第34卷 第12期 计 算 机 工 程 2008年6月
Vol.34 No.12 Computer Engineering June 2008
·开发研究与设计技术· 文章编号：1000—3428(2008)12—0248—02 文献标识码：A 中图分类号：TP311.51

EPCglobal应用层事件引擎设计与实现

孟 和，赵 政，薛桂香，李玉峰
(天津大学计算机科学与技术学院，天津 300072)

摘 要：射频识别中间件的推出很好地解决了将物理基础架构收集到的信息传递给企业应用程序并且被企业应用程序所使用的技术与成本问题，加速了射频识别技术的推广使用。该文介绍了一个支持EPCglobal应用层事件规范的射频识别中间件引擎，引擎由内核模块、设备管理模块、事件过滤模块和配置模块组成，将应用程序和设备接口分离，过滤和处理读卡器捕获的原始观测值，提供设备管理和查询的应用程序级接口等功能。阐述该服务器的功能结构、设计和实现。

关键词：射频识别；应用层事件；事件过滤

Design and Implementation of EPCglobal Application Level Events Engine

MENG He, ZHAO Zheng, XUE Gui-xiang, LI Yu-feng
(School of Computer Science and Technology, Tianjin University, Tianjin 300072)

【Abstract】 Radio-Frequency Identification(RFID) middleware turns out to be the right solution for exposing and digesting the information collected by physical infrastructure for enterprise applications, which in practice accelerates the widely-using of RFID technology. This paper introduces an RFID middleware, which supports the EPCglobal Application Level Events(ALE) specification. The engine is composed of four modules, which are kernel module, device management module, event filter module, and configuration module. The engine provides features to encapsulate the applications from device interfaces, processes the raw observations captured by the readers and sensors, and provides an application-level interface for managing readers and querying RFID observations. The functional architecture, design and implementation of the engine are discussed.

【Key words】 Radio-Frequency Identification(RFID); Application Level Events(ALE); event filter

1 概述

选择合适的电子标签和读卡器、决定天线的部署等是构建一个RFID系统的第一步。由于RFID设备与企业后端系统之间的整合技术复杂，因此要想将这些物理基础架构收集到的信息传递给企业应用程序，并被其使用， RFID中间件就成为必要的解决方案。RFID中间件很好地解决技术与成本这2方面问题，加速了RFID技术的推广使用。

因为RFID中间件在企业应用和读卡器的数据采集间扮演很重要的角色，所以国内外一些大学和研究团体也开始了对RFID中间件的研究。MIT自动识别中心所提出的EPC网络[1]，包括电子产品代码(EPC)编码、Savant、对象名字服务(ONS)、EPC Information Services、物理标示语言(PML)等关键技术，得到很多学术机构和企业的支持。应用层事件(ALE)最初作为Savant应用的一部分开发而成，如今归属于标准组织和供应链利益集团组成的联盟EPCglobal，它是旨在把低层的电子产品代码(EPC)数据和较高层的企业系统相互连接的EPCglobal网络计划的一部分，已成为事实上的标准。

2 引擎设计

ALE引擎是介于读卡器和企业应用系统之间的中间件系统。通过网络互连的ALE系统形成一个对电子标签产生的事件进行管理和响应的框架。ALE引擎将来自企业应用程序的请求发送给读卡器，并且接收电子标签的唯一标识信息和通过传感器采集获得的其他信息，将该信息传递给应用程序[2]。它对服务接口进行就像SQL对关系数据库的内部机制进行的抽象处理。应用程序或者EPCIS可以通过ALE查询引擎，不必关心网络协议或者设备的具体情况。

除了合并多个来源读取的EPC数据外，ALE引擎还有很多其他功能。它可以简化剔除来自某家生产厂商或者来自仓库某个地方的标签。基于时间和增量变化的标准也有助于异常处理，如把曾经超过特定辐射范围、但后来又回到辐射范围的某个标签隔离开。最重要的是ALE可以对硬件和厂商的变化进行隔离，克服了扩展性问题，并且解决了复杂的编程同步问题。否则，为了能够共享后端应用之间的多个读卡装置资源，势必需要编程同步。

作者设计和开发了一个支持EPCglobal ALE规范的RFID中间件引擎，它由多个有独立功能的模块组成，包括引擎内核、设备管理模块、事件过滤模块和配置模块，如图1所示。这些模块结合起来满足特定企业应用程序所要求的功能。引擎可以按照需要，通过配置以周期性或者异步事件的方式，来报告捕获的信息。引擎包括2个主要与外部系统通信的接口，即读卡器协议接口和引擎服务接口。读卡器协议接口提供引擎与读卡器之间的通信，引擎服务接口使得引擎

作者简介：孟 和(1980－)，男，博士研究生，主研方向：数据库，计算机网络；赵 政，教授、博士生导师；薛桂香、李玉峰，博士研究生

收稿日期：2007-06-29 **E-mail：**menghe@gmail.com

服务除了为完成特定功能模块的交互，还可以与外部服务交互，例如与 EPC 信息服务(EPCIS)或其他本地企业应用程序的交互。

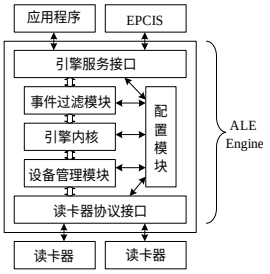

图 1　EPCglobal ALE 引擎组成

3　引擎实现

3.1　引擎内核

事件管理是 ALE 引擎设计中最为主要的功能。事件是指通过与读卡器的交互而获得的一系列 EPC 数据，事件管理主要是对来自电子标签的数据进行过滤、聚集和统计。因为 RFID 系统会持续读取电子标签的信息，所以引擎要处理来自 RFID 系统的大量数据。

因为企业应用程序是针对特定类型的，并不是对所有电子标签的所有数据都感兴趣，例如读卡器可能会重复读取来自多个电子标签的数据、多个读卡器读到同样电子标签的数据，就是一些冗余信息。数据过滤器可以减少这些信息。

发送到引擎的事件请求都表示为ECSpec，从引擎返回的数据都表示为ECReports。ECSpec和ECReports是表示EPC的 2 个标准XML片断实例，以一种结构化和统一的方式在EPC信息的读取、存储和传输过程中对其进行描述，使得对标签信息的理解、存储和传送更容易。ECSpec和ECReports的XML模板定义预留了可供应用程序扩展的部分，通过这些扩展来捕获和报告物理事件和测量值[3]。核心的XML模板主要描述通用的物理对象属性和可观测属性，例如有效期、生产日期、重量或者某个物体在某地出现的次数等。

如图 2 所示，引擎提供 2 种基本交互模型：同步模型和异步模型。

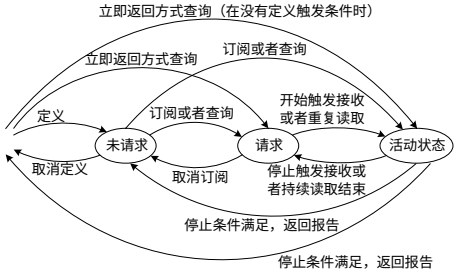

图 2　ALE 引擎内核状态转移图

最基本的交互模型是请求/响应模型，即所有在 ALE 服务中被调用的方法和操作都同步执行，引擎支持的 2 种同步模型模式分别是立即返回方式和轮询方式。引擎也提供了一个异步模型，异步模型中客户端可以订阅一个事件，当事件发生时，ALE 服务会异步地将数据交付给客户端。

通过 ALE 的定义方法指定一个名称，可以使 ECSpec 进入“ 未请求 ”状态；ECSpec 进入“ 请求 ”状态后，会等待 ALE 服务的处理；当 ECSpec 进入“ 活动 ”状态，ALE 服务器开始根据“ 事件周期 ”中指定的开始触发条件，在条件满足的情况下开始收集来自“ 读卡器 ”的原始 EPC 观测值，当“ 事件周期 ”中指定的停止触发条件满足时返回查询报告。

3.2　设备管理模块

引擎的设备管理模块可以对来自不同厂商的硬件的变化进行隔离，克服了扩展性问题，并且解决了复杂的编程同步问题，使得企业应用程序可以共享后端应用之间的多个读卡器资源。

如图 3 所示，设备管理模块主要包括 Reader 包、EventHandler 包和 ReaderTypes 包。Reader 包主要是一些表示物理读卡器和逻辑读卡器的类和接口：PhysicalReader 表示物理设备；LogicalReader 表示物理设备上的天线，是设备管理的最小单位；ReadBuffer 类包含读取自逻辑读卡器的标签数据；TagReport 包含 ReadBuffer 中读到的每一张单独的卡。EventHandler 包主要是一些表示事件和事件处理器的类和接口：EventType 表示一个读卡器事件类型(例如读卡事件、触发事件)，该类是一个类型安全的枚举结构；EventArgs 用来表示事件数据，EventHandler 会用其来实现事件通告和处理。Readertypes 包放置所有物理和逻辑读卡器的实现类，例如，XXXPhysicalReader 表示 XXX 读卡器的接口，而 XXXLogicalReader 表示 XXX 逻辑读卡器的借口。

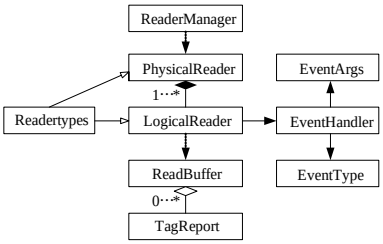

图 3　ALE 引擎设备管理模块主要类图

3.3　事件过滤模块

引擎的事件过滤模式的指定可以通过“ 包含模式 ”和“ 排除模式 ”2 种模式[4]。也就是说最后返回的报告中出现的EPC至少要与“ 包含模式 ”列表中的一个模式匹配，并且不能与任何出现在“ 排除模式 ”中的模式匹配上，描述如下：

F(R)={epc|ep ∈ R&(epc ∈ includePattern1|epc ∈ includePattern2|…|epc∈includePatternN)

&!(epc ∈ excludePattern1|epc ∈ excludePattern2|…|epc ∈ excludePatternN)}

如图 4 所示，事件过滤模块主要由 Filter 包和 Filtertypes 包来实现。

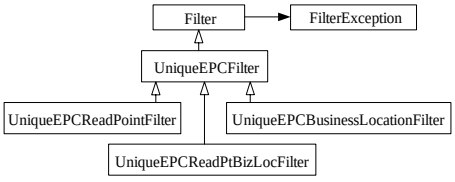

图 4　ALE 引擎设事件过滤模块主要类图

Filter 包主要是表示设备读标签过滤器的类和接口，每个过滤器提供基于特定给定标准来提取某些感兴趣的设备读取到的标签数据，其中的 FilterException 表示过滤器处理过程中的错误。Filtertypes 包主要是过滤器接口的具体实现类。例如，UniqueEPCFilter 提供从冗余的 EPC 读数中过滤唯一 EPC 的功能，UniqueEPCReadPointFilter, UniqueEPCBusiness-LocationFilter, UniqueEPCReadPtBizLocFilter 则分别根据读卡点、业务作业点以及两者结合起来对 EPC 读数进行过滤。

(下转第 253 页)