的处理间隔。 该参数应至少是监视项的 revisedSamplingInterval 的两倍

7.17　MonitoringMode（监视模式）

MonitoringMode 是一个枚举数据，它规定是否需要启动对监视项采样和报告。对订阅的发布使能参数不影响订阅的监视项的监视模式的值。该参数值的定义见表 136。

表 136　MonitoringMode 值

值	描述
DISABLED_0	监视项不被采样或评估，通知也不产生或排队。通知报告被禁用
SAMPLING 1	监视项要采样和评估，通知也要产生和排队。通知报告被禁用
REPORTING_2	监视项要采样和评估，通知要产生和排队。通知报告被启用

7.18 NodeAttribute(节点属性)参数

7.18.1 概述

AddNodes 服务允许规定要添加哪些节点属性。NodeAttribute 是一个扩展参数,参数的结构体依赖于添加时的属性类型。它定义了用于定义属性结构体的节点类。参数类型 Id 的定义见表 137。扩展参数类型定义见 7.11。

表 137 NodeAttribute 参数类型 Id

符号 Id	描述
ObjectAttributes	定义对象节点类的属性
VariableAttributes	定义变量节点类的属性
MethodAttributes	定义方法节点类的属性
ObjectTypeAttributes	定义对象类型节点类的属性
VariableTypeAttributes	定义变量类型节点类的属性
ReferenceTypeAttributes	定义引用类型节点类的属性
DataTypeAttributes	定义数据类型节点类的属性
ViewAttributes	定义视图节点类的属性

表 138 定义了用于节点属性参数的位掩码,规定了哪些属性由客户端设置。

表 138 规定属性的位掩码

字段	位	描述
AccessLevel	0	指示是否设置了 AccessLevel 属性
ArrayDimensions	1	指示是否设置了 ArrayDimensions 属性
保留	2	与 IEC 62541-3 中定义的写掩码保持一致
ContainsNoLoops	3	指示是否设置了 ContainsNoLoops 属性
DataType	4	指示是否设置了 DataType 属性
Description	5	指示是否设置了 Description 属性
DisplayName	6	指示是否设置了 DisplayName 属性
EventNotifier	7	指示是否设置了 EventNotifier 属性
Executable	8	指示是否设置了 Executable 属性
Historizing	9	指示是否设置了 Historizing 属性
InverseName	10	指示是否设置了 InverseName 属性
IsAbstract	11	指示是否设置了 IsAbstract 属性
MinimumSamplingInterval	12	指示是否设置了 MinimumSamplingInterval 属性
保留	13	与 IEC 62541-3 中定义的写掩码保持一致
保留	14	与 IEC 62541-3 中定义的写掩码保持一致
Symmetric	15	指示是否设置了 Symmetric 属性

表 138（续）

字段	位	描述
UserAccessLevel	16	指示是否设置了 UserAccessLevel 属性
UserExecutable	17	指示是否设置了 UserExecutable 属性
UserWriteMask	18	指示是否设置了 UserWriteMask 属性
ValueRank	19	指示是否设置了 ValueRank 属性
WriteMask	20	指示是否设置了 WriteMask 属性
Value	21	指示是否设置了 Value 属性
保留	22～32	保留用。全为零

7.18.2 ObjectAttribute(对象属性)参数

ObjectAttribute 参数定义见表 139。

表 139 ObjectAttribute

名称	类型	描述
ObjectAttributes	Structure	定义对象节点类的属性
specifiedAttributes	UInt32	指示哪些字段包含有效值的一个位掩码。 如果对应位为 0，表示该字段被忽略。 位值定义见表 138
displayName	LocalizedText	见 IEC 62541-3 中的描述
Description	LocalizedText	见 IEC 62541-3 中的描述
evenNotifier	Byte	见 IEC 62541-3 中的描述
WriteMask	UInt32	见 IEC 62541-3 中的描述
userWriteMask	UInt32	见 IEC 62541 3 中的描述

7.18.3 VariableAttribute(变量属性)参数

VariableAttribute 参数定义见表 140。

表 140 VariableAttribute

名称	类型	描述
VariableAttributes	Structure	定义变量节点类的属性
specifiedAttributes	UInt32	指示哪些字段包含有效值的一个位掩码。如果对应位为 0 表示该字段应被忽略。位值定义见表 138
displayName	LocalizedText	见 IEC 62541 3 中的描述
Description	LocalizedText	见 IEC 62541 3 中的描述
Value	由事件类型属性定义	见 IEC 62541 3 中的描述

表 140（续）

名称	类型	描述
dataType	NodeId	见 IEC 62541 3 中的描述
valueRank	Int32	见 IEC 62541-3 中的描述
arrayLevel	UInt32[]	见 IEC 62541-3 中的描述
accessLevel	Byte	见 IEC 62541-3 中的描述
userAccessLevel	Byte	见 IEC 62541-3 中的描述
minimumSamplingInterval	Duration	见 IEC 62541-3 中的描述
Historizing	Boolean	见 IEC 62541-3 中的描述
WriteMask	UInt32	见 IEC 62541-3 中的描述
userWriteMask	UInt32	见 IEC 62541 3 中的描述

7.18.4 MethodAttribute(方法属性)参数

MethodAttribute 参数定义见表 141。

表 141 MethodAttribute

名称	类型	描述
BaseAttributes	Structure	定义方法节点类的属性
specifiedAttributes	UInt32	指示哪些字段包含有效值的一个位掩码。如果对应位为 0 表示该字段应被忽略。位值定义见表 138
displayName	LocalizedText	见 IEC 62541 3 中的描述
Description	LocalizedText	见 IEC 62541 3 中的描述
Executable	Boolean	见 IEC 62541-3 中的描述
userExecutable	Boolean	见 IEC 62541-3 中的描述
WriteMask	UInt32	见 IEC 62541-3 中的描述
userWriteMask	UInt32	见 IEC 62541-3 中的描述

7.18.5 ObjectTypeAttribute(对象类型属性)参数

ObjectTypeAttribute 参数定义见表 142。

表 142 ObjectTypeAttribute

名称	类型	描述
ObjectTypeAttributes	Structure	定义对象类型节点类的属性
specifiedAttributes	UInt32	指示哪些字段包含有效值的一个位掩码。如果对应位为 0 表示该字段应被忽略。位值定义见表 138
displayName	LocalizedText	见 IEC 62541-3 中的描述

表 142（续）

名称	类型	描述
Description	LocalizedText	见 IEC 62541 3 中的描述
isAbstract	Boolean	见 IEC 62541-3 中的描述
WriteMask	UInt32	见 IEC 62541-3 中的描述
userWriteMask	UInt32	见 IEC 62541-3 中的描述

7.18.6 VariableTypeAttribute(变量类型属性)参数

VariableTypeAttribute 参数定义见表 143。

表 143 VariableTypeAttribute

名称	类型	描述
VariableTypeAttributes	Structure	定义变量类型节点类的属性
specifiedAttributes	UInt32	指示哪些字段包含有效值的一个位掩码。如果对应位为 0 表示该字段应被忽略。位的值定义在表 138 中
displayName	LocalizedText	见 IEC 62541-3 中的描述
Description	LocalizedText	见 IEC 62541-3 中的描述
Value	由数据类型属性定义	见 IEC 62541-3 中的描述
dataType	NodeId	见 IEC 62541 3 中的描述
valueRank	Int32	见 IEC 62541 3 中的描述
arrayDimensions	UInt32[]	见 IEC 62541 3 中的描述
isAbstract	Boolean	见 IEC 62541 3 中的描述
WriteMask	UInt32	见 IEC 62541-3 中的描述
userWriteMask	UInt32	见 IEC 62541-3 中的描述

7.18.7 ReferenceTypeAttribute(引用类型属性)参数

ReferenceTypeAttribute 参数定义见表 144。

表 144 ReferenceTypeAttribute

名称	类型	描述
ReferenceTypeAttributes	Structure	定义引用类型节点类的属性
specifiedAttributes	UInt32	指示哪些字段包含有效值的一个位掩码。如果对应位为 0 表示该字段应被忽略。位的值定义见表 138
displayName	LocalizedText	见 IEC 62541-3 中的描述
Description	LocalizedText	见 IEC 62541-3 中的描述
isAbstract	Boolean	见 IEC 62541-3 中的描述

表 144（续）

名称	类型	描述
Symmetric	Boolean	见 IEC 62541 3 中的描述
inverseName	LocalizedText	见 IEC 62541-3 中的描述
WriteMask	UInt32	见 IEC 62541-3 中的描述
userWriteMask	UInt32	见 IEC 62541-3 中的描述

7.18.8 DataTypeAttribute(数据类型属性)参数

DataTypeAttribute 参数定义见表 145。

表 145 DataTypeAttribute

名称	类型	描述
DataTypeAttributes	Structure	定义数据类型节点类的属性
specifiedAttributes	UInt32	指示哪些字段包含有效值的一个位掩码。如果对应位为 0 表示该字段应被忽略。位的值定义见表 138
displayName	LocalizedText	见 IEC 62541-3 中的描述
Description	LocalizedText	见 IEC 62541-3 中的描述
isAbstract	Boolean	见 IEC 62541-3 中的描述
WriteMask	UInt32	见 IEC 62541 3 中的描述
userWriteMask	UInt32	见 IEC 62541 3 中的描述

7.18.9 ViewAttribute(视图属性)参数

ViewAttribute 参数定义见表 146。

表 146 ViewAttribute

名称	类型	描述
ViewAttributes	Structure	定义视图节点类的属性
specifiedAttributes	UInt32	指示哪些字段包含有效值的一个位掩码。如果对应位为 0 表示该字段应被忽略。位的值定义在表 138 中
displayName	LocalizedText	见 IEC 62541 3 中的描述
Description	LocalizedText	见 IEC 62541 3 中的描述
containsNoLoops	Boolean	见 IEC 62541-3 中的描述
evenNotifier	Byte	见 IEC 62541-3 中的描述
WriteMask	UInt32	见 IEC 62541-3 中的描述
userWriteMask	UInt32	见 IEC 62541-3 中的描述

7.19 NotificationData(通知数据)参数

7.19.1 概述

用在 subscription 服务中的通知消息结构允许指定不同的 NotificationData 类型。NotificationData 参数是一个扩展参数,它的结构取决于要发送的通知类型。该参数的定义见表 147。其他类型可以通过附加标准或者根据 OPC UA 的其他指定。扩展参数类型定义见 7.11。

在单个通知数据结构中,单个监视项可能存在多个通知。这种情况下,服务器应保证通知按监视项中排列的顺序出现。这些通知不需要作为相邻块出现。

表 147 NotificationData 参数类型 Id

符号 Id	描述
DataChange	用于数据变化通知的通知数据参数
Event	用于事件通知的通知数据参数
StatusChange	用于订阅状态变化通知的通知数据参数

7.19.2 DataChangeNotification(数据改变通知)参数

表 148 定义了用于数据变化通知的 NotificationData 参数。这些结构包含了将被报告的监视数据项。基于以下两个条件报告监视数据项:

a) 如果 MonitoringMode 为 REPORTING 同时数值出现变化或者状态(由 StatusCode 呈现)被探测到。

b) 如果 MonitoringMode 为 SAMPLING,MonitoringItem 连接到一个触发项,并且后者启动了触发。

MonitoringItem 服务集的描述见 5.12,特别是 MonitoringItemModel 和 TriggeringModel。

创建 MonitoringItem 之后,无需应用过滤器,监视属性的数值或状态也会进行排列。在第一次采样后,如果当前数值无效,那么在从数据源获取到初始值或状态后,第一个通知就会加入队列。

表 148 DataChangeNotification

名称	类型	描述
DataChangeNotification	Structure	数据变化通知的数据
MonitoredItems[]	MonitoredItemNotification	被探测到变化的监视项的列表
ClientHandle	IntegerId	客户端支持的监视项句柄。整型 Id 定义见 7.13
Value	DataValue	依赖于采样和队列配置的监视属性的状态码、数值和时间戳。 如果状态码出现错误,那么数值和时间戳都会被忽略。 在服务器的监视项队列缓冲数据到达极限并卸下多于数据之前,如果不是每个探测到的变化都被返回,那么状态码的数据值中的 Overflow 参数就应被置位。 数据值是一种常见类型,定义见 7.7
DiagnosticInfos[]	DiagnosticInfo	诊断信息列表。列表的大小和顺序与监视项参数的大小和顺序相匹配。列表中,每个包含在监视项参数的节点都有个条目。如果没有申请诊断信息或者哪一个监视项无效,那么列表就为空。诊断信息是一种通用类型,定义见 7.8

7.19.3 EventNotificationList(事件通知列表)参数

表 149 定义了用于事件通知的 NotificationData 参数。

EventNotificationList 是一个用于向客户端订阅返回事件字段的表结构。这个表结构由一个或多个事件组成,每个事件都是包含一个或多个字段的数组。每个事件返回的字段的选择和顺序与 EventFilter 选择的参数一致。

表 149 EventNotificationList

名称	类型	描述
EventNotificationList	Structure	事件通知数据
Events[]	EventFieldList	要发布的事件列表
clientHandle	IntegerId	客户端支持的监视项句柄。整型 Id 见 7.13
EventFields[]	BaseDataType	所选事件字段列表。这个列表应与事件过滤器中所选的字段一一对应。 7.16.3 指示了服务器如何进行错误处理

7.19.4 StatusChangeNotification(状态改变通知)参数

表 150 定义了用于 StatusChangeNotification 的 NotificationData 参数。

StatusChangeNotification 参数通知客户端关于订阅状态中的变化。

表 150 StatusChangeNotification

名称	类型	描述
StatusChangeNotification	Structure	事件通知的数据
Status	StatusCode	表示状态变化的状态码
diagnosticInfo	DiagnosticInfo	状态变化的诊断信息

7.20 NotificationMessage(通知消息)

该参数组成的定义见表 151。

表 151 NotificationMessage

名称	类型	说明
NotificationMessage	Structure	消息包含一个或多个通知
sequenceNumber	Counter	通知报文的序列号
publishTime	UtcTime	消息发往客户端的时间,如果这个报文是重发往客户端,这个参数包含第一次发往客户端的时间
notificationData[]	ExtensibleParameter NotificationData	NotificationData 结构列表。 NotificationData 扩展参数类型规定见 7.19。他指定了可以发送的通知类型。ExtensibleParameter 类型规定见 7.11。类型相同的通知分组到一个通知数据元素中。如果订阅包含事件和值监视项,数组不应超过 2 个元素。如果订阅只包含值或事件其中一个监视项,数组大小总是 1

7.21 NumericRange(数值范围)

该参数定义见表 152。A.3 中可以找到数字范围的正式 BNF(巴科斯范式)定义。

字符串的句法包含以下两种结构。第一种结构是字符串代表了一个单独的整数。例如,"6"是有效的,而"6,0"和"3,2"则无效。最小和最大值可以使用这个参数定义来表示,而不用通过这个参数的类型来定义。第二种结构用冒号(":")分割两个整数来代表一个范围。第一个整数的值始终比第二个小。例如,"5:7"是有效的,而"7:5"和"5:5"则无效。

最小和最大值可以使用这个参数,定义这些整数来表示,而不是通过这个参数的类型来定义。

其他字符包含空格符在内都不允许出现。

多维数组可以通过指定一个范围来索引,每一维度通过','来分开。例如,通过范围"1:2,0:1"能在一个 4×4 矩阵中选择一个 2×2 块。多维数组的一个元素通过指定一个单一数字来选择。例如,"1,1"表示选择一个二维数组中的元素[1,1]。

在 ArrayDimensions(数组维度)属性里规定维度出现的顺序。所有的维度都应指定一个有效的数字范围。

所有索引都从 0 开始,任何一个索引的最大值都比其维度的长度小 1。

当读值的索引范围不在数组的边界内时,服务器将返回索引范围内存在的元素。如果范围内没有元素存在,服务器应返回 Bad_OutOfRange。

当写值时数组的大小应该与指定的数值范围大小匹配,如果客户端指定的元素没有全部写进,服务器应返回一个错误。

数字范围也可以用来指定二进制串和字符串值的子串。在最后的索引在二进制串或字符串中指定了子串的情况下,二进制串和字符串数组可以相当于二维数组。如果最后的索引忽略,则表示选择了整个二进制串或字符串。

表 152 NumericRange

名称	类型	说明
NumericRange	String	一个数字或一个数字范围。 一个空字符串表示该参数不被使用

7.22 QueryDataSet(查询数据集)

此参数组成在表 153 中定义。

表 153 QueryDataSet

名称	类型	说明
QueryDataSet	Structure	查询应答中返回的节点关联数据
nodeId	ExpandedNodeID	节点描述的节点 Id
TypeDefinitionNode	ExpandedNodeID	节点描述的类型定义,该类型定义的节点 Id
values []	BaseDataType	选定的属性值。返回的顺序和请求项的顺序一致。有一个为给定类型定义节点的每一请求项匹配的所选实例,它包括从选定类型定义节点的实例使用相对路径的关联节点。给定的请求项如果没有值,返回一个空值。如果请求项有多个值,返回一个数组。如果请求项是引用,返回一个引用描述或引用描述数组

7.23 ReadValueId(读值 Id)

此参数组成在表 154 中定义。

表 154 ReadValueId

名称	类型	说明
ReadValueId	Structure	用于读和监视的标识项
nodeId	NodeId	一个节点的节点 ID
attributeId	IntegerId	属性的 Id,这应是一个有效的属性 Id。IntegerId 在 7.13 中定义。属性的 IntegerId 在 IEC 62541-6 中定义
IndexRange	NumericRange	该参数用来标识一个数组中的一个元素或数组中一个索引范围。如果指定了一个范围内的元素,那将返回一个组合的值。第一个元素用索引 0 来表示。数字范围类型在 7.21 中定义。如果指定的属性不是一个数组,这个参数将为空。如果指定的属性是一个数组,同时这个参数为空,则表示包含范围内的所有元素
dataEncoding	QualifiedName	这个参数规定了数据类型编码的浏览名称,服务器在返回一个变量的值时会使用数据类型编码。为其他属性指定这个参数是错误的。 一个客户端可以通过一个变量的数据类型节点得到包含编码引用,然后再通过 HasEncoding 引用可以发现哪个数据类型编码是可用的。 OPC UA 定义的浏览名应被服务器识别,即使该数据类型节点在服务器地址空间中不可见。这些浏览名是: DefaultBinary 缺省或本地的二进制(非 XML)编码。 DefaultXML 缺省 XML 编码。 每种数据类型至少支持其中一种编码。如果数据类型没有真正的二进制编码(如,它们只有一种非 XML 的文本编码),将使用 DefaultXML 名来标识编码,DefaultXML 名被视为非 XML 编码的缺省值。 数据类型支持至少一种基于 XML 编码情况下,应标识为 Default-XML 编码的一种编码。予以支持其他标准机构定义的知名编码。 如果这个参数没有被指定,服务器应选择 DefaultBinary 编码或者 DefaultXML 编码,依照消息编码(见 IEC 62541 6)来会话。如果服务器不支持这个消息编码,服务器将选择它支持的缺省编码。 如果该参数被规定用于监视项,服务器应在状态码(见 7.33)里 StructureChanged 比特。如果数据类型描述的数据类型版本或者关联的数据类型字典的数据类型编码变化,则数据类型编码改变

7.24 ReferenceDescription(引用描述)

此参数组成在表 155 中定义。

表 155 ReferenceDescription

名称	类型	说明
ReferenceDescription	Structure	浏览服务返回的引用参数
referenceTypeId	NodeId	定义了引用的引用类型，该引用类型的节点 Id
isForward	Boolean	如果参数为 TRUE，服务器遵循向前引用，如果为 FALSE，服务器遵循一个相反的引用
nodeId	ExpandedNodeId	服务器指派的目标节点的节点 Id，由服务器索引来识别。扩展节点类型定义见 7.10。 如果服务器索引表明目标节点是一个远程节点，节点 Id 应包含绝对命名空间 URI。如果目标节点是一个本地节点，节点 Id 应包含命名空间索引
browseName*	QualifiedName	目标节点浏览名
displayName	LocalizedText	目标节点的显示名
nodeClass*	NodeClass	目标节点的节点类
typeDefinition*	ExpandedNodeId	目标节点的类型定义节点 Id。类型定义只适用于节点类对象和变量。所有其他节点类，应返回一个 null 节点 Id
* 如果服务器索引指明目标节点是一个远程节点，浏览名、节点类和类型定义可能是 NULL 或者空。如果不是，它们可能不是最新的，因为本地服务器可能不会连续监视远程服务器上的变化。		

7.25 RelativePath(相关路径)

此参数组成在表 156 中定义。

表 156 RelativePath

名称	类型	说明
RelativePath	structure	定义一个可跟随的引用和浏览名序列
Elements[]	RelativePath Element	一个可跟随的引用和浏览名序列。 对序列中每个元素的处理，先是发现目标，然后以这些目标作为下一个元素的起始节点。结尾元素的目标是相对路径的目标
referenceTypeId	NodeId	从当前节点跟随的引用类型。 如果引用类型 Id 在节点实例上是不可用的，当前路径不能被进一步跟随
isInverse	Boolean	指示是否跟随反向引用。值为 TRUE 时跟随反向引用
includeSubtypes	Boolean	指出引用类型的子类型是否被跟随。值为 TRUE 时包含子类型
targetName	QualifiedName	目标节点的浏览名。 结尾元素可以有一个空 targetName。这种情形下所有通过引用类型 ID 标识的引用目标都是 RelativePath 的目标。 应为所有其他元素来规定 targetName。 如果指定的已存在的浏览名称没有目标，当前路径将不在被跟随

RelativePath 能用于任何起始节点，RelativePath 目标是一个节点集合，它们可以根据 RelativePath 元素中的序列找到。

RelativePath 的文本格式可以在 A.2 中找到。这个格式在例子中用于解释利用了 RelativePath 结构的服务。

7.26 RequestHeader(请求首部)

此参数组成在表 157 中定义。

表 157 RequestHeader

名称	类型	说明
RequestHeader	structure	一个会话中提交的所有请求的通用参数
authenticationToken	Session AuthenticationToken	保密会话标识,用来验证关联的会话请求。AuthenticationToken 类型在 7.29 中定义
timestamp	UtcTime	客户端发送请求的时间
requestHandle	IntegerId	与请求关联的请求句柄。这个客户端定义的句柄可以用来取消这个请求。这个操作也有应答返回
returnDiagnostics	UInt32	一个位掩码,在 DiagnosticInfo 响应参数返回中用于标识供应商特定诊断类型。 这个参数的值包含零个、一个或多个下面的值。没有值表明没有返回诊断结果。 位值 返回诊断结果 0x0000 0001 ServiceLevel / SymbolicId 0x0000 0002 ServiceLevel / LocalizedText 0x0000 0004 ServiceLevel / AdditionalInfo 0x0000 0008 ServiceLevel / Inner StatusCode 0x0000 0010 ServiceLevel / Inner Diagnostics 0x0000 0020 OperationLevel / SymbolicId 0x0000 0040 OperationLevel / LocalizedText 0x0000 0080 OperationLevel / AdditionalInfo 0x0000 0100 OperationLevel / Inner StatusCode 0x0000 0200 OperationLevel / Inner Diagnostics 这些值都有两部分组成,级别和类型,在下面描述。如果没有请求,由 0 值表示,或者处理请求中没有遇到诊断信息,此时不返回诊断信息。 级别: ServiceLevel 在服务诊断信息里返回的诊断结果。 OperationLevel 服务为个别请求行为定义的诊断信息里,返回的诊断结果。 类型: SymbolicId 返回一个命名空间限定,一个错误或条件的符号标识符。这个标识符的最大长度是 32 个字符。 LocalizedText 最多返回 256 字节用于描述符号 Id 的本地化文本。 AdditionalInfo 返回一个包含附加诊断信息的字节字符串,如内存映像。供应商格式化字节字符串,并可能取决于遇到的错误或条件类型。 InnerStatusCode 返回关联的操作或服务的内部状态码。 InnerDiagnostics 返回与操作或服务关联的内部诊断信息。内部诊断信息结构内容由掩码中的其他位决定。 **注意**:设置这个位可能引起在返回结果中包含多层次嵌套的诊断信息结构。

表 157（续）

名称	类型	说明
auditEntryId	String	一个标识符用于标识与请求关联的客户端的安全审核日志条目。 一个空字符串表示这个参数没有被使用。 AuditEntryId 通常包含谁开始的行为和以及从哪里开始的。在 AuditEvent 中包含的 AuditEventId，允许阅读者将初始行为与一个事件做关联。 更多的审核机制细节在 6.2 中和 IEC 62541-3 中定义
timeoutHint	UInt32	超时以毫秒为单位，用于客户端通信栈，设置每次基础调用的超时。 对服务器这个参数只是一个提示，可以用来取消长运行操作，并释放资源。如果服务器检测到超时，它可以通过发送 Bad_Timeout 服务结果来取消操作，服务器在收到请求后，在取消这个操作前，需要等待最小化超时。 值为 0 时表示没有超时
additionalHeader	Extensible Parameter AdditionalHeader	保留给将来使用。 不理解该参数的应用应该忽略它

7.27 ResponseHeader（响应首部）

此参数的组成在表 158 中定义。

表 158 ResponseHeader

名称	类型	说明
ResponseHeader	Structure	所有应答通用参数
timestamp	UtcTime	服务器发送响应的时间
requestHandle	IntegerId	客户端请求时给的请求句柄
serviceResult	StatusCode	OPC UA 定义的服务调用结果。状态码定义见 7.33
serviceDiagnostics	DiagnosticInfo	服务调用诊断信息。如果诊断信息在请求首部里没有设置，这个参数为空。DiagnosticInfo 定义见 7.8
stringTable []	String	在应答中包含了所有的诊断信息参数，诊断信息中包括了每一个惟一的命名空间，符号化标识以及本地化文本字符，它们都是以字符串形式存储在列表中。在表里每一项索引都是从 0 开始
additionalHeader	Extensible Parameter AdditionalHeader	保留给将来使用。 应用程序不理解该参数可以忽略它

7.28 ServiceFault（服务故障）

该参数组成在表 159 中定义。

当发生服务级别错误时，将返回 ServiceFault 参数用于替代服务响应消息。由 RequestHeader 提

供的 requestHandle,在 ResponseHeader 里应被设置,即使这些参数是无效的。在 ResponseHeader 里返回的诊断结果的级别,由 RequestHeader 里的 returnDiagnostics 参数指定。

此参数的具体使用取决于在 IEC 62541-6 中定义的映射。

表 159 ServiceFault

名称	类型	说明
ServiceFault	structure	发生服务级别的错误时发送的错误应答
responseHeader	ResponseHeader	通用应答参数(ResponseHeader 信息见 7.27)

7.29 SessionAuthenticationToken(会话验证令牌)

SessionAuthenticationToken 类型是一个不透明的标识符,用于标识特定会话的关联请求。这个标识符与 SecureChannelId 或客户端证书结合使用,验证传入的消息。它是 sessionId 的隐藏形式,在客户端和服务器应用程序内部使用。

服务器在创建会话应答中返回一个 SessionAuthenticationToken。客户端后续在每次请求时发送这个值,服务器可以验证请求的发送者是否与原始创建会话请求的发送者一致。

这个讨论的目的,一个服务器由应用(代码)和通信栈组成,如图 27 所示。会话认证令牌提供的安全取决于服务器程序和通信栈的信任关系。通信栈应能验证消息的发送者,到服务器他使用安全通道 Id 或客户端证书来识别发送者。在这些情况下,SessionAuthenticationToken 是一个 UInt32 标识符,允许服务器区分同一发送者创建的不同会话。

图 27 服务器逻辑分层

在某些情况下,应用和通信栈不能在运行时交换信息,这意味着应用将不能访问安全通道 ID 或用于创建安全通道的证书。在这些情况下,应用将创建一个至少有 32 字节长的随机二进制串值。这个值应保密,并始终在加密情况下通过安全通道交换。管理员负责确保加密启用。在 IEC 62541-7 的行规定义了一个会话认证令牌二进制串的附加要求。

客户端和服务器程序应编写独立的安全通道实现。因此,他们始终把会话认证令牌作为秘密信息,即使使用一些安全通道实现时它不被要求。

如图 28 所示,当客户端获取了 SessionAuthenticationToken 时,客户端、服务器和服务器通信栈之间的信息交换。在图 28 中 GetSecureChannelInfo 体现一个 API 取决于通信栈实现。

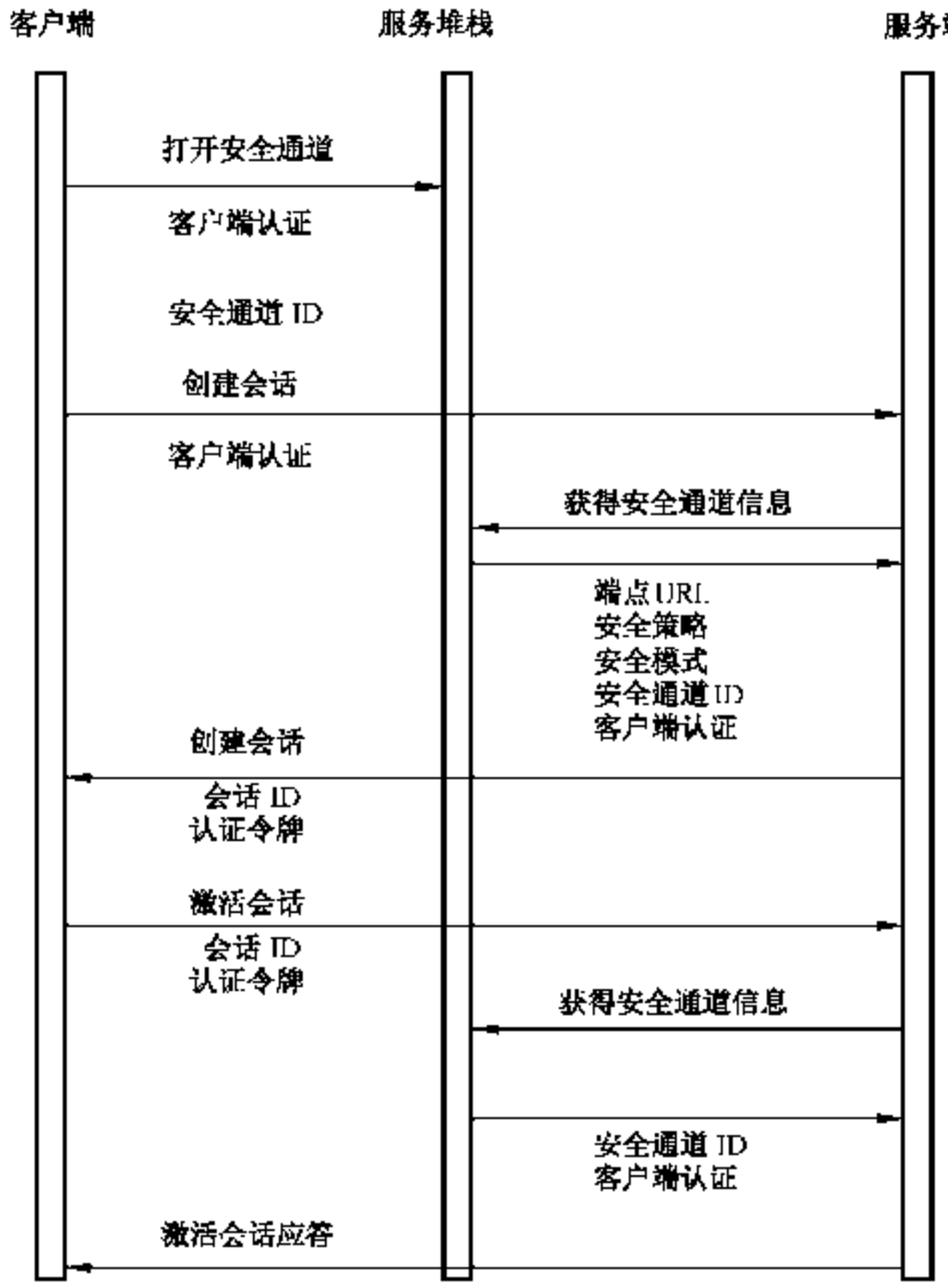

图 28　获取 SessionAuthenticationToken

SessionAuthenticationToken 是 NodeId 数据类型的一个子类型。然而，该令牌绝不用来标识地址空间中的节点。服务器可能会分配命名空间索引的一个值。其含义是服务器特定的。

7.30　SignatureData(签名数据)

此参数组成在表 160 中定义。

表 160　SignatureData

名称	类型	说明
SignatureData	structure	包含证书创建的数字签名
Signature	ByteString	这是与证书相关的私钥生成的签名
Algorithm	String	包含算法 URI 的字符串。 这个 URI 字符串值，可使用在 IEC 62541 7 中规定安全行规部分定义的名称

7.31　SignedSoftwareCertificate(签名的软件证书)

SignedSoftwareCertificate 是一个包含编码证书的二进制串。编码 SignedSoftwareCertificate 取决于安全技术映射，在 IEC 62541 6 中完全定义。表 161 规定了 SignedSoftwareCertificate 应包含的信息。

表 161 SignedSoftwareCertificate

名称	类型	说明
SignedSoftwareCertificate	structure	一个 SignedSoftwareCertificate，由发证机构创建的签名
version	String	证书编码的版本标识符
serialNumber	ByteString	由颁发者指定的证书的惟一标识符
signatureAlgorithm	String	用来签名证书的算法。 这个字段语法取决于证书编码
signature	ByteString	由颁发者创建的签名
issuer	Structure	标识用于创建签名的证书颁发者的名称
validFrom	UtcTime	证书生效时间
validTo	UtcTime	证书过期时间
subject	Structure	标识证书描述的产品的名称。 这个字段应包含来自软件证书的产品名称和供应商名称
subjectAltName	Structure	产品的备用名称列表。列表应包括软件证书里指定的产品 Uri
publicKey	ByteString	证书关联的公开密匙
keyUsage[]	String	指定可以被使用的证书密匙。 签名软件证书可能值用于创建数字签名，有不可否认性。这个字段内容取决于证书编码
softwareCertificate	ByteString	来自软件证书的 XML 编码，以 UTF8 文本存储。 IEC 62541 6 描述了软件证书的 XML 表示

7.32 SoftwareCertificate(软件证书)

该参数组成的定义见表 162。

表 162 SoftwareCertificate

名称	类型	说明
SoftwareCertificate	Structure	描述一个产品的证书
productName	String	认证产品的名称。应指定这个字段
productUri	String	认证产品的全局惟一标识符。应指定这个字段
vendorName	String	负责产品的供应商名称。应指定这个字段
vendorProductCertificate	ByteString	DER 编码形式的 X.509 证书，由供应商分配给产品。该字段可以省略
softwareVersion	String	软件版本。应指定这个字段
buildNumber	String	内部版本号。应指定这个字段
buildDate	UtcTime	构建的日期和时间。应指定这个字段
issuedBy	String	发证机构的 URI。应指定这个字段
issueDate	UtcTime	指定发证机构签发证书的时间。应指定这个字段
vendorProductCertificate	ByteString	DER 编码形式的 X.509 证书，由供应商分配给应用程序

表 162（续）

名称	类型	说明
supportedProfiles [ ]	structure	支持的行规列表
organizationUri	String	行规定义的组织的 URI 标识
profileId	String	标识行规的字符串
complianceTool	String	标识用于符合性测试的工具或认证方法的名称
complianceDate	UtcTime	符合性测试的日期和时间
complianceLevel	Enum ComplianceLevel	一个枚举指定行规里的符合性级别，它具有以下值： UNTESTED_0 功能描述未测试成功。 PARTIAL_1 功能描述已部分测试和通过严格测试，由发证机构界定。 SELFTESTED_2 功能描述由使用发证机构授权的自检系统成功测试。 CERTIFIED_3 功能描述由发证机构授权的测试组织成功地测试
unsupportedUnitIds []	String	没有测试的可选一致性单元的标识符。IEC 62541-7 进行了详细的解释

7.33 StatusCode(状态码)

7.33.1 概述

在 OPC UA 中，状态码是一个数字值，它用来报告 OPC UA 服务器执行操作的结果。该编码可能具有更详细的状态相关的诊断信息；无论如何，这些编码为客户端应用程序提供足够的信息来决定如何处理 OPC UA 的服务结果。

状态码是一个 32 位无符号整数。它的前 16 位代表编码的数值，用来检测特定错误或状态。后 16 位是标志位，包含更多的信息，但不影响状态码的意义。

所有的 OPC UA 客户端在使用状态码之前，总会先检查与之相关的结果。如果有一个不确定的/警告状态，那么与之相关的结果，应谨慎使用，因为这些结果在有些情况下可能会无效。一个坏/失败状态的结果应永远不会被使用。

如果操作完成正常，OPC UA 服务器应该返回好/成功的状态码，并且结果是有效的。不同的状态值可以向客户端提供额外的信息。

如果 OPC UA 服务器无法以客户端要求的方式完成操作，将使用不确定/警告状态码，但是操作并没有完全失败。确切的位分配如表 163 所示：

表 163 状态码位分配

字段	位范围	描述
Severity	30:31	表示状态代码是 good、bad 或 uncertain 的状态。这些位的含义如下： Good/Success:00 表示该操作是成功的，并且相关的结果可以被使用。 Uncertain/Warning:01 表示该操作部分成功，相关的结果可能不适合某些情况。 Bad Failure:10 表示操作失败并且任何相关的结果都不能使用。 Reserved:11 保留供将来使用，所有客户端把该状态代码的严重等级表示为 bad
保留	29:28	预留供将来使用，始终为零

表 163（续）

字段	位范围	描述
SubCode	16:27	该编码是一个数值被分配用来代表不同的条件，每个编码都有一个符号名和数值。OPC UA 规范中的所有描述都参考该符号名，IEC 62541-6 将符号名映射到一个数值
StructureChanged	15:15	表明自上次通知后，相关数值的结构发生了改变。客户端将不处理数据值，直至重新读取元数据。如果数据类型的编码用于变量的改变，服务器将设置这个位。7.23 中描述了如何指定变量的数据类型的编码。如果变量的数据类型属性发生变化，该位也会被设置。当数据类型为 BaseDataType 的参数变化时，该位不会被设置。如果阵列维度或价值等级属性或枚举的字符串变量的参数变化时，该位也会被设置。此位用来警告客户端，分析复杂的数据值，因为已更改的数据值的序列化形式，他们的分析例程可能会失败。该位只对返回的状态码部分数据值更改通知或历史值时有效。在其他情况下使用状态码应始终设置该位为零
SemanticsChanged	14:14	表示语义相关的数值已经改变。客户端不会处理数据值，直到重新读取与变量相关的元数据。如果元数据发生改变，而客户端不重新读取元数据将会导致应用程序错误，所以，元数据发生改变时，服务器应设置此位。 例如，当工程单位的变化后，如果客户端使用的原数值进行计算可能造成的问题。IEC 62541-8 定义了在什么条件下，服务器应为 DA 变量设置该位。其他规范可以定义附加条件。一个服务器可以定义导致该位被设置的其他条件。该位只对返回的状态码部分数据值更改通知或历史值时有效。在其他情况下用的状态代码应始终设置该位为零
保留	12:13	预留供将来使用，始终为零
InfoType	10:11	在信息位中包含了类型信息，这些位的含义如下： NotUsed:00 信息位没有被使用，始终为 0 DataValue:01 从服务器返回的相关数据值的状态码和信息位 保留:1X 预留供将来使用，信息位将被忽略
InfoBits	0:9	代表状态码的额外信息位。这些位的结构取决于信息类型字段

表 164 描述了当信息类型设置为数值(01)时，信息位的结构。

表 164　数据值信息位

信息类型	位范围	描述
LimitBits	8:9	与数据相关联的限制位，具有如下含义： 限制　　位　　描述 None　　00　　数值可随意改变 Low　　01　　该值是在数据源的下限 High　　10　　该值是在数据源的上限 Constant　　11　　该值是常量不能更改
Overflow	7	如果此位被设置，将不再返回所有被检测到的改变，直至服务器的队列缓冲区的监视项目达到极限，不得不清除数据
预留	5:6	预留供将来使用，始终为零

表 164（续）

信息类型	位范围	描述
HistorianBits	0:4	只有当读取历史数据时，才对这些位进行设置。他们表示数据值来自哪里，并提供影响客户使用数据值的信息。 历史位具有如下含义： Raw XXX00 原始数据的值 Calculated XXX01 计算数据值 Interpolated XXX10 是插值数据 预留值 XXX11 未定义 Partial XX1XX 不完整的区间计算值 ExtraData X1XXX 隐藏在其他具有相同时间戳值数据下的原始数据 Multi Value 1XXXX 多值匹配的标准(即多个集合在不同的时间戳相同的时间间隔内的最低值) IEC 62541-11 更详细介绍如何使用这些位

7.33.2 通用 StatusCode

表 165 定义了所有服务的结果的常见状态代码。IEC 62541-6 将符号名称和数值进行匹配。

表 165 通用服务结果编码

符号 ID	描述
Good	操作成功
Good_CompletesAsynchronously	将完成异步处理
Good_SubscriptionTransferred	订阅被转移到另一个进程
Bad_CertificateHostNameInvalid	连接到服务器的主机名与认证不匹配
Bad_CertificateIssuerRevocationUnknown	不确定发行证书是否有效
Bad_CertificateIssuerUseNotAllowed	发行证书，不能用于请求的操作
Bad_CertificateIssuerTimeInvalid	发行证书已过期或尚未生效
Bad_CertificateIssuerRevoked	发行证书已失效
Bad_CertificateInvalid	作为一个参数提供的证书是无效的
Bad_CertificateRevocationUnknown	无法确定证书是否有效
Bad_CertificateRevoked	证书已被吊销
Bad_CertificateTimeInvalid	证书已过期或尚未生效
Bad_CertificateUriInvalid	应用说明中的 URI 与证书中的 URI 不匹配
Bad_CertificateUntrusted	证书不被信任
Bad_CertificateUseNotAllowed	证书不能用于请求的操作
Bad_CommunicationError	低级通信发生错误
Bad_DataTypeIdUnknown	由于数据类型 ID 不确认，扩展对象不能(反)序列化
Bad_DecodingError	因为无效的数据，解码停止

表 165（续）

符号 ID	描述
Bad_EncodingError	因为在被序列化的对象的无效数据，编码停止
Bad_EncodingLimitsExceeded	通信协议栈所施加的消息编码/解码限制超出
Bad_IdentityTokenInvalid	用户身份令牌无效
Bad_IdentityTokenRejected	用户身份令牌有效，但被服务器拒绝
Bad_InternalError	内部错误导致程序或配置失败
Bad_InvalidArgument	一个或多个参数无效。每个服务定义具体参数的状态码，这些状态码用来代替一般的错误代码，此错误代码仅用于通信协议栈和有效状态码服务列表所定义的服务中
Bad_InvalidState	操作无法完成，因为对象关闭、未初始化或一些其他无效状态
Bad_InvalidTimestamp	时间戳在服务器允许的范围之外
Bad_NonceInvalid	当前似乎不是一个随机值，或者不是正确的长度
Bad_NothingToDo	无操作，因为客户端通过的操作列表没有元素
Bad_OutOfMemory	内存不足，无法完成操作
Bad_RequestCancelledByClient	请求被客户端取消
Bad_RequestTooLarge	请求消息的大小超过服务器设置限制
Bad_ResponseTooLarge	响应消息的大小超过客户端设置限制
Bad_RequestHeaderInvalid	请求信息首部丢失或无效
Bad_ResourceUnavailable	操作系统资源不可用
Bad_SecureChannelIdInvalid	指定的安全通道无效
Bad_SecurityChecksFailed	安全验证发生错误
Bad_ServerHalted	服务器已停止，无法处理任何请求
Bad_ServerNotConnected	操作无法完成，因为客户端没有连接到服务器
Bad_ServerUriInvalid	服务器 URI 无效
Bad_ServiceUnsupported	服务器不支持请求的服务
Bad_SessionIdInvalid	进程 ID 无效
Bad_SessionClosed	客户端关闭进程
Bad_SessionNotActivated	进程不能使用，因为还没有被激活
Bad_Shutdown	操作被取消，因为应用程序被关闭
Bad_SubscriptionIdInvalid	订阅 ID 无效
Bad_Timeout	操作超时
Bad_TimestampsToReturnInvalid	返回参数的时间戳无效
Bad_TooManyOperations	请求不能被处理，因为指定了太多的操作
Bad_UnexpectedError	发生意外的错误
Bad_UnknownResponse	服务器收到一个无法识别的响应
Bad_UserAccessDenied	用户没有权限执行请求的操作

表 165（续）

符号 ID	描述
Bad_ViewIdUnknown	视图 Id 未指定有效的视图节点
Bad_ViewTimestampInvalid	视图时间戳不可用或不被支持
Bad_ViewParameterMismatchInvalid	视图参数相互不一致
Bad_ViewVersionInvalid	视图版本不可用或不被支持

表 166 定义了所有层次的操作结果的常见状态代码。IEC 62541-6 将符号名称和数值进行匹配。常见的服务结果编码，也可以包含在操作层次中。

表 166　常见的操作层次的结果编码

符号 ID	描述
Good_Clamped	写的值被接受，但被限制
Good_Overload	由于资源限制，采样频率降低
Uncertain	该值不确定，但具体原因不明
Bad	该值是坏的，但具体原因不明
Bad_AttributeIdInvalid	指定的节点不支持该属性
Bad_BrowseDirectionInvalid	浏览方向无效
Bad_BrowseNameInvalid	浏览名称无效
Bad_ContentFilterInvalid	内容过滤器是无效的
Bad_ContinuationPointInvalid	提供的延续点失效
Bad_DataEncodingInvalid	数据编码是无效的
Bad_DataEncodingUnsupported	服务器不支持的节点请求数据编码
Bad_EventFilterInvalid	事件过滤器是无效的
Bad_FilterNotAllowed	监测过滤器不能用于指定属性
Bad_FilterOperandInvalid	内容过滤器所使用的操作是无效的
Bad_HistoryOperationInvalid	历史的细节参数无效
Bad_HistoryOperationUnsupported	服务器不支持请求的操作
Bad_IndexRangeInvalid	该索引范围参数的语法是无效的
Bad_IndexRangeNoData	没有数据存在指定索引的范围内
Bad_MonitoredItemFilterInvalid	监测项目的过滤参数是无效的
Bad_MonitoredItemFilterUnsupported	服务器不支持所要求的监测项目的过滤器
Bad_MonitoredItemIdInvalid	监测项目标识未指定有效的监测项
Bad_MonitoringModeInvalid	监视模式是无效的
Bad_NoCommunication	与数据源的通信已定义，但未建立，并且没有最后已知值。 此状态/子状态被用作收到的第一个值前的缓存值
Bad_NoContinuationPoints	操作无法处理，因为所有的延续点已分配

表 166（续）

符号 ID	描述
Bad_NodeClassInvalid	节点类无效
Bad_NodeIdInvalid	节点 ID 的语法是无效的
Bad_NodeIdUnknown	指定的节点 ID 在服务器的地址空间中不存在的
Bad_NoDeleteRights	服务器不允许节点被删除
Bad_NodeNotInView	要浏览的节点不是视图的一部分
Bad_NotFound	要求的项目没有被发现或搜索操作没有成功结束
Bad_NotImplemented	请求的操作未执行
Bad_NotReadable	访问级别不允许读取或订阅节点
Bad_NotSupported	不支持请求的操作
Bad_NotWritable	访问级别不允许对节点的写操作
Bad_ObjectDeleted	对象已经被删除，无法使用
Bad_OutOfRange	该值超出范围
Bad_ReferenceTypeIdInvalid	引用类型的 ID 并不是指一个有效的引用类型的节点
Bad_SourceNodeIdInvalid	节点的 ID 并不是一个有效的节点
Bad_StructureMissing	强制性的结构参数丢失或空
Bad_TargetNodeIdInvalid	目标节点的 ID 不是一个有效的节点
Bad_TypeDefinitionInvalid	该类型定义的节点 ID 没有引用适当类型的节点
Bad_TypeMismatch	为属性提供的值是不相同类型的属性值
Bad_WaitingForInitialData	等待服务器，以获得底层数据源的值。 创建一个监视项后，服务器可能需要一些时间为这些项获取真实值。在这种情况下，服务器可以选择在第一个有效值的通知之前发送此通知

7.34 TimestampToReturn（返回时间戳）

TimestampToReturn 值是一个枚举变量，它在指定的监测项目或历史节点读取中传送时间戳属性。这个参数的值在表 167 中定义。

表 167 TimestampToReturn 值

数值	描述
SOURCE_0	返回源的时间戳；在读历史中使用时，源时间戳是用来确定返回哪些历史数据值
SERVER_1	返回服务器时间戳；在读历史中使用时，服务器时间戳是用来确定返回哪些历史数据值
BOTH_2	返回源和服务器的时间戳；在读历史中使用时，源时间戳是用来确定返回哪些历史数据值
NEITHER_3	不返回时间戳。如果没有变量的值，这作为监视项的缺省值。对于读历史，这是无效设置

7.35 UserIdentityToken(用户标识令牌)参数

7.35.1 概述

在服务器服务中设置使用的 UserIdentityToken 结构允许客户端指定代表其身份的用户。用于识别用户的具体机制取决于系统配置。不同类型的身份令牌基于当前系统中最常使用的机制。表 168 定义了当前用户的身份令牌。在 7.11 中定义了可扩展的参数类型。

表 168 TimestampToReturn 参数类型 ID

符号 ID	描述
AnonymousIdentityToken	无用户的信息可用
UserNameIdentityToken	由用户名和密码定义的用户
X509IdentityToken	由 X509v3 证书定义的用户
IssuedIdentityToken	由 WS-Security 令牌定义的用户

当传递到服务器时,客户端应提供拥有 UserIdentityToken 的证明。一些令牌包括机密信息,如密码,服务器将接受作为证据。为了保护这些机密信息,它被传递给服务器之前,令牌需要加密。其他类型的标记允许客户端创建令牌的密令签名。在这些情况下,客户通过追加最后服务器临时的服务器证书证明持有 UserIdentityToken;并使用密令产生一个传递到服务器上的签名。

每个端点允许的 UserIdentityToken,应当有一个端点描述中指定的用户令牌政策。当加密或签名时,用户令牌策略规定使用什么样的安全策略。如果省略此安全政策,客户端在端点描述时使用安全策略。如果匹配的安全策略设置为 None,将不需要加密或签名。建议用户不要把用户令牌的安全策略设置为 None,因为这些秘密可能被攻击者用来获得对系统的访问。

表 169 描述了如何在申请加密前序列化 UserIdentityToken。

表 169 UserIdentityToken 加密令牌格式

名称	类型	描述
length	Byte[4]	加密的数据的长度包括临时服务器,但不包括长度域本身。此字段是 4 字节无符号整数,低字节在前
tokenData	Byte[*]	令牌数据
serverNonce	Byte[*]	最后一个临时服务器,由创建进程或激活进程响应的服务器返回

7.35.2 AnonymousIdentityToken(匿名标识令牌)

AnonymousIdentityToken 是用来表明客户端有没有用户凭据。表 170 定义了 AnonymousIdentityToken 参数。

表 170 AnonymousIdentityToken

名称	类型	描述
AnonymousIdentityToken	Structure	一个匿名用户身份
policyId	String	令牌符合的用户令牌策略标识符。用户令牌策略结构定义见 7.36

7.35.3 UserNameIdentityToken(用户名称标识令牌)

UserNameIdentityToken 用于通过简单的用户名/密码凭据到达服务器。如果安全策略有要求,此令牌应当加密。服务器应指定一个用户令牌政策安全策略,如果安全通道有一个 None 安全政策。如果加密令牌,密码应转换到 UTF8 字节串然后序列化,如表 169 所示。服务器应当对密码解密,并且验证临时服务器。如果安全策略是 None,那么密码只包含 UTF-8 编码的密码。表 171 定义了 UserNameIdentityToken 参数。

表 171 UserNameIdentityToken

名称	类型	描述
UserNameIdentityToken	structure	用户名值
policyId	String	用户令牌符合的令牌策略标识符。用户令牌策略的结构在 7.36 中定义
userName	String	一个字符串,标识用户
password	ByteString	用户的密码,此参数与服务器证书应使用安全策略中指定的算法加密
encryptionAlgorithm	String	一个字符串,其中包含的加密算法的 URI。URI 字符串值是被定义的名称,它可用于在 IEC 62541 7 规定的安全配置的一部分。如果密码未加密,这个参数是空的

7.35.4 X509IdentityToken(X509 标识令牌)

X509IdentityToken 用来传递由用户发出 X509v3 证书。如果安全策略有要求,签名应一直伴随该令牌。服务器应为用户令牌策略指定一个安全策略,如果安全通道有一个 None 安全政策。表 172 定义了 X509IdentityToken。

表 172 X509IdentityToken

名称	类型	描述
X509IdentityToken	structure	X509 值
policyId	String	与令牌符合的用户令牌策略标识符,用户令牌政策结构定义见 7.36
certificateData	ByteString	X509 证书为 DER 格式

7.35.5 IssuedIdentityToken(已发布标识令牌)

IssuedIdentityToken 用于传递符合 WS-Security 的安全令牌到服务器。

WS-Security 定义了一些可能被用来代表不同类型的安全令牌的令牌资料。例如,Kerberos 和 SAML 令牌有 WSS 的标记配置文件,并可以在 OPC UA 中作为 XML 安全性令牌交换。

WSS X509 和用户名标记不能作为 XML 安全令牌交换。OPC UA 的应用程序应该使用适当的 OPC UA 的身份令牌在 WSS 安全令牌类型中传递信息。

这些令牌可能是加密或要求签名的。IEC 62541 7 定义了包括用户安全相关的配置文件,如果需要签名,他们还包括加密或签名的任何要求。额外的安全配置文件指定加密和签名算法。

如果令牌被加密,XML 将转换为 UTF8 字节串,然后序列化,如表 169 所示。

服务器应对令牌数据进行解密并验证临时服务器。

如果安全策略是“None”或如果令牌只需要签名，用 UTF8 加密的 XML 令牌数据就表示该令牌。

表 173 定义了 IssuedIdentityToken 参数。

表 173 IssuedIdentityToken

名称	类型	描述
UserNameIdentityToken	structure	WSS 值
policyId	String	与令牌符合的用户令牌策略标识符，用户令牌政策结构定义见 7.36
userName	String	令牌的 XML 表示，这个参数可以用服务器的证书加密
password	ByteString	一个字符串，其中包含加密算法的 URI。可能被用到的 OPCUA 的定义的名称的列表，在 IEC 62541-7 中被指定。如果令牌不加密，该参数是空值

7.36 UserTokenPolicy(用户令牌策略)

此参数组件在表 174 中定义。

表 174 UserTokenPolicy

名称	类型	描述
UserTokenPolicy	structure	指定服务器会接受的用户身份标记
policyId	String	由服务器分配的为用户令牌策略的标识符。当它构建了一个符合策略的用户的身份标记时，客户端指定此值。这个值在一台服务器中是独有的
tokenType	Enum UserIdentity TokenType	用户类型标识要求的令牌 此值是下列值之一的枚举类型值： ANONYMOUS_0 不需要令牌。 USERNAME_1 用户名/密码令牌。 CERTIFICATE_2 一个 X509v3 证书令牌。 ISSUEDTOKEN_3 任何的 WS Security 定义令牌。 令牌类型为 ANONYMOUS 时，服务器不需要任何用户识别。在这种情况下，客户端常用证书作为用户识别
issuedTokenType	String	令牌类型的 URI。IEC 62541 7 定义的 URI 为通用令牌类型。供应商可能会指定自己的令牌。这一领域可能只被指定如果是 ISSUEDTOKEN TokenType。WS Security 的令牌有时由 XML QualifiedNames 确定的。一个为 URI 令牌可以构造附加的名称命名空间“:”分隔符。可重构的 XML QualifiedName 搜索最新的“:”分隔符
issuerEndpointUrl	String	一个可选的 URL 为令牌发证服务。 这个值的含义取决于所发出的令牌类型
securityPolicyUri	String	该安全策略用在当通过服务器活动进程请求加密或签名时。7.35 介绍了如何使用此参数，如果省略此值，使用安全通道的安全策略

7.37 ViewDescription(视图描述)

此参数组件在表 175 中定义。

表 175 ViewDescription

名称	类型	描述
ViewDescription	structure	指定一个视图
viewId	NodeId	节点 ID 的查询视图。null 值指示整个地址空间
timestamp	UtcTime	所需的时间和日期。 相应的版本是最接近的先前创建时间戳之一。时间戳或视图版本参数可以用客户端设置,但不能两者兼得。如果视图版本被设置此参数应为空
viewVersion	UInt32	所需的视图版本号。当节点加入或从视图中删除,浏览视图版本属性的值将被更新。时间戳或查看版本参数可以通过客户端设置,但不能两者兼得。视图版本属性被定义在 IEC 62541-3 中。如果时间戳被设置,此参数应为 0。如果时间戳是空的,视图版本是 0,那么将使用当前视图

附 录 A
（资料性附录）
BNF 定义

A.1 BNF 概述

在本附录中使用的 BNF(巴科斯诺尔表示式)中，使用“<”和“>”标记符号，“[”和“]”表示可选择的路径，“ ”表示可替换项。如果使用“(”和“)”符号，表示设置。

A.2 相对路径的 BNF

相对路径(RelativePath)是一种结构，用来描述随后的引用序列和节点序列。本附录描述了一个能够在文档或文件中存储配置信息的相对路径文本格式。

相对路径的各组成部分的文本格式的项目说明见表 A.1。

表 A.1 相对路径

符号	含义
/	斜杠字符表示服务器后面是子类型的分层引用(HierarchicalReferences)
.	句点字符表示服务器后面是子类型的集合引用类型(Aggregate ReferenceType)
<[#！ ns:]ReferenceType>	被“<”和“>”界定的字符串指定了后面跟着引用类型(ReferenceType)的浏览名称(BrowseName)。缺省情况下，引用类型(ReferenceType)的子类型的任意引用(References)也跟在后面。浏览名称(BrowseName)前面有一个“#”表示后面不跟随子类型。浏览名称(BrowseName)前面有一个“!”表示后面为反向引用(inverse Reference)。 浏览名称(BrowseName)可以用命名空间索引来指定(用数字前缀后面加一个冒号来表示)。这个命名空间索引用来指定引用类型(ReferenceType)的浏览名称(BrowseName)的命名空间组成。如果命名空间前缀省略，那么命名空间索引缺省为 0
[ns:]BrowseName	跟在“/”、“.”或者“>”符号后面的字符串指定了一个返回或跟随的目标节点的浏览名称。这个浏览名称可以将其命名空间索引作为前缀。如果命名空间前缀省略，那么命名空间索引缺省为 0。 从一个路径上省略最后的浏览名称等价于一个通配符操作，匹配所有 Nodes，路径指定的引用目标
&	符号“&”是转义字符。它用于描述出现在浏览名称中的保留字符。一个保留字符在前面插入“&”就可以转义使用了。下面是带有转义字符的浏览名称的例子： 接收到的浏览路径名称 解析为 "&/Name_1" "/Name_1" "&.Name_2" ".Name_2" "&:Name_3" ":Name_3" "&&Name_4" "&Name_4"

表 A.2 提供了使用文本格式指定相对路径的例子。

表 A.2 相对路径例子

浏览路径	描述
"/2:Block&.Output"	后面跟着浏览名称(BrowseName)为"2:Block.Output"的任意正向分层引用(forward hierarchical Reference)
"/3:Truck.0:NodeVersion" "<1:ConnectedTo>1:Boiler/1:HeatSensor"	后面跟着浏览名称(BrowseName)为"/3:Truck"的任意正向分层引用(forward hierarchical Reference),并且,从那里引出一个正向集合引用(forward Aggregates Reference)指向浏览名称(BrowseName)为"0:NodeVersion"的目标 后面跟着浏览名称(BrowseName)为"1:ConnectedTo"的任意正向引用(forward Reference),并查找到浏览名称(BrowseName)为"1:Boiler"的目标。从那里后面跟着任意分层引用(hierarchical References)并查找到浏览名称(BrowseName)为"1:HeatSensor"的目标
"<1:ConnectedTo>1:Boiler/"	后面跟着浏览名称(BrowseName)为"1:ConnectedTo"的任意正向引用(forward Reference),并查找到浏览名称(BrowseName)为"1:Boiler"的目标。从那里可以查找到分层引用(hierarchical References)的所有目标
"<0:HasChild>2:Wheel" "<! HasChild>Truck"	后面跟着浏览名称(BrowseName)为"HasChild"且采用缺省 OPC UA 命名空间的任意正向引用(forward Reference)。这样可以查找到浏览名称(BrowseName)为"Wheel"且命名空间索引为"2"的目标 后面跟着浏览名称(BrowseName)为"HasChild"(例如,后面跟着 HasParent 引用)的任意反向引用(inverse Reference)。这样可以查找到浏览名称(BrowseName)为"Truck"的目标。在两种情况下,浏览名称(BrowseName)的命名空间构成都假定是 0
"<0:HasChild>"	查找到所有浏览名称(BrowseName)为"HasChild"且采用缺省 OPC UA 命名空间的正向引用(forward Reference)目标

下面的 BNF 描述了相对路径文本格式的语法。

<relative-path> ::= <reference-type> <browse-name> [relative-path]

<reference-type> ::= '/' '.' | '<' ['=' '!'] <browse-name> '>'

<browse name> ::= [<namespace index> ':'] <name>

<namespace-index> ::= <digit> [<digit>]

<digit> ::= '0' | '1' | '2' | '3' '4' | '5' | '6' | '7' '8' | '9'

<name> ::= (<name-char> | '&' <reserved-char>) [<name>]

<reserved char> ::= '/' | '.' '<' | '>' | ':' | '=' | '!' | '&'

<name-char> ::=除了保留字之外的所有有效的字符串字符(见 IEC 62541-3)。

A.3 数字范围的 BNF

下面的 BNF 描述了数值范围(NumericRange)参数类型的语法。

<numeric-range> ::= <dimension> [',' <dimension>]

<dimension> ::= <index> [':' <index>]

<index> ::= <digit> [<digit>]

<digit> ::= '0' | '1' | '2' | '3' '4' | '5' | '6' | '7' '8' | '9'

附 录 B
（资料性附录）
内容过滤器和查询的例子

B.1 简单的内容过滤器例子

B.1.1 概述

这些例子提供了非常简单的内容过滤器。和这些例子相似的过滤器可以用于过程事件。

下面这些关于属性运算对象(这些运算对象的定义见7.4.4)如何使用的约定应用于这些例子。

——属性运算符:对于一个节点来说,一个节点的属性或者与节点相关联的一个特性的值属性。在例子中,用NodeId的字符名称代替真正的NodeId,属性的ID也如此。

——相对路径的字符串表述用来代替真实的结构。

所有例子中的命名空间索引都是12(这样和使用4、23或其他任意值一样简单)。想获得关于命名空间索引的更多信息,请参阅IEC 62541-3。命名空间索引的使用阐明了例子里面使用的信息模型并不是标准定义的模型,而是为了这些例子创建的。

B.1.2 例子1

例如,"(((AType.A = 5) or InList(BType.B,3,5,7)) and BaseObjectType.displayName LIKE "Main%")"所描述的逻辑将会产生一个逻辑树,如图B.1所示,和一个内容过滤器,如表B.1所示。在这个例子中,如果想要返回有用的东西,AType和BType都必须是BaseObjectType的子类型,否则,"And"运算的结果永远为"假"(false)。

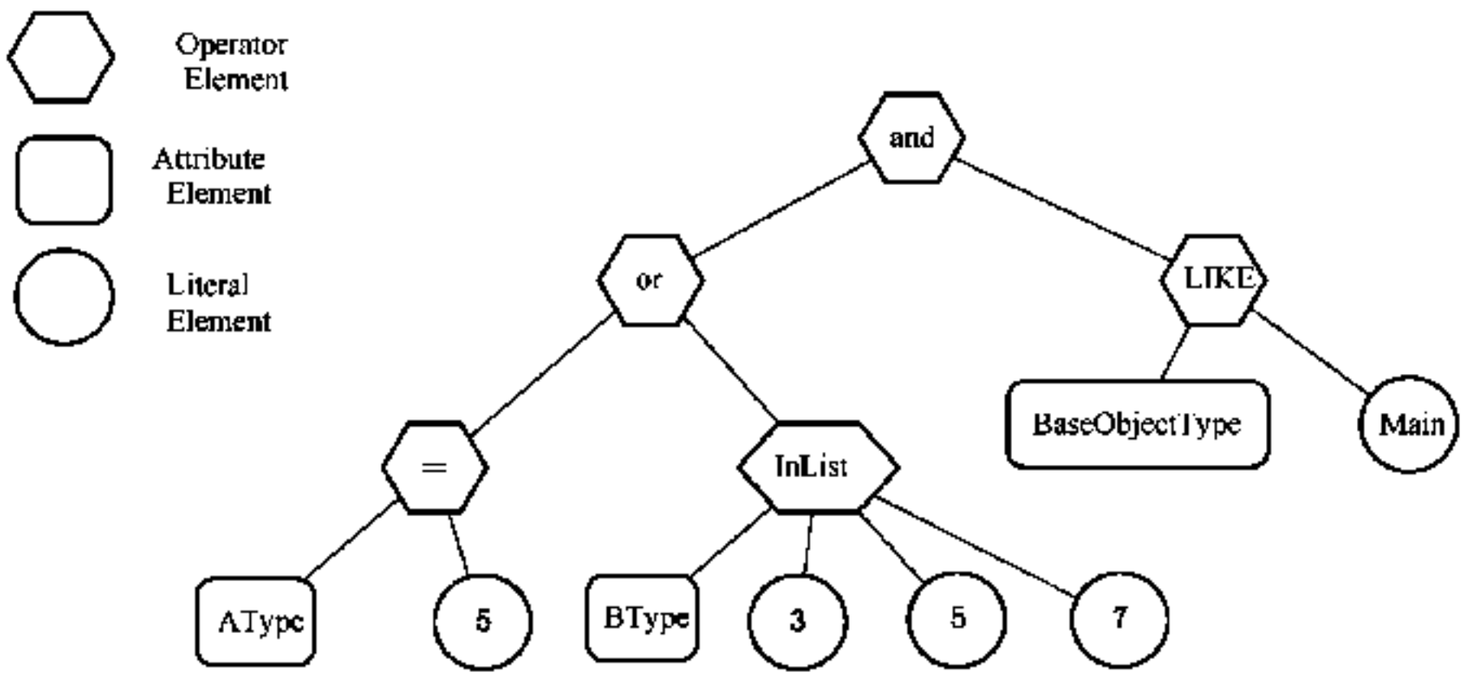

图 B.1 过滤器逻辑树例子

表B.1描述了例子中使用到的元素、运算符和操作数。

表 B.1 内容过滤器例子

元素[]	操作符	运算对象[0]	运算对象[1]	运算对象[2]	运算对象[3]
0	And	ElementOperand = 1	Element Operand = 4		
1	Or	ElementOperand = 2	Element Operand = 3		
2	Equals	AttributeOperand = NodeId: AType, BrowsePath:". 12: A",AttributeId:value	LiteralOperand = '5'		
3	InList	AttributeOperand = NodeId: BType,BrowsePath:".12:B", AttributeId:value	LiteralOperand '3'	LiteralOperand '5'	LiteralOperand '7'
4	Like	AttributeOperand NodeId: BaseObjectType, Browse Path:".", AttributeId: displayName	LiteralOperand = "Main%"		

B.1.3 例子 2

再举一个例子,选择包含在区域 1 或区域 2 的视野(View)里面的所有系统事件(包括派生类型)的过滤器,将会产生一个逻辑树,如图 B.2 所示,还会产生一个内容过滤器,如表 B.2 所示。

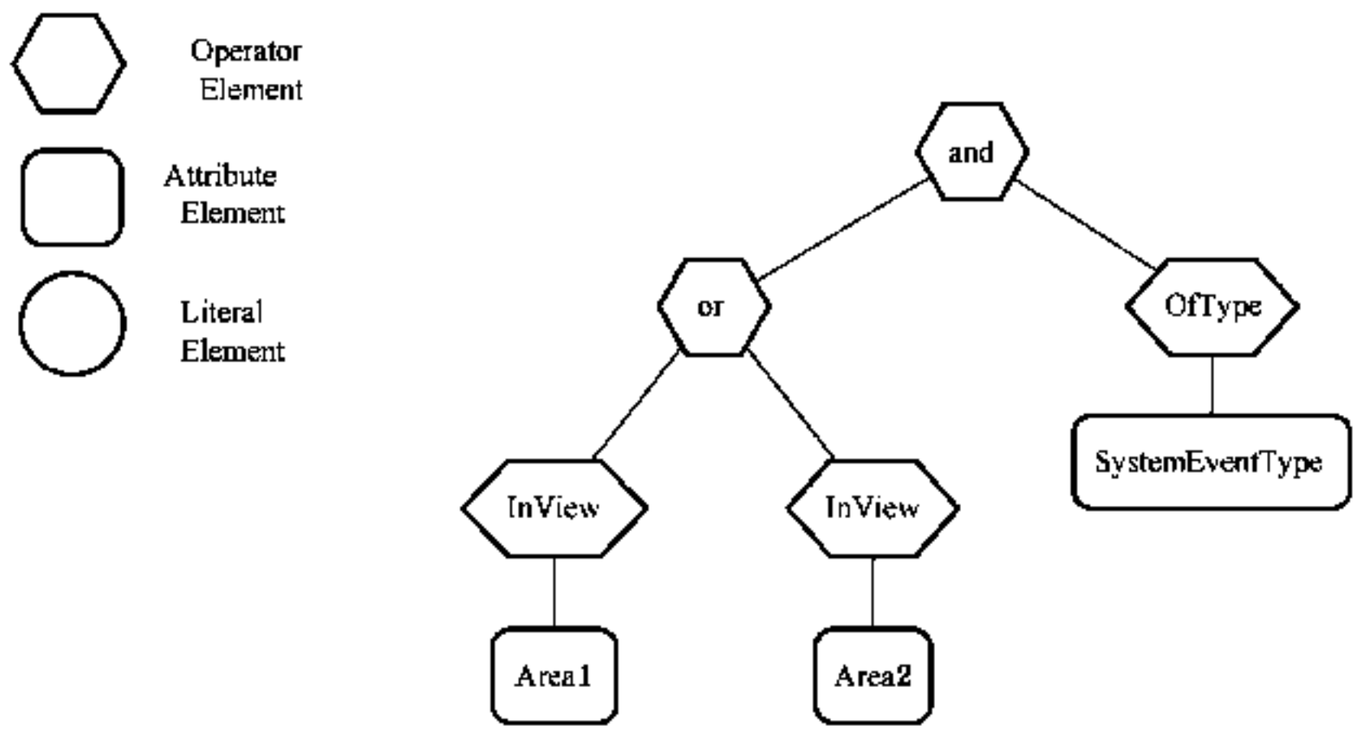

图 B.2 过滤器逻辑树例子

表 B.2 描述了例子中的元素、运算符和操作数。

表 B.2 内容过滤器例子

元素[]	运算符	运算对象[0]	运算对象[1]
0	And	ElementOperand 1	ElementOperand 4
1	Or	ElementOperand 2	ElementOperand 3
2	InView	AttributeOperand = NodeId: Area1, BrowsePath:".", AttributeId:NodeId	
3	InView	AttributeOperand = NodeId: Area2, BrowsePath:".", AttributeId:NodeId	
4	OfType	AttributeOperand NodeId: SystemEventType, BrowsePath:".", AttributeId:NodeId"	

B.2 内容过滤器(查询)的复杂例子

B.2.1 概述

这些查询例子展示了复杂的内容过滤器。下面这些关于属性运算对象的约定应用于这些例子(这些运算对象的定义参见 7.4.4)。

——属性运算对象(AttributeOperand):对于一个节点来说,一个节点的属性或者与一个节点关联的一个特性的属性值。在例子中,用 NodeId 的字符名称代替真正的 NodeId,属性的 ID 也如此。

——相对路径的字符串表述用来代替真实的结构。

——所有例子中的命名空间索引都是 12(这样和使用 4、23 或其他任意值一样简单)。想获得关于命名空间索引的更多信息,请参阅 IEC 62541-3。命名空间索引的使用阐明了例子里面使用的信息模型不是这个标准定义的模型,而是为了这些例子创建的。

B.2.2 使用类型模型

例子中使用的类型模型在下面描述:

新引用类型:

"HasChild"派生于 HierarchicalReference。

"HasAnimal"派生于 HierarchicalReference。

"HasPet"派生于 HasAnimal。

"HasFarmAnimal"派生于 HasAnimal。

"HasSchedule"派生于 HierarchicalReference。

PersonType 派生于 BaseObjectType ,同时具有如下特性:

"LastName";

"FirstName";

"StreetAddress"

"City";

"ZipCode"。

可能有 HasChild 引用 PersonType 节点。

可能有 HasAnimal 引用 AnimalType(或其子类型)节点。

AnimalType 派生于 BaseObjectType,同时具有如下特性:

"Name"

可能有 HasSchedule 引用 FeedingScheduleType 节点。

DogType 派生于 AnimalType,同时具有如下特性:

"NickName";

"DogBreed";

"License"。

CatType 派生于 AnimalType,同时具有如下特性:

"NickName";

"CatBreed"。

PigType 派生于 AnimalType,同时具有如下特性:

"PigBreed"

ScheduleType 派生于 BaseObjectType,同时具有如下特性:

"Period"

FeedingScheduleType 派生于 ScheduleType,同时具有如下特性:

"Food";

"Amount"。

AreaType 派生于 BaseObjectType,它是一个简单的文件夹,同时不包含任何特性。

该类型系统实例,如图 B.3 所示。该图中,所有对象类型的引用,对象特性和对象子类型都使用了 OPC UA 的表示方法。此外,类型所支持的引用通过内部的矩形框进行表示。真正的引用只存在实例中,因此,在图中,没有到另一个对象的连线,同时,他们可能是所列引用的子类型。

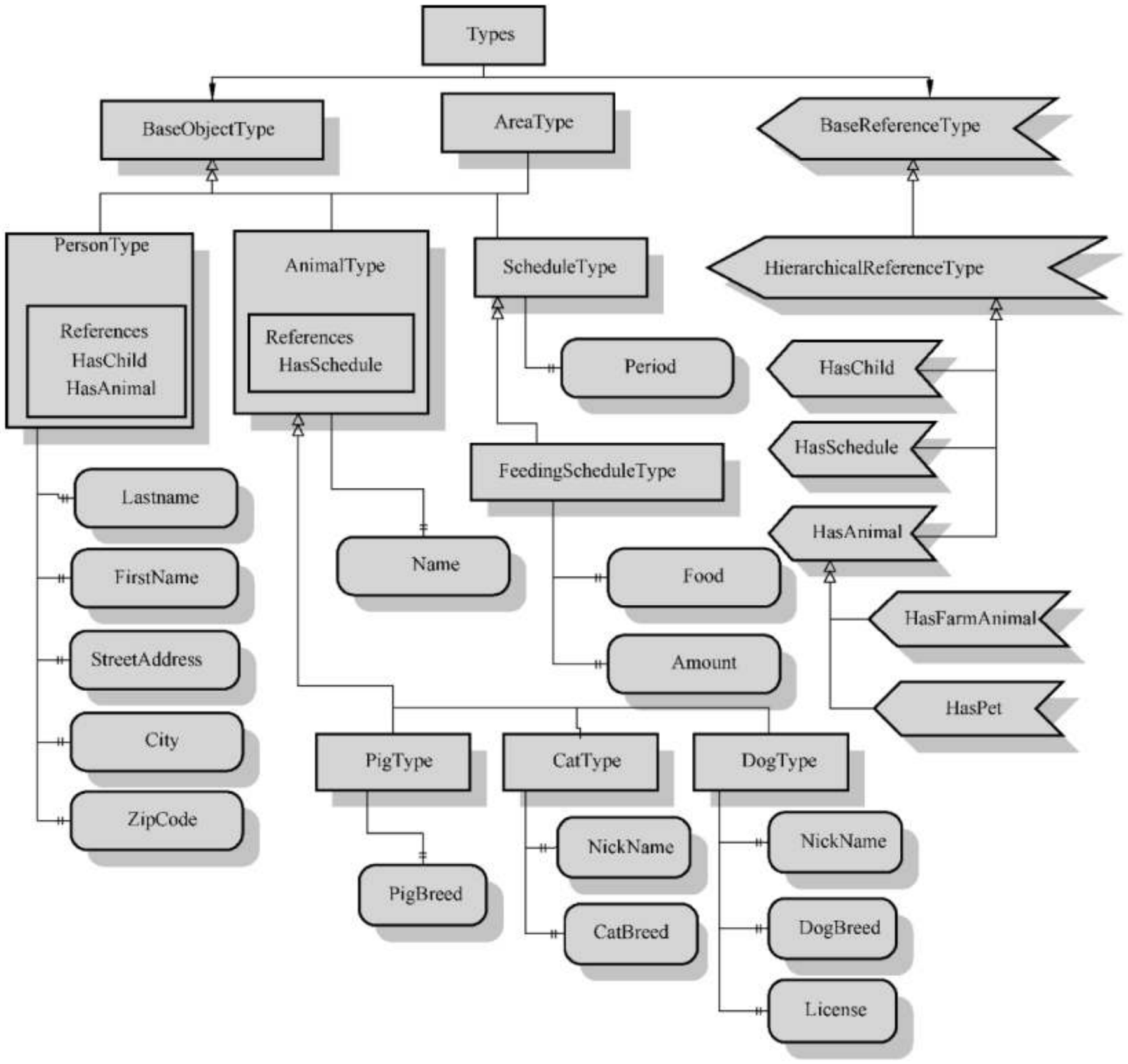

图 B.3　节点类型例子

对应的实例集例子如图 B.4 所示。这些实例包括一个对象的类型引用。属性也有类型引用，但是为了简单，这个引用被省略。矩形框中名称表示对象的名称，括号中的数字表示 NodeId。使用 OPC UA 方法定义类型引用，自定义的引用类型使用 Reference 名称表示。图中表示了 BrowseName、NodeId 和 Value 属性。结点都用彩色矩形框围着。两区域的结点都包含在一个已经存在的 person 节点组中。所有自定义结点都在命名空间 12 中，没有包括在图 B.4 中。

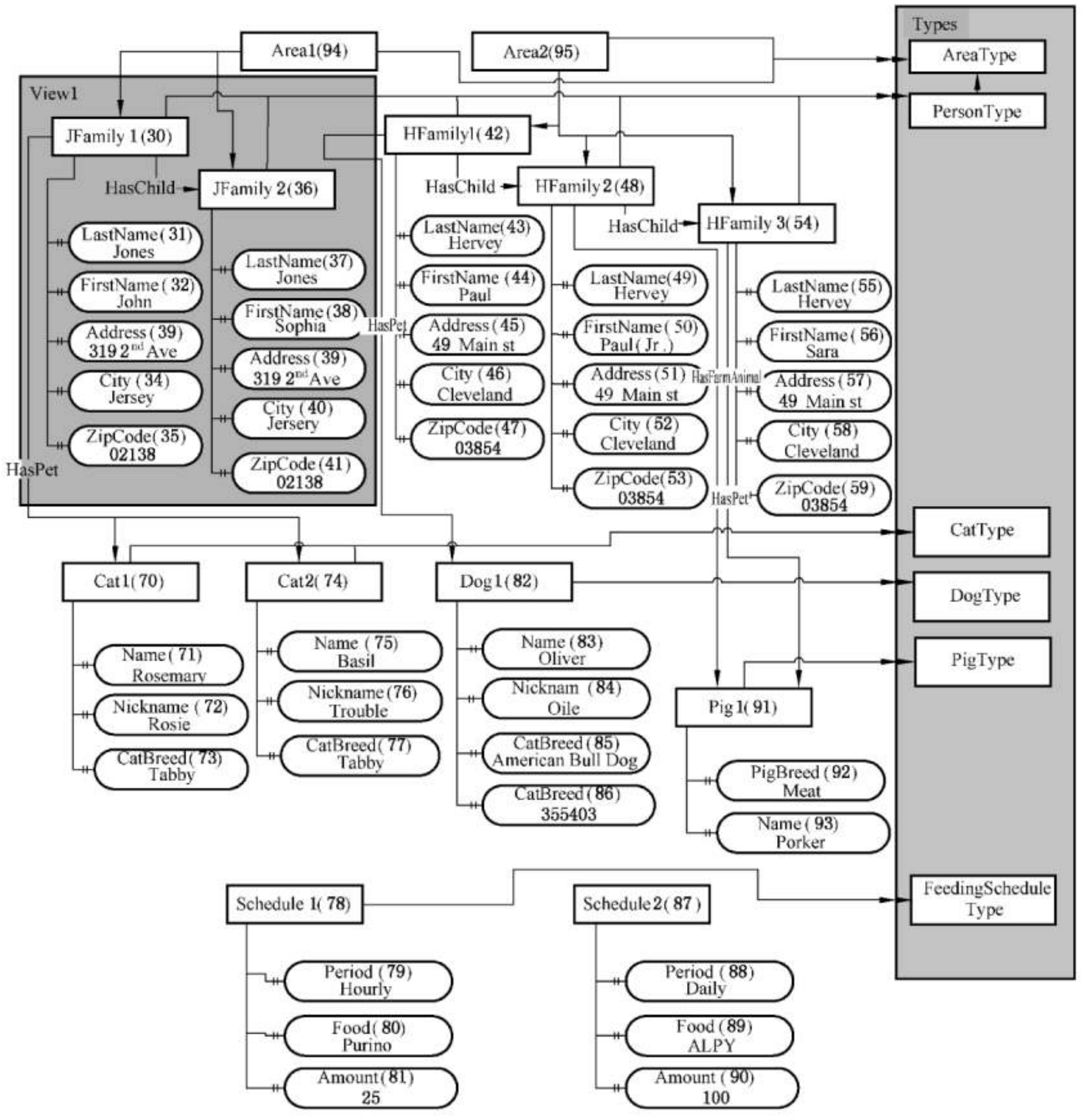

图 B.4　节点实例例子

B.2.3　实例

在 7.4.4 的所有实例中，所定义的节点类型以符号表的形式进行列举，在实际的调用中，它将分配一个 NodeId 给一个节点。属性 Id 也是一个属性的符号名称，在实际调用中，它也转换成用一个整数 Id 表示。同时，为了简洁，在所有的例子中，BrowseName 都包含在结果表中，通常其不会被返回。

这些例子都假设命名空间 12 是所有自定义描述的命名空间。

B.2.4　例 1

这个例子要求一个简单的层过滤，一个人拥有一个宠物，并且该宠物具有时间表。

例 1：从一个人可以获得 PersonType.lastName，AnimalType.name，ScheduleType.period，其中这个人有个一宠物同时这个宠物有一个时间表。

例子中的节点类型描述的参数如表 B.3 所示。

表 B.3　例 1 节点类型描述

类型定义节点	包含子类型	相对路径	属性 ID	索引范围
PersonType	否	".12:LastName"	值	不可用
		"<12:HasPet>12:AnimalType.12:name"	值	不可用
		"<12:HasPet>12:AnimalType<12:HasSchedule>12:Schedule.12:period"	值	不可用

对应的内容过滤如图 B.5 所示。

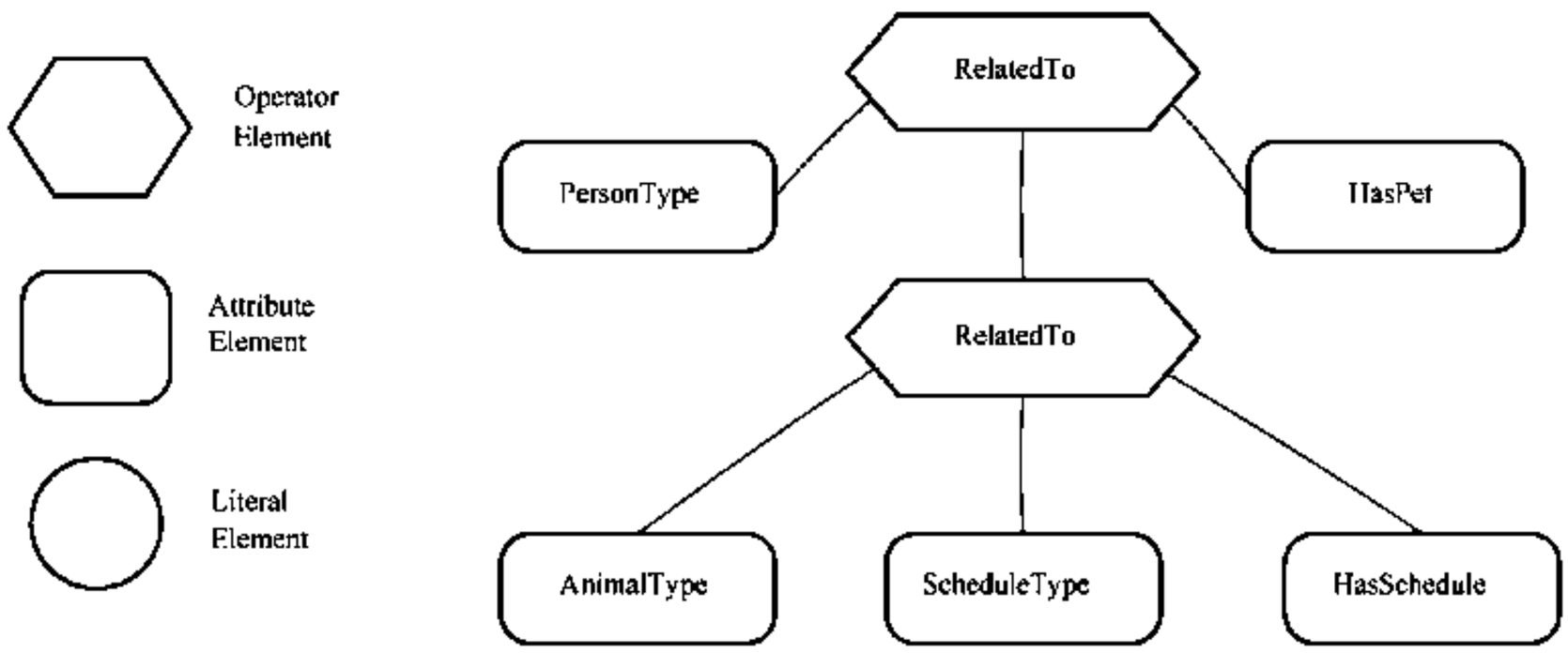

图 B.5　例 1 过滤器

表 B.4 中描述了在例子中使用的内容过滤元素、运算符和操作数。

表 B.4　例 1 内容过滤

元素	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
1	RelatedTo	AttributeOperand=Nodeid:PersonType.BrowsePath"."AttributeId:NodeId	ElementOperand=2	AttributeOperand—NodeId:HasPet.BrowsePath"."AttributeId:NodeId	LiteralOperand=1
2	RelatedTo	AttributeOperand Nodeid:PersonType.BrowsePath"."AttributeId:NodeId	AttributeOperand NodeId:ScheduleType.BrowsePath "."AttributeId:NodeId	AttributeOperand NodeId:HasSchedule.BrowsePath"."AttributeId:NodeId	LiteralOperand=1

表 B.5 中描述了，当执行图 B.4 中实例时，所返回的查询结果集 QueryDataSet。

表 B.5　例 1 查询结果集

NodeId	类型 NodeId	相对路径	值
12:30(JFamily1)	PersonType	".12:lastName"	Jones
		"<12:HasPet>12:AnimalType.12:name"	Rosemary
			Basil
		"<12:HasPet>12:AnimalType<12:HasSchedule> 12:Schedule.12:period"	Hourly
			Hourly
12:42(HFamily1)	PersonType	".12:lastName"	Hervey
		"<12:HasPet>12:AnimalType.12:name"	Olive
		"<12:HasPet>12:AnimalType<12:HasSchedule> 12:Schedule.12:period"	Daily

值列是每个节点描述数组，数组中元素顺序是与节点类请求元素对应的顺序。此外，如果单个属性有多个值，它将在一个更大的数组中返回。例如表中 Rosemary 和 Basil 将在数组<hasPet>.AnimalType.name 元素中返回。为了表达的清晰，它们分别在单独行中表示。

注：相对路径列和浏览名(NodeId 列中括号内描述)并不是查询结果集中内容，仅仅只是为了表示的清晰。TypeDefinitionNodeId 是一个整型的 NodeId，而不是表中的符号名称。

B.2.5　例 2

第 2 例子描述了正在接收的不相交节点列表，同时也描述了一个被接收的结果数组。

例 2：从一个人可以获得 PersonType.lastName，AnimalType.name，这个人有一个孩子(或者他有一个宠物猫，这只猫有一个喂食的时间表)。

例子中用到的节点类型描述的参数如表 B.6 所示。

表 B.6　例 2 节点类型描述

类型定义节点	包含子节点	相对路径	属性 ID	索引范围
PersonType	否	".12:LastName"	值	不可用
AnimalType	是	".12:name"	值	不可用

对应的内容过滤描述如图 B.6 所示。

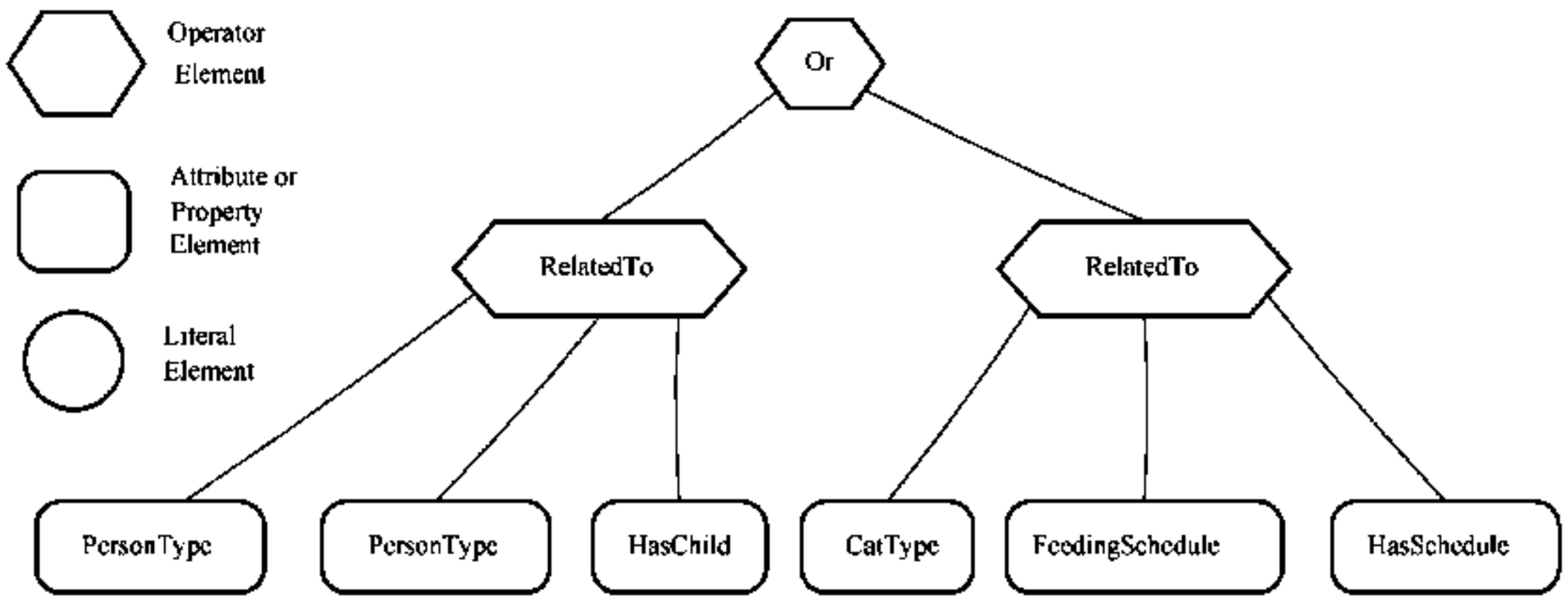

图 B.6　例 2 过滤器

表 B.7 中描述了例子中使用的元素、运算符和操作数。值得注意的是一个 Cattype 是 AnimalType-Cattype 的一个子类。

表 B.7 例 2 内容过滤

元素[ ]	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
0	Or	ElementOperand=1	ElementOperand=2		
1	RelatedTo	AttributeOperand NodeId:PersonType, BrowsePath".", AttributeId:NodeId	AttributeOperand=NodeId: PersonType,BrowsePath ".",AttributeId:NodeId	AttributeOperand NodeId: HasChild, BrowsePath ".", AttributeId:NodeId	LiteralOperand "1"
2	RelatedTo	AttributeOperand= NodeId:CatType, BrowsePath".", AttributeId:NodeId	AttributeOperand=NodeId: FeedingScheduleType, BrowsePath".",AttributeId: NodeId	AttributeOperand = NodeId: HasSchedule, BrowsePath ".", AttribteId:NodeId	LiteralOperand="1"

表 B.8 中描述了查询的结果包含了查询数据集。

表 B.8 例 2 查询结果集

NodeId	类型定义 NodeId	相对路径	值
12:30(Jfamily1)	PersonType	.12:lastName	Jones
12:42(HFamily1)	PersonType	.12:lastName	Hervey
12:48(HFamily2)	PersonType	.12:lastName	Hervey
12:70(Cat1)	CatType	.12:name	Rosemary
12:74(Cat2)	CatType	.12:name	Basil

注：相对路径列和浏览名(NodeId 列中括号内描述)并不是查询结果集中内容，仅仅只是为了表示的清晰。类型定义 NodeId 是一个整型的 NodeId，而不是表中的符号名称。

B.2.6 例 3

第三个例子提供了一个更为复杂的查询，其返回结果由多重标准过滤后得到。

例 3：通过一个人可获得 PersonType.LastName，AnimalType.name，ScheduleType.Period，其中这个人有一个宠物，宠物有喂食的时间表，同时这个人的邮政编码为'02138'，schedule.period 是每天一次或每小时一次并且喂养的食物量>10。

表 B.9 给出了例子中使用的节点类型描述参数。

表 B.9 例 3 节点类型描述

类型定义节点	包含子类型	相对路径	属性 ID	索引范围
PersonType	否	"12:PersonType.12:lastName"	值	不可用
		"12:PersonType<12:HasPet>12:AnimalType.12:name"	值	不可用
		"12:PersonType<12:HasPet>12:AnimalType<12:HasSchedule> 12:FeedingSchedule.period"	值	不可用

相应的内容过滤器见图 B.7

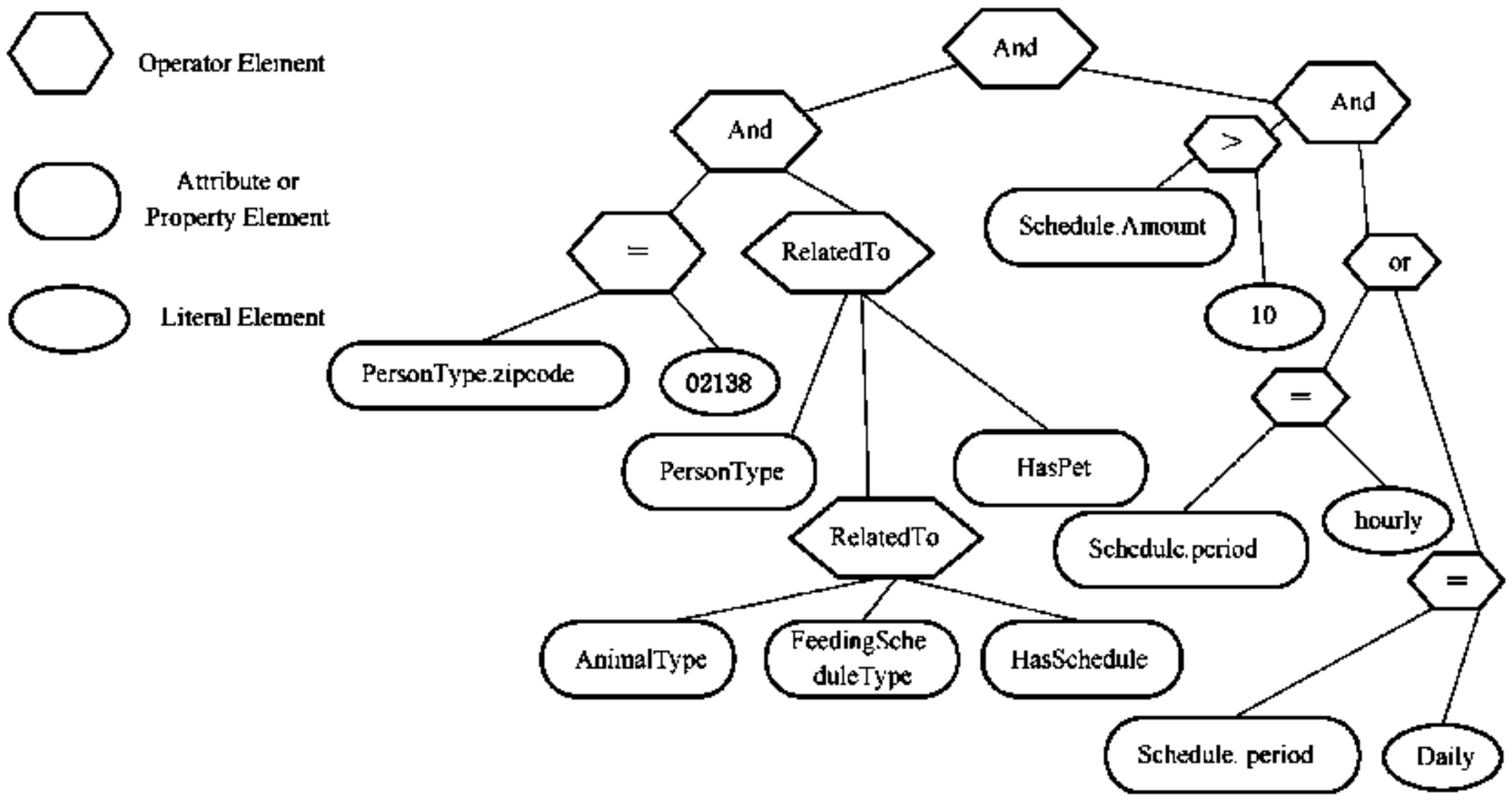

图 B.7　例 3 过滤器逻辑树

表 B.10　例 3 内容过滤器

元素 []	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
0	And	Element Operand　1	Element Operand　2		
1	And	Element Operand　3	Element Operand　5		
2	Or	Element Operand— 3	Element Operand— 9		
3	RelatedTo	Element Operand= 7	Element Operand= 8		
4	RelatedTo	AttributeOperand　NodeId:12:PersonType,BrowsePath ".",AttributeId:NodeId	Element Operand= 5	AttributeOperand = NodeId:12:HasPet,BrowsePath ".",AttributeId:NodeId	LiteralOperand = '1'
5	Equals	AttributeOperand = Node:12:AnilmalType,BrowsePath ".",AttributeId:NodeId	AttributeOperand = NodeId:12:FeedingScheduleType,BrowsePath ".",AttributeId:NodeId	AttributeOperand = NodeId:12:HasSchedule,BrowsePath ".",AttributeId:NodeId	LiteralOperand = '1'
6	Equals	AttributeOperand = NodeId:12:PersonTypeBrowsePath ".",AttributeId:zipcode	LiteralOperand — '02138'		
7	Equals	AttributeOperand — NodeId:12:PersonType BrowsePath "12:HasPet>12:AnimalType<12:HasSchedule>12:FeedingSchedule".AttributeId:Period	LiteralOperand = 'Daily'		

表 B.10（续）

元素 []	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
8	Equals	AttributeOperand = NodeId: 12:PersonType BrowsePath "12:HasPet>12:AnimalType<12: HasSchedule>12: FeedingSchedule",AttributeId: Period	LiteralOperand = 'Hourly		
9	GreaterThan	AttributeOperand NodeId: 12:PersonType BrowsePath "12:HasPet>12:AnimalType<12: HasSchedule>12: FeedingSchedule",AttributeId: Amount	ElementOperand = 10		
10	Cast	LiteralOperand 10	AttributeOperand = NodeId:String、 BrowsePath ".", AttributeId:NodeIdt		

查询的结果见表 B.11 中查询数据集。

表 B.11 例 3 查询数据集

NodeId	类型定义 NodeId	相对路径	值
12:30 (JFamily1)	PersonType	".12:lastName"	Jones
		"<12:hasPet>12:PersonType.12:name"	Rosemary
			Basil
		"<12:hasPet>12:AnimalType<12:hasSchedule>12:FeedingSchedule. 12:period"	Hourly
			Hourly

注：相对路径列和浏览名称(NodeId 列的括号中)不在查询数据集合中，只在这里展示。类型定义 NodeId 是一个整数而不是表中的符号名。

B.2.7 例 4

第 4 个例子给出了作为 RelatedToOperator 一部分的插图。

例 4：通过一个人可以获得 PersonType.lastName，这个人有一个孩子，孩子有一个宠物。

表 B.12 给出了例子中使用的节点类型描述参数。

表 B.12　节点类型描述

类型定义节点	包含子类型	相对路径	属性 Id	索引范围
PersonType	否	".12:lastName"	值	不可用

相应的内容过滤器见图 B.8。

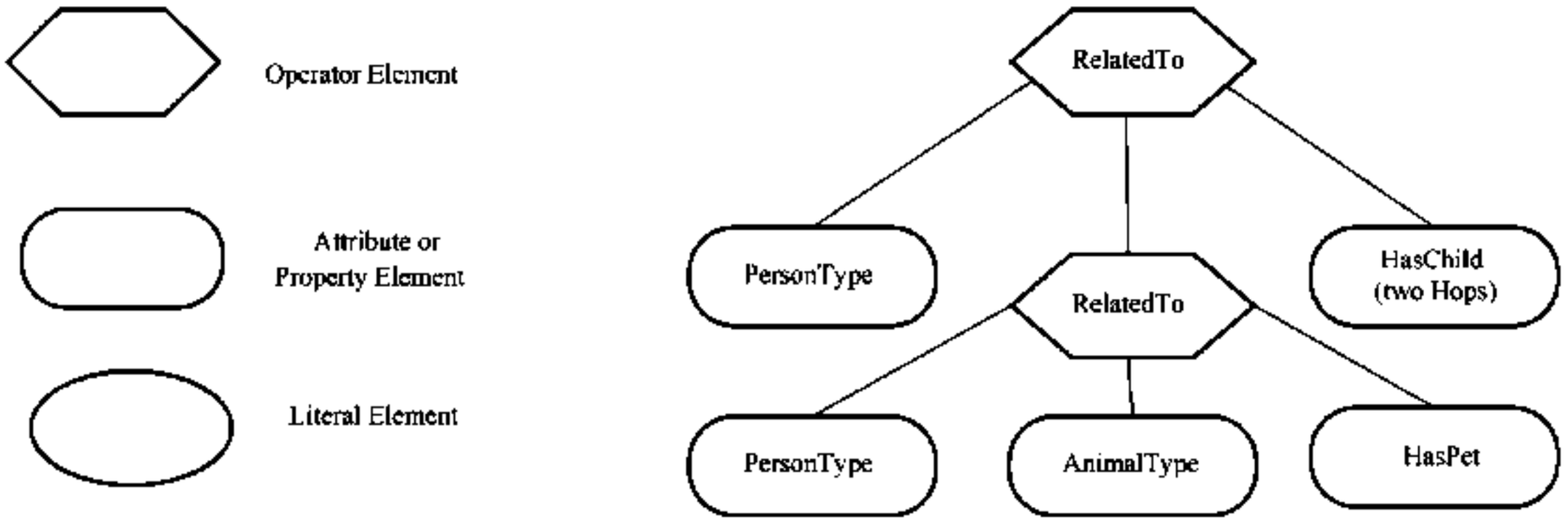

图 B.8　例 4 过滤器逻辑树

表 B.13 描述了例子中使用到的元素、运算符和操作数。

表 B.13　例 4 内容过滤器

元素 []	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
0	RelatedTo	AttributeOperand = NodeId: 12:PersonType. BrowsePath ".", AttributeId:NodeId	Element Operand = 1	AttributeOperand NodeId: 12:HasChild. BrowsePath ".", AttributeId:NodeId	LiteralOperand = '2'
1	RelatedTo	AttributeOperand NodeId: 12:PersonType. BrowsePath ".", AttributeId:NodeId	AttributeOperand NodeId: 12:AnimalType. BrowsePath ".", AttributeId:NodeId	AttributeOperand = NodeId:12:HasPet. BrowsePath ".", AttributeId:NodeId	LiteralOperand = '1'

查询的结果见表 B.14 中的查询数据集合。值得注意的是猪"Pig1"作为 Sara 的宠物被引用，但同时也作为 Sara 父亲 Paul 的农场动物被引用。

表 B.14　例 4 查询数据集

NodeId	类型定义 NodeId	相对路径	值
12:42 (HFamily1)	PersonType	".12:lastName"	Hervey

注：相对路径列和浏览名称(NodeId 列的括号中)不在查询数据集中，只在这里展示。类型定义 NodeId 是一个整数而不是表中的符号名。

B.2.8 例 5

第 5 个例子给出了一个别名使用的例子。

例 5：获取孩子们的姓氏，他们的名称相同都来自他们的父亲或母亲名称。

表 B.15 给出了例子中使用的节点类型描述参数。

表 B.15 例 5 节点类型描述

类型定义节点	包含子类型	相对路径	属性 Id	索引范围
PersonType	FALSE	"<12:HasChild>12:PersonType.12:lastName"	值	不可用

相应的内容过滤器见图 B.9。

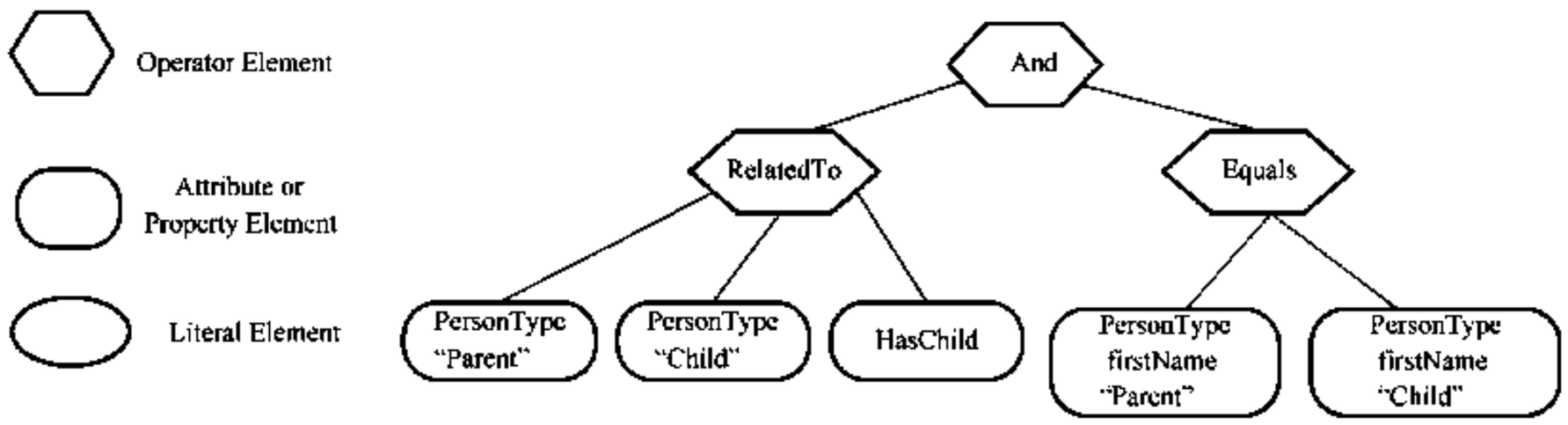

图 B.9 例 5 过滤器逻辑树

在这个例子中，一个 PersonType 的引用别名为"Parent"，另一个 PersonType 的引用别名为"Child"。然后 Rarent.firstName 和 Child.firstName 的值比较。表 B.16 描述了例子中用到的元素、运算符和操作数。

表 B.16 例 5 内容过滤器

元素 []	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
0	And	ElementOperand 1	ElementOperand 2		
1	RelatedTo	AttributeOperand = NodeId:12:PersonType, BrowsePath ".", AttributeId:NodeId, Alias:"Parent"	AttributeOperand = NodeId:12:PersonType, BrowsePath ".", AttributeId:NodeId, Alias:"Child"	AttributeOperand = NodeId:12:HasChild, AttributeId:NodeId	LiteralOperand = "1"
2	Equals	AttributeOperand = NodeId:12:PersonType, BrowsePath ".", AttributeId:FirstName, Alias:"Parent"	AttributeOperand NodeId:12:PersonType, BrowsePath ".", AttributeId:firstName, Alias:"Child"		

查询的结果见表 B.17 中查询数据集。

表 B.17　例 5 查询数据集

NodeId	类型定义 NodeId	相对路径	值
12:42 (HFamily1)	PersonType	"<12:HasChild>12:PersonType.12:lastName"	Hervey

B.2.9　例 6

第 6 个例子中提供了一种不同类型请求的插图，其中客户端显示了其关注的部分服务器地址空间。请求包含了将要返回的引用列表。

例 6：通过一个人可以获得 PersonType.NodeId、AnimalType.NodeId、PersonType.HasChild 引用、PersonType.HasAnimal 引用。其中这个人有一个孩子，他孩子有一个宠物。

表 B.18 描述了在例子中使用到的节点类型描述参数。

表 B.18　例 6 节点类型描述

节点定义节点	包含子类型	相对路径	属性 ID	索引范围
PersonType	否	".12:NodeId"	值	不可用
		<12:HasChild>12:PersonType <12:HasAnimal>12:AnimalType.NodeId	值	不可用
		<12:HasChild>	值	不可用
		<12::HasChild>12:PersonType <12:HasAnimal>	值	不可用

对应的内容过滤在图 B.10 中说明。

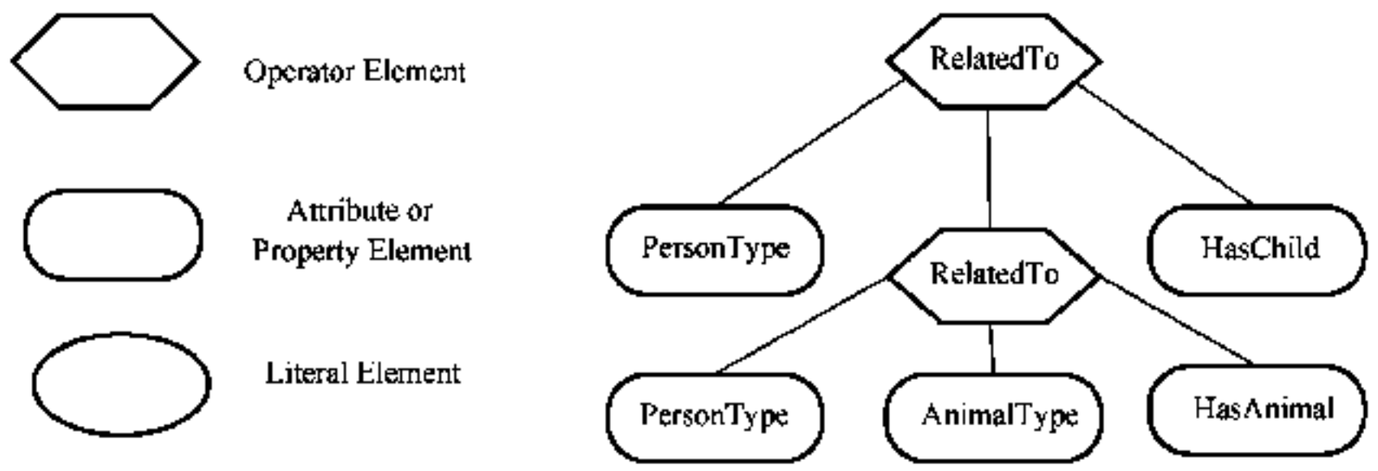

图 B.10　例 6 过滤逻辑树

表 B.19 描述了例子中使用的元素、运算符、操作数。

表 B.19　例 6 内容过滤器

元素[_]	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
0	RelatedTo（关联）	AttributeOperand（属性运算符）= NodeId:12:PersonType.BrowsePath（浏览路径）".", AttributeId:NodeId	ElementOperand（元素操作数）=1	AttributeOperand（属性操作数）= Node:12:HasChild.BrowsePath "."，AttributeId:NodeId	LiteralOperand（文本操作数）'1'

表 B.19（续）

元素[]	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
1	RelatedTo	AttributeOperand — NodeId: 12:PersonType,BrowsePath ".",AttributeId:NodeId	AttributeOperand —NodeId: 12:AnimalType, BrowsePath ".", AttributeId: NodeId	AttributeOperand — NodeId:12:HasAnimal, BrowsePath ".", AttributeId:NodeId	LiteralOperand = '1'

表 B.20 显示了包含在查询数据集中的查询结果。

表 B.20 例 6 查询数据集

NodeId	类型定义 NodeId	相对路径	值
12:42 (HFamily1)	PersonType	".NodeId"	12:42 (HFamily1)
		<12:HasChild>12:PersonType<12:HasAnimal> 12:AnimalType.NodeId	12:91 (Pig1)
		<12:HasChild>	HasChild *ReferenceDescription*
		<12:HasChild>12:PersonType<12:HasAnimal>	HasFarmAnimal *ReferenceDescription*
12:48 (HFamily2)	PersonType	".NodeId" <12:HasChild>12:PersonType<12:HasAnimal> 12:AnimalType.NodeId	12:48 (HFamily2) 12:91 (Pig1)
		<12:HasChild>	HasChild *ReferenceDescription*
		<12:HasChild>12:PersonType<12:HasAnimal>	HasPet *ReferenceDescription*

注：相对路径和浏览名(括号中)不包含在查询数据集中仅仅为了显示清楚，表中包含的 TypeDefinitionNodeId(类型定义 NodeId)是一个整数而不是一个符号名。这种情况下的值字段将是请求的 NodeId，但对例子来说括号中提供的浏览名由浏览名提供引用类型。引用列表在表 B.20 中，引用描述的值在 7.24 中。

表 B.21 提供了和表 B.20 中相同的一个查询数据集的例子并且没有附加额外的字段和最小的符号 ID。每一个请求的属性都有一个条目，对于一个属性返回的多个条目通常由逗号分隔(原书为 comas 可能是拼写错误应为 commas)。如果一个结构体将要被返回，该结构体将封闭在方括号中。对于引用描述结构体来说，包含一个结构体、DisplayName(显示名)以及(浏览名)被认为是一样的，他们在表 B.4 中定义。

表 B.21　例 6 无附加信息的查询数据集

NodeId	类型定义 NodeId	值
12:42	PersonType	12:42
		12:91
		[HasChild,TRUE,[48,HFamily2,HFamily2,PersonType],
		[HasFarmAnimal,TRUE[91,Pig1,Pig1,PigType]
12:48	PersonType	12:54
		12:91
		[HasChild,TRUE,[54,HFamily3,HFamily3,PersonType]]
		[HasPet,TRUE,[91,Pig1,Pig1,PigType]]

PersonType,HasChild,PigType,HasPet,HasFarmAnimal 等标识符在表 B.21 中使用，它们将传送给实际的 ExtendedNodeIds(ExpandedNodeId)。

B.2.10　例 7

第七个例子显示一个请求的插图，一个客户端想要显示基于一个开始点的部分的地址空间，这个开始点通过浏览获得。请求列表包含了将要返回的列表引用。在这种情况下，浏览了区域 2 的人想要查询开始点一下的查询信息。

例 7：通过在区域 2(Cleveland 节点)并且有一个孩子的人，可以得到 PersonType.NodeId,AnimalType.NodeId,PersonType.HasChild 引用，PersonType.HasAnimal 引用。

表 B.22 描述了例子中使用的 NodeTypeDescription(节点类型描述)参数。

表 B.22　例 7 节点类型描述

类型定义节点	包含子类型	相对路径	属性 ID	索引范围
PersonType	否	".NodeId"	值	不可用
		<12:HasChild>	值	不可用
		<12:HasAnimal>NodeId	值	不可用
		<12:HasAnimal>	值	不可用

对应的内容过滤器在图 B.11 中描述。

在典型情况下浏览调用将返回一个 NodeId，因此第一个过滤是 BaseObjectType 其 NodeId 是 95，同时 95 作为 NodeId 与区域 2 关联，所有的节点都是从 BaseObjectType 派生，同时 NodeId 是一个基础属性，因此这个过滤将作用于所有这种性质的查询。

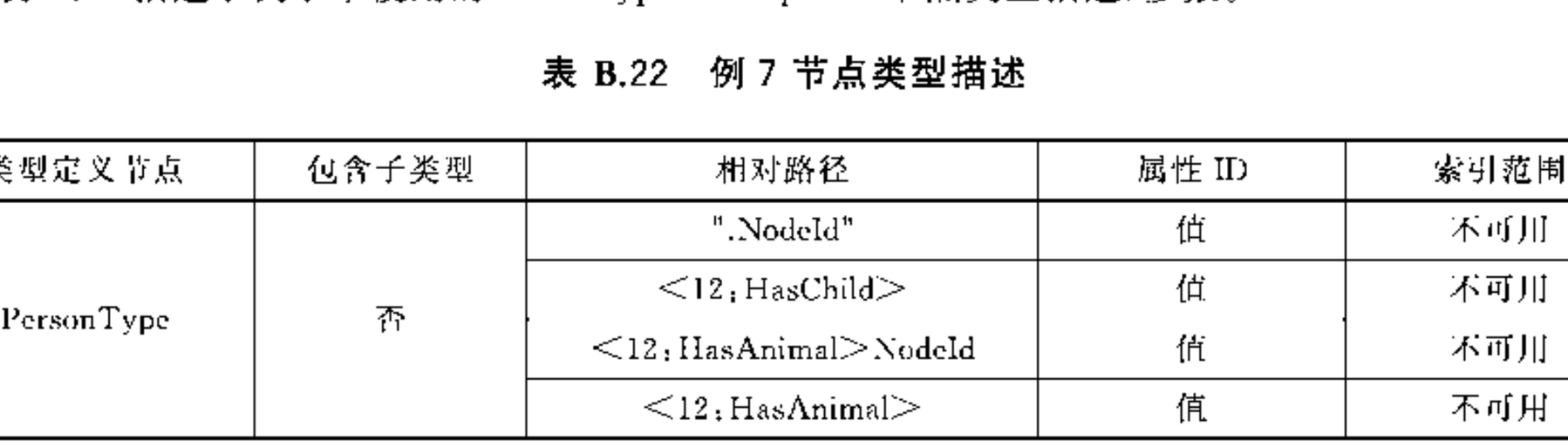

图 B.11　例 7 过滤逻辑树

表 B.23 描述了例子中使用的元素、运算符以及操作数。

表 B.23 例 7 内容过滤器

元素[]	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
0	RelatedTo	AttributeOperand = NodeId: BaseObjectType, BrowsePath ".", AttributeId: NodeId	ElementOperand = 1	AttributeOperand = Node:HierachicalReference, BrowsePath ".", AttributeId:NodeId	LiteralOperand = '1'
1	RelatedTo	AttributeOperand = NodeId: 12:PersonType, BrowsePath ".", AttributeId: NodeId	AttributeOperand = NodeId: 12:PersonType, BrowsePath ".", AttributeId:NodeId	AttributeOperand = NodeId: 12:HasChild, BrowsePath ".", AttributeId:NodeId	LiteralOperand = '1'
2	Equals	AttributeOperand = NodeId:BaseObjectType, BrowsePath ".", AttributeId: NodeId	LiteralOperand = '95'		

表 B.24 显示了包含在查询数据集中的查询结果。

表 B.24 例 7 查询数据集

NodeId	类型定义 NodeId	相对路径	值
12:42 (HFamily1)	PersonType	".NodeId"	12:42 (HFamily1)
		<12:HasChild>	HasChild *ReferenceDescription*
		<12:HasAnimal>12:AnimalType.NodeId	NULL
		<12:HasAnimal>	HasFarmAnimal *ReferenceDescription*
12:48 (HFamily2)	PersonType	".NodeId"	12:48 (HFamily2)
		<12:HasChild>	HasChild *ReferenceDescription*
		<12:HasAnimal>12:AnimalType.NodeId	12:91 (Pig1)
		<12:HasAnimal>	HasFarmAnimal *ReferenceDescription*

注：相对路径和浏览名(括号中)不包含在查询数据集中，此处仅仅为了显示清楚，表中包含的类型定义 NodeId 是一个整数而不是一个符号名。这种情况下的值字段将是请求的 NodeId，但对例子来说括号中提供的浏览名由浏览名提供引用类型。引用列表在表 B.24 中，引用描述的值在 7.24 中。

B.2.11 例 8

第八个例子提供了一个请求的插图，该请求的地址空间被服务器定义的视图所限制。这个请求和第二个例子相同，在第二个例子中说明的是接收的一个不相交接点的列表以及一个能接收的结果数组。对节点而言所有的参数以及内容过滤器相同是至关重要的，只有视图的描述被特定为视图 1。

例 8：通过一个人可以获得 PersonType.lastName，AnimalType.name，这个人限定在视图 1 的地址空间中，他有一个孩子或者(一个宠物猫，猫有一个喂食时间表)。

例子中使用的节点类型描述参数在表 B.25 中描述。

表 B.25 例 8 节点类型描述

类型定义节点	包含子类型	相对路径	属于 ID	索引范围
PersonType	否	".12:LastName"	值	不可用
AnimalType	是	".name"	值	不可用

对应的内容过滤器在图 B.12 中说明。

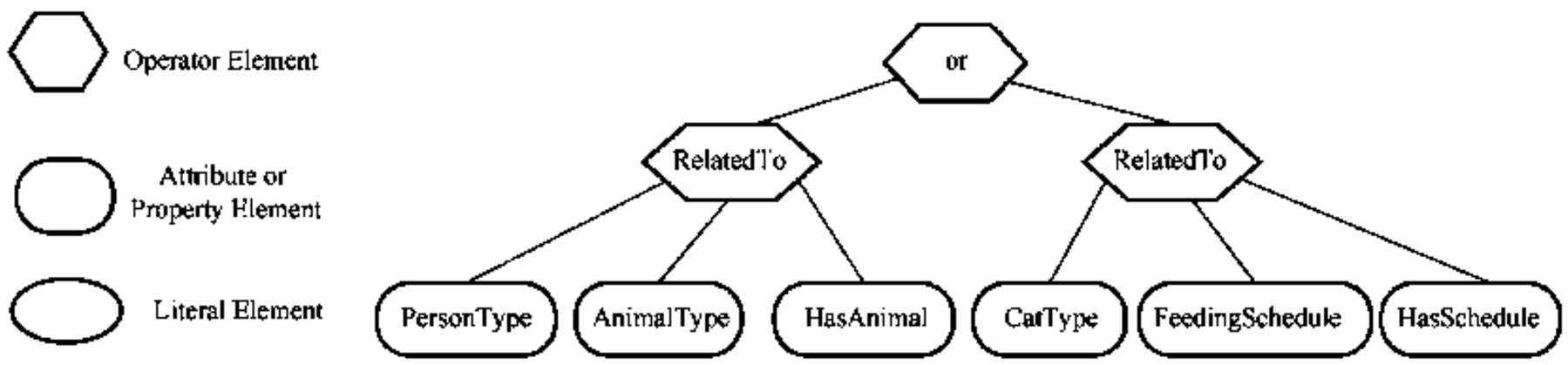

图 B.12 例 8 过滤逻辑树

表 B.26 描述了例子中使用的元素、运算符和操作数。值得注意的是一个 CatType 是 AnimalType 的子类型。

表 B.26 例 8 内容过滤器

元素[]	运算符	操作数[0]	操作数[1]	操作数[2]	操作数[3]
0	Or	ElementOperand=1	ElementOperand = 2		
1	RelatedTo	AttributeOperand = NodeId: 12:PersonType, BrowsePath ".", AttributeId:NodeId	AttributeOperand NodeId: 12:PersonType, BrowsePath ".", AttributeId:NodeId	AttributeOperand = NodeId:12:HasChild, BrowsePath ".", AttributeId:NodeId	LiteralOperand = '1'
2	RelatedTo	AttributeOperand NodeId:12:CatType, BrowsePath ".", AttributeId:NodeId	AttributeOperand NodeId: 12:FeedingScheduleType, BrowsePath ".", AttributeId:NodeId	AttributeOperand NodeId: 12:HasSchedule, BrowsePath ".", AttributeId:NodeId	LiteralOperand '1'

表 B.27 显示了包含查询数据集的查询结果。如果与例 2 的结果相比较的话，不同之处在于忽略了猫节点。这些节点不在视图中因此没有包含在结果集中。

表 B.27 查询数据集

NodeId	类型定义 NodeId	相对路径	值
12:30(Jfamily1)	PersonType	.12:LastName	Jones

注：相对路径列和浏览名(括号中 NodeId 列)不包含在查询数据集中仅仅是为了显示清楚，表中包含的类型定义 NodeId 是一个整形而不是一个符号名。

B.2.12 例 9

第 9 个例子提供了一个进一步的请求插图其地址空间通过在服务器中定义视图的方式以达到限制。这个请求和第二个例子类似，只不过节点的请求在相对路径中表示。重要的是要注意内容过滤是一样的，仅仅试图描述被特定为“视图 1”。

例 9：通过一个人可以获得 PersonType.lastName、AnimalType.name，人限定在视图 1 的地址空间中，他有一个孩子或(一个宠物猫，猫有一个喂食时间表)。表 B.28 描述例子中使用的节点类型描述参数。

表 B.28 例 9 节点类型描述

类型定义节点	包含子类型	相对路径	属性编号	索引范围
PersonType	否	".NodeId" <12:HasChild>12:PersonType<12:HasAnimal>12:AnimalType.NodeId	值	不可用
		<12:HasChild>		
		<12:HasChild>12:PersonType <12:HasAnimal>		
PersonType	否	".12:LastName"	值	不可用
		<12:HasAnimal>12:AnimalType. 12:Name	值	不可用
AnimalType	是	".12:name"	值	不可用

对应的内容过滤器见图 B.13。

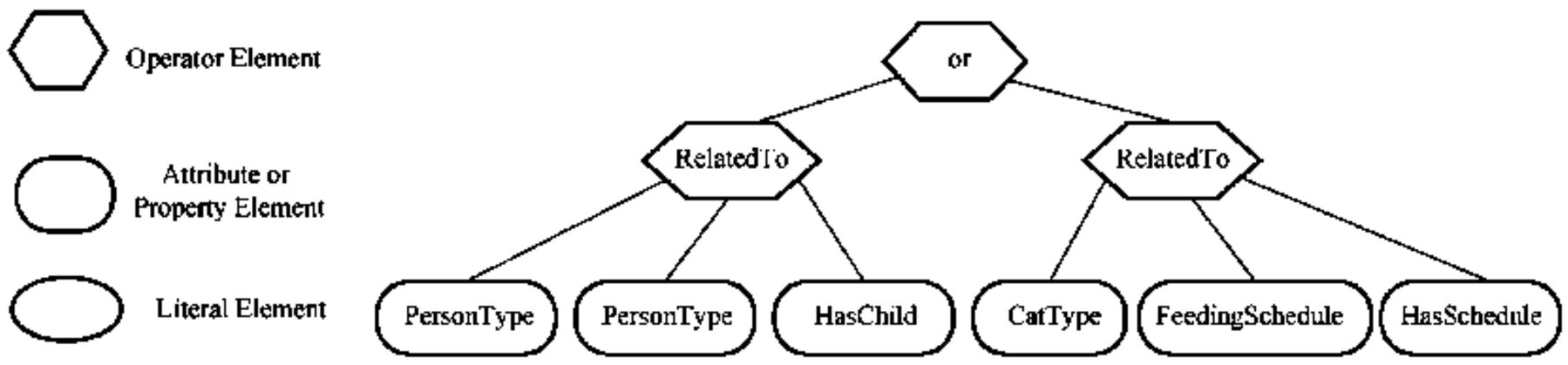

图 B.13 过滤逻辑树

表 B.29 描述了例子中使用的元素、运算符、操作数。

表 B.29 例 9 ContentFilter

元素[_]	运算符	操作数[0_]	操作数[1_]	操作数[2_]	操作数[3_]
0	Or	ElementOperand=1	ElementOperand = 2		
1	RelatedTo	AttributeOperand NodeId: 12:PersonType. BrowsePath ".", AttributeId:NodeId	AttributeOperand = NodeId: 12:PersonType. BrowsePath ".", AttributeId: NodeId	AttributeOperand = NodeId:12:HasChild. BrowsePath ". ", AttributeId:NodeId	LiteralOperand = '1'
2	RelatedTo	AttributeOperand = NodeId:12:CatType. BrowsePath ". ", AttributeId:NodeId	AttributeOperand = NodeId: 12:FeedingScheduleType. BrowsePath ".", AttributeId: NodeId	AttributeOperand = NodeId: 12:HasSchedule. BrowsePath ". ", AttributeId:NodeId	LiteralOperand '1'

表 B.30 中显示了包含查询数据集的查询结果。如果和例 2 结果比较的话，就是尽管宠物节点在视图之外，也被包含在了列表中。这是可能的，因为名称在视图中被相对路径以及根节点引用。

表 B.30 例 9 查询数据集

NodeId	类型 NodeId	相对路径	值
12:30(Jfamily1)	PersonType	.12:LastName	Jones
		<12:HasAnimal>12:AnimalType. 12:Name	Rosemary
		<12:HasAnimal>12:AnimalType. 12:Name	Basil

注：相对路径列和浏览名(括号中 NodeId 列)不包含在查询数据集中，其仅仅为了显示清楚，表中包含的类型定义 NodeId 是一个整形而不是一个符号名。

参 考 文 献

[1] IEC 62541-11 OPC Unified Architecture—Part 11:Historical Access
[2] IEC 62541-12 OPC Unified Architecture—Part 12:Discovery
[3] IEC 62541-13 OPC Unified Architecture—Part 13:Aggregates

中华人民共和国
国家标准
OPC 统一架构 第 4 部分:服务
GB/T 33863.4—2017/IEC 62541-4:2011
*
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.org.cn
服务热线:400 168 0010
2017 年 7 月第一版
*
书号:155066·1-56670

GB/T 33863.4-2017