Auto-ID Middleware

Auto-ID Middleware
Software Requirement Specification
Document Version 1.0

Auto-ID Middleware
Software Requirement Specification

Revision History

Version	Date	Author	Remarks
1.0			Initial Version

Table of Contents

1 Overview

Auto-ID Middleware 是一个中间件系统，能够跟各种不同硬件厂商的标签读/写器相连，采集及存储识读器检测到的电子标签/条形码的信息，并提供接口与 SAP AII、CIP 和其他 ERP 系统相连。它的主要特点是：

- 可以同时连接/控制多个不同硬件厂商的 Auto-ID 读/写设备；
- 采用 Plug-in 和 Agent 的机制，把条形码和 RFID 电子标签、标签识读器和标签打印机、手持式和固定标签读/写设备，通过 TCP/IP 或串口通讯的方式整合到同一平台中；
- 通过用户自定义任务的机制，灵活支持各种实时或非实时的 Tag 数据采集和货品打包逻辑；
- 支持多种开放、标准的外部接口同步或异步地传送 Tag 信息，如 SOAP、HTTP、JMS、FTP 等；Tag 信息可以统一地封装在 PML 文档中发送给外部系统；
- 提供 Web 页面直观方便地供用户配置、监控系统及执行标签的读/写逻辑；
- 支持多种数据库系统，如 Oracle、MS SQL Server、MySQL、PostgreSQL 及 MaxDB；
- Middleware 是 Java 编程的系统软件，可运行在多种操作系统和 Application Server 上；
- Middleware 是企业级应用 (Enterprise Application)，具有高可靠性、安全性、标准化、可扩展性、可配置性及良好的性能指标；

1.1 Glossary

RFID: Radio Frequency Identification
Tag: 标签，泛指 RFID 电子标签和条形码标签
EPC: Electronic Product Code
Bar Code: 条形码
Reader: 标签识读器
Printer: 标签打印机
Device: 标签读/写设备，泛指 RFID Tag 及 Bar Code 的、手持的或固定的标签识读器或标签打印机
Device Plug-in: 运行在服务器端与 Device 或 Device Agent 通讯的标签读/写设备插件程序
Device Agent: 运行在于硬件设备直接相连的计算机上的数据转换和控制程序，它通过 TCP/IP 和 Middleware 交互
EMS: Event Management System
TMS: Task Management System
Event：由 Tag ID、Device ID、时间戳等组成的一个事件
Event Queue: 事件队列
Event Logger: 事件记录器
SAP AII: SAP Auto-ID Infrastructure
DC: Device Controller, software component linking SAP AII with device hardware
CIP: Collaboration Inventory Portal
SOAP: Simple Object Access Protocol
JMS: Java Message Service
PML: Physical Markup Language

1.2 References

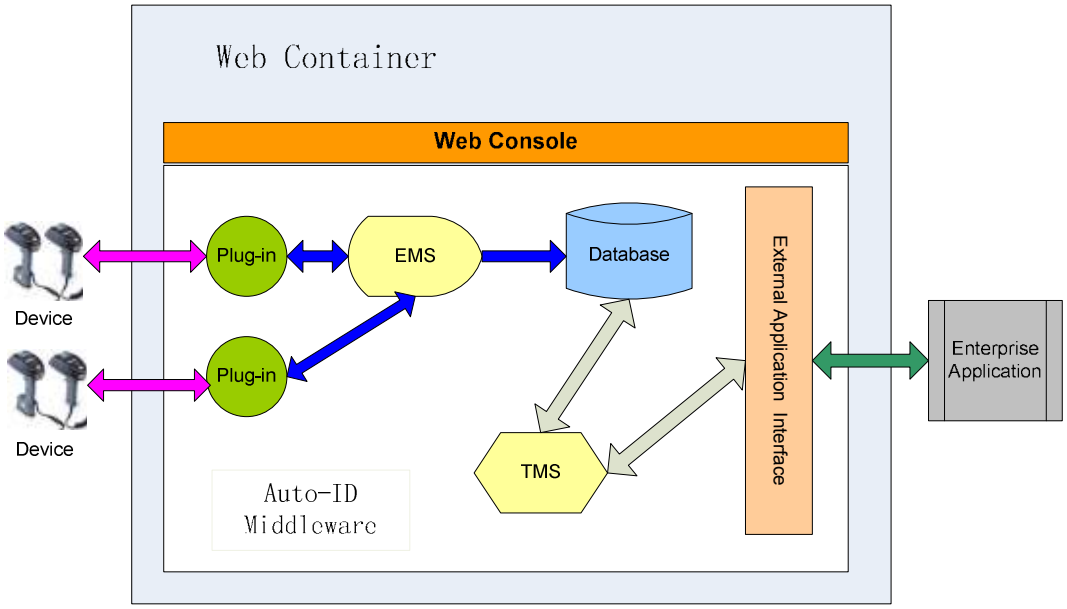
- Alien Reader Interface Guide & Java Developer Guide
- SAP Auto-ID Infrastructure 2.1 Interim Version
- SAP AII-DC Interface Description 1.0
- PML Core Specification, http://www.epcglobalinc.com/standards_technology/Secure/v1.0/PML_Core_Specification_v1.0.pdf
- EPC Tag Data Standards Version 1.1 Rev.1.24, http://www.epcglobalinc.org/standards_technology/EPCTagDataSpecification11rev124.pdf

2 Objective

本文档描述了 Auto-ID Middleware 的功能需求、体系结构、处理逻辑和运行开发环境。文档的最终读者是 Business Analyst, Project Architect, System Deployment Engineer, Software Developer, QA and Technical Writer。

3 System Architecture

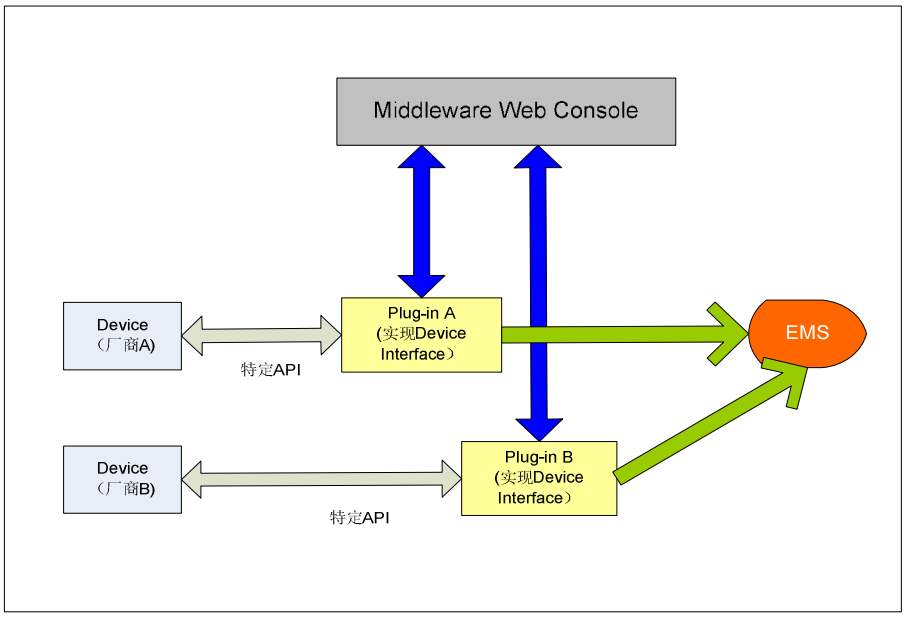
Middleware 的体系结构如下图所示：



4 Device Plug-in and Agent

Plug-in 是一个 Java Class，对 Middleware 支持的每种 Device/Device Agent 都要编写其对应的 Plug-in。Middleware 会启动一个线程来运行每个配置的 Plug-in。Plug-in 通过 TCP/IP Socket 或串口，调用 Device/Device Agent 特定的 API 建立和 Device/Device Agent 的连接，而 Plug-in 和 EMS、Web Console 之间通过统一的接口 Device 来交互，这样不同厂商的硬件设备都可通过 Plug-in 无缝地集成到 Middleware 中。

Device Agent 是一个单独运行的软件，可以使用任何语言编写。对只支持串口通讯的硬件设备，连接到其附近的计算机，计算机上运行对应的 Device Agent 软件。它通过串口与 Auto-ID 设备交互，同时通过 TCP/IP 与 Middleware 的 Plug-in 连接。Device Agent 主要的作用是数据流的协议转换（串口与 TCP/IP）及可能的格式转换。



4.1 Interface Device

每个 Device Plug-in 都要求实现 com..middleware.plugin.Device 接口，如果是标签识读器还要实现 java.lang.Runnable 接口以支持自动连续的 Tag ID 的读取：

```
package com..middleware.plugin;

import java.util.concurrent.BlockingQueue;

public interface Device {
    public void setDeviceID(String deviceID);
    public void setDeviceName(String deviceName);
    public void setUserName(String username);
    public void setPassword(String password);
    public void setPersistTime(int persistTime);
    public void setEventQueue(BlockingQueue<Event> queue);

    public void openNetworkDevice(String ip, int port, int openMode)
        throws DeviceOperationException, UnsupportedFeatureException;
    public void openSerialDevice(String serialPortName, int baudRate, int openMode)
        throws DeviceOperationException, UnsupportedFeatureException;
    public void closeDevice()
        throws DeviceOperationException;

    public String getDeviceID();
    public String getDeviceName();
    public String getDeviceIPAddress();
    public int getDeviceIPPort();
    public String getDeviceSerialPort();
    public int getDeviceBaudRate();
    public String getUserName();

    public boolean isDeviceWorking() throws DeviceOperationException;
    public boolean isReader();
    public boolean isPrinter();
    public boolean isOneTimeReader();

    public Tag[] getTagList()
        throws DeviceOperationException, UnsupportedFeatureException;
    public boolean writeTag(String label, String tagInfo)
        throws DeviceOperationException, UnsupportedFeatureException;
}
```

接口 Device:			
方法名	输入参数	返回结果	说明

setDeviceID	名称 | 描述 deviceID | 设置 Device ID		设置 device ID
setDeviceName	名称 | 描述 deviceName | 设置设备名		设置 device name
setUserName	名称 | 描述 username | 设置连接所需的用户名		设置连接所需的用户名
setPassword	名称 | 描述 password | 设置连接所需的密码		设置连接所需的密码
setPersistTime	名称 | 描述 persistTime | 设置 Tag 在 RFIDReader 的 tag list 中存放的时间		设置 persist time(秒)
setEventQueue	名称 | 描述 queue | Reference to Middleware event queue		Middleware 启动 Plug-in 时把事件队列传入。如果 device 为 Reader，Plug-in 的实现类要存储它
openNetworkDevice	名称 | 描述 ip | 读卡器的 IP 地址 port | 读卡器的端口号 openMode | 打开方式，可以为 READER_MODE|PRINTER_MODE	如果 Device 不支持这项操作，抛出 UnsupportedFeatureException；如果发生异常，抛出 DeviceOperationException	此方法用于和指定的 device 建立一个 TCP/IP 连接
openSerialDevice	名称 | 描述 serialPortName | 读卡器连接的串口号 baudRate | 波特率 openMode | 打开方式，可以为 READER_MODE|PRINTER_MODE	如果 Device 不支持这项操作，抛出 UnsupportedFeatureException；如果发生异常，抛出 DeviceOperationException	此方法用于和指定的 device 建立一个串口连接
closeDevice	无	如果发生异常，抛出 DeviceOperationException	此方法用于关闭和 device 的连接
getDeviceID	无	返回 device 的标识号	
getDeviceName	无	返回设备名	
getDeviceIPAddress	无	返回 device 的 IP 地址	
getDeviceIPPort	无	返回 device 的 IP 端口。	
getDeviceSerialPort	无	返回 device 的串口名。	
getDeviceBaudRate	无	返回 device 串口连接的波特率。	
getUserName	无	返回访问 device 的用户名	
isDeviceWorking	无	返回 Device 的工作状态。如果发生异常，抛出 DeviceOperationException	返回 true/false
isReader	无	返回 true 如果 device 是一个 reader	返回 true/false
isPrinter	无	返回 true 如果 device 是一个 printer	返回 true/false

isOneTimeReader	无	返回 true 如果 device 是一个一次性读取的 Reader(如 Barcode Reader)	返回 true/false
getTagList	无	数组 Tag[]。如果 Device 不支持这项操作，抛出 UnsupportedFeatureException；如果发生异常，抛出 DeviceOperationException	返回 RFIDReader 此时所能检测到的所有 Tag 的信息
writeTag	名称 / 描述 label / 打印的标签 tagInfo / 要写入 Tag 的信息	返回 true/false。如果 Device 不支持这项操作，抛出 UnsupportedFeatureException；如果操作发生异常，则抛出 DeviceOperationException,	标签打印机打印标签及向写 Tag。True 表明操作成功并且写到 Tag 中的信息也能正确读出加以验证。False 表明操作成功但写入信息没被验证

在 Middleware 启动 Plug-in 时 Middleware 要调用 setDeviceID()赋给 device 一个唯一的标识符(EPC 或 Bar Code)，Plug-in 要使用它来填充 Event 中的 deviceID 字段；setDeviceName()赋给 device 一个唯一名字，其可以出现在 PML 文档中。方法 setPersistTime()由 RFID Reader 使用，指定生成 Remove 事件的时间间隔，当一个 Tag 在 persistTime 的间隔内都没被 Reader 检测到时，就认为这个 Tag 已经移出了 Reader 的检测范围，Plug-in 就会生成一个 Remove 事件。Fixed RFID Reader 一定要实现 getTagList()方法，返回调用时所能检测到的所有 Tag 的信息；标签打印机一定要实现 writeTag()方法来打印标签及写 Tag。

Middleware 的事件队列在 Middleware 启动 Plug-in 时由 setEventQueue()传入。Fixed RFID Reader 在新的 Tag 初次被检测到和 Tag 移出时都要向 Middleware 的事件队列中写入事件（对应事件类型 Event.ADD 和 Event.REMOVE）；手持式 RFID Reader 和 Barcode Reader 在读到标签时向事件队列中写入事件（对应事件类型 Event.READ）。Reader 要实现 Runnable 接口供 Middleware 启动线程，运行 Plug-in 实现的事件检测和写入到事件队列的处理逻辑。

4.2 Interface Tag

Tag 接口的定义如下:

```
package com..middleware.plugin;
public interface Tag {
    public String getTagID();
    public long getFirstSeenTime();
    public void setTagID(String tagID);
    public void setFirstSeenTime(long firstSeenTime);
}
```

接口 Tag，代表 Device 读取的标签信息

方法名	输入参数	返回结果	说明
getTagID	无	返回标签的 ID 号	返回标签的 ID 号
getFirstSeenTime	无	时戳	返回第一次检测到该 Tag 的时间
setTagID	名称 / 描述 tagID / 标签 ID 号	无	设置 Tag 的 ID 号
setFirstSeenTime	名称 / 描述 firstSeenTime / 时戳	无	设置第一次检测到该 Tag 的时间

4.3 Interface Event

Event 接口的定义如下:

```
package com..middleware.plugin;
public interface Event {
```

```
    public static final int ADD = 0;
    public static final int REMOVE = 1;
    public static final int READ = 2;

    public int getEventType();
    public String getTagID();
    public String getReaderID();
    public long getEventTime();

    public void setEventType(int eventType);
    public void setTagID(String tagID);
    public void setReaderID(String deviceID);
    public void setEventTime(long eventTime);
}
```

接口 Event，代表事件队列中的事件

方法名	输入参数	返回结果	说明
getEventType	无	返回事件类型(为 ADD 或 REMOVE)	返回事件类型
getTagID	无	返回标签的 ID	
getReaderID	无	返回 Reader 的 ID	
getEventTime	无	时戳	返回事件发生的时间
setEventType	名称: eventID; 描述: 事件类型	无	设置事件类型
setTagID	名称: tagID; 描述: 标签 ID 号	无	设置 Tag 的 ID 号
setReaderID	名称: readerID; 描述: Reader 的 ID	无	设置产生该消息的 reader ID
setEventTime	名称: eventTime; 描述: 时戳	无	设置事件发生时间

4.4 Pre-bundled Plug-in and Agent

Middleware 1.0 会捆绑上若干已实现的 Plug-in and Agent: Alien, Intermec, Symbol, Generic Barcode Reader & Printer。

5 Event Management System

EMS 用于把多个 Device 产生的 Event 写入到数据库，EMS 的任务是：

- 启动/初始化系统配置的多个 Device，并提供接口供 Web Console 控制 Device Plug-in；
- 保持一个系统事件队列，对事件进行缓冲，使得数据记录器(Logger)和 Device Plug-in 能够互不干扰地工作；
- 将系统事件队列中的 Event 写入到数据库中；

5.1 EMS 包括的组件

1. 数据记录器(Logger)

其作用是把事件队列中的 Event 数据存入数据库。

2. 事件队列(Event Queue)

多个 Device 可以同时向数据队列写入 Event，事件队列在 Logger 把它们写入到数据库之前暂存这些数据。EventQueue 是容量可配置的队列。如果容量配置过大，那么可能导致浪费过多的内存；但如果配置过小，可能会当 Event 过多而 Logger 来不及把队列中数据写入到数据库时，因等待而产生时延。

3. 初始化及若干 Utility Class

提供接口供系统启动时初始化 EMS 的内部数据结构及启动 Plug-in 线程。也实现 Utility Class 供 Web Console 控制 Device Plug-in。

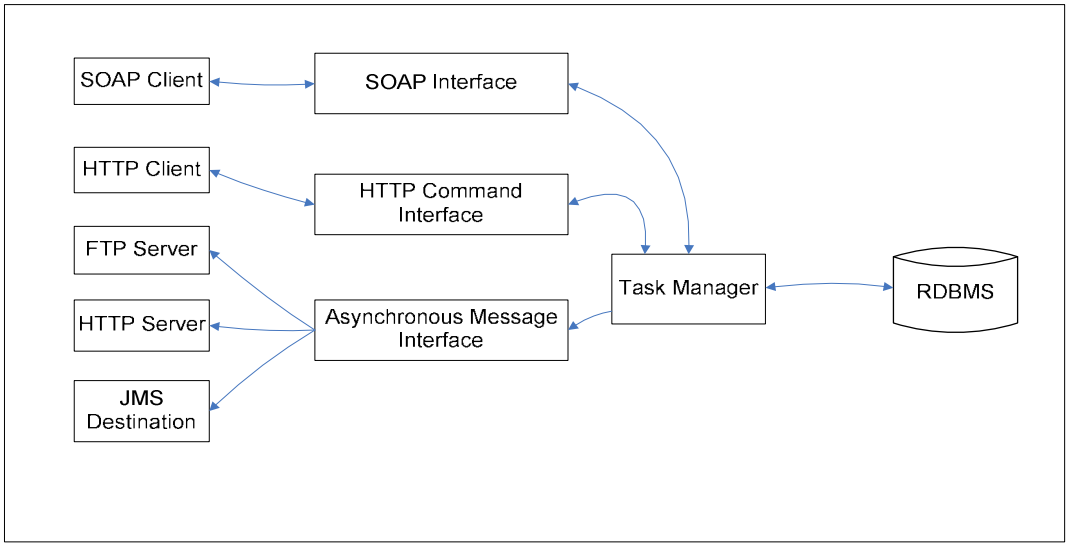
5.2 EMS 配置

Queue Size：Event Queue 的最大容量。部署人员应根据实际硬件的内存容量、数据库读写速度、Middleware 连接的 Device 的数目、通过 Device 带有标签的货品的流量设置合适的数值（5 个以内的 device 同时工作推荐值 1000，10 个以内的 device 同时工作推荐值为 2000）。

Queue Size 的参数存放在数据库表中，可在系统启动前直接修改它。

6 Task Management System

下图为 TMS 的系统框图：



6.1 SOAP Interface

SOAP 服务器让外部系统可以 SOAP RPC 与 Middleware 交互。它负责解析 SOAP 请求，把它传给任务管理器，接受任务管理器返回的结果，打包以 SOAP 的响应送回给外部系统。

6.2 HTTP Command Interface

HTTP XML 服务器用于通过 HTTP 协议接收外部系统发来的 XML 格式的命令，指示 Printer 执行打印标签及写 Tag 的操作，并把结果以 XML 的形式返回。

6.3 Asynchronous Message Interface

异步消息服务器接收任务管理器送来的消息把它写入到对应的目的地：本地文件目录、FTP Server、HTTP Server 或 JMS Queue/Topic。

6.4 Task Manager

Task Manager 响应接口服务器的请求，执行对应的处理逻辑检索数据库，把检索的结果返回给接口服务器。Task Manager 也负责管理/执行 TMS 任务。它要在系统启动时初始化内部的数据结构及启动若干任务线程，并提供接口供 Web Console 和 SOAP Interface 对任务进行控制。

7 Notification Message Format

Middleware 要支持多种 Notification Message 格式，传送读到的 Tag 信息给外部程序。系统定义了两种格式：

7.1 Full PML

Tag 信息的数据格式采用 PML Core Schema (Refer to PML Core Specification 1.0)。Element pmlcore:Command 可取用户在 Task 中定义好的字符串。下例给出在给出一个 Reader 读到多个 Tag 返回的 XML 字符串：

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
< pmlcore:Sensor>
    <pmluid:ID>Cargo Unloading Group</pmluid:ID>
    <pmlcore:Observation>
        <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
        <pmlcore:Command>IN</pmlcore:Command>
        <pmlcore:Tag>
            <pmluid:ID>30140242204031400000001</pmluid:ID>
        </pmlcore:Tag>
        <pmlcore:Tag>
            <pmluid:ID>30140242204031400000002</pmluid:ID>
        </pmlcore:Tag>
        <pmlcore:Data>
            <pmlcore:XML>
                <ReaderID>Reader_1</ReaderID>
            </pmlcore:XML>
        </pmlcore:Data>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

第一个<pmluid:ID>为定义好的Middleware/Device Group的名字，<plmcore:Command>为特定Reader/Task定义好的命令字符串。<ReaderID>可以存放deviceID或deviceName。<pmlcore.DateTime>采用W3CDTF的格式。

7.2 Simplified PML

简明 PML 格式即不采用 XML Namespace。下例给出在给出一个 Reader 读到多个 Tag 返回的 XML 字符串：

```
<?xml version="1.0" encoding="UTF-8"?>
<Sensor>
    <ID>Cargo Unloading Group</ID>
    <Observation>
        <DateTime>2002-11-06T13:04:34-06:00</DateTime>
        <Command>IN</Command>
        <Tag>
            <ID>30140242204031400000001</ID>
        </Tag>
        <Tag>
            <ID>30140242204031400000002</ID>
        </Tag>
        <Data>
            <XML>
                <ReaderID>Reader_1</ReaderID>
            </XML>
        </Data>
    </Observation>
</Sensor>
```

7.3 Other Observations

除了以上的包含读到Tag EPC/ID的XML格式外，Middleware要支持读取一个GTIN/SSCC标签来触发外部系统执行写标签操作，此时XML的格式为：

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
    <pmluid:ID>Device Controller</pmluid:ID>
```

```
    <pmlcore:Observation>
        <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
        <pmlcore:Command>PRNT</pmlcore:Command>
        <pmlcore:Data>
            <pmlcore:XML>
                <ReaderID>Bar_Code_Reader</ReaderID>
                <GTIN>0037000567394</GTIN>（或<SSCC>000370000000000266</SSCC>）
            </pmlcore:XML>
        </pmlcore:Data>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

注意这里把标签的ID放在<pmlcore:XML>下的<GTIN>或<SSCC>中而不是7.1的<pmlcore:Tag>下的<pmluid:ID>中，是由于标签的ID是标识特定公司特定Product的GTIN/SSCC，而不是标识唯一单品的SGTIN/SSCC。

8 User Access Control

Middleware 支持以下用户角色（Role）：

- 观察员(Viewer)：通过 WebConsole 登陆，他只能看到系统 Device 的当前状态及实时读取可访问 Device 所检测到 Tag 的信息；
- 操作员(Operator)：通过 WebConsole 登陆，他自动具有 Viewer 权限，可以启动/停止可访问 Device、启动/停止可访问 Task、执行手工 Task、通过可访问 Device 写 Tag；
- 外部用户(ExternalUser)：该角色较特殊，他不能通过 Web Console 登陆，外部程序通过他来登陆系统建立一个对话连接(Session)，通过外部接口与 Middleware 交互。他只能与可访问 Device 交互，但不能调用任务控制方法。
- 系统管理员(Administrator)：他是系统的超级用户，可以配置 EMS 和 TMS，对 TMS 任务、用户帐号、Device、Device Group 进行管理。外部程序可使用他建立对话连接。

建立一个用户帐号时，要赋给他一个用户角色。同时定义他所能管理/交互的 Device 的列表。

用户登录系统要进行验证，要支持加密的方式传递用户密码（HTTPS）。用户登录成功后，系统自动查找他的角色及所管理的 Device 列表，他只能进行权限所允许的操作及查看相关的信息。

9 External Interface

9.1 SOAP Interface

Middleware 支持外部系统以 SOAP RPC 同步访问。外部系统首先要登陆 Middleware 建立一个对话(Session)。SOAP RPC 的方法可分为三类：

系统登陆/退出/Session Health Check

1. String login(String user, String password) throws AccessDeniedException, SystemOperationException;

登陆 Middleware 建立起一个 Session，如成功，返回 Session 的唯一的标识符。外部系统要记住这个 Session ID，以后的调用都要给出此参数。

2. void logout(String sessionId) throws SystemOperationException;

结束对话。

3. void setFormatType(int formatType, String sessionId) throws InvalidSessionException, InvalidParameterException;

设置返回 PML Format 的选项：FULL or SIMPLIFIED(默认为 FULL)。

4. void setTopLevelIDType(int topLevelIDType, String sessionId) throws InvalidSessionException, InvalidParameterException, SystemOperationException;

设置返回 PML Top Level <pmlcore:ID> 的选项：MIDDLEWARE_NAME or DEVICE_GROUP_NAME(默认为 MIDDLEWARE_NAME)。

5. void setReaderIDType(int readerIDType, String sessionId) throws InvalidSessionException, InvalidParameterException, SystemOperationException;

设置返回 PML <ReaderID>的选项：READER_NAME or READER_ID(默认为 READER_NAME)。

6. void checkSessionHealth(String sessionId) throws InvalidSessionException, SystemOperationException;

当 Session 在一段时间内(30 分钟)没发生任何调用，Middleware 会自动结束这个 Session 即使外部系统没显式地调用 logout。如果外部系统还想保留这个 Session，就要调用 checkSessionHealth，Middleware 会重置 session timeout。

读 Tag 信息

7. String getTagInfo(String groupName, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

返回指定 DeviceGroup 中**多个** Reader 当前读到所有 Tag 的信息。Device Group 由组名唯一标识，Device 可由它的名字或 ID 码来识别，它们之间用逗号分隔，如 `name:device 1, id:a81934bf00000000`。sessionId 是以前调用 login 建立 Session 时，Middleware 返回的本次 Session 的唯一的标识符。

8. String getTagInfo(String startTime, String endTime, String groupName, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

返回指定 DeviceGroup 中**多个** Reader 在 startTime 到 endTime 期间读到所有 Tag 的信息。如果传入的 startTime 为 null，则表明从当前系统时间开始。

9. String getTagEvent(String startTime, String endTime, int triggerType, String groupName, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

返回指定 DeviceGroup 中**多个** Reader 在 startTime 到 endTime 期间所有发生 Tag Event 的信息。triggerType 可为预定义常数之一：Event.ADD、Event.REMOVE 或 Event.READ。

10. String getTagInfoByEmpty(String startTime, String groupName, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

从 startTime 开始，到没有任何 Tag 能被读到为止(没有 Tag 处在 Device 读取范围内)，返回指定 DeviceGroup 中一个 Reader 读到的所有 Tag 的信息。

11. String getTagInfoByEmpty(String startTime, String endTime, String groupName, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

从 startTime 开始，到没有任何 Tag 能被读到或 endTime 为止，返回指定 DeviceGroup 中一个 Reader 读到的所有 Tag 的信息。

12. String getTagInfoByIdle(String startTime, long idlePeriod, String groupName, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

从 startTime 开始，到经过空闲时段(毫秒)后，如果没有新的 Tag 被检测到为止，返回指定 DeviceGroup 中一个 Reader 读到的所有 Tag 的信息。

13. String getTagInfoByIdle(String startTime, String endTime, long idlePeriod, String groupName, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

从 startTime 开始，到经过空闲时段(毫秒)后，如果没有新的 Tag 被检测到或 endTime 为止，返回指定 DeviceGroup 中一个 Reader 读到的所有 Tag 的信息。

14. String getTagInfoByNumber(String startTime, int number, String groupName, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

从 startTime 开始，到已读到 number 个标签为止，返回指定 DeviceGroup 中一个 Reader 读到的所有 Tag 的信息。

15. String getTagInfoByNumber(String startTime, String endTime, int number, String groupName, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

从 startTime 开始，到已读到 number 个标签或 endTime 为止，返回指定 DeviceGroup 中一个 Reader 读到的所有 Tag 的信息。

写 Tag

只能在 Operator 建立的 session 中，才能调用 writeTag。

16. boolean writeTag(String label, String tagID, String deviceId, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

通过指定的一个 Device 打印标签及写 tagID。

任务控制

只能在 Administrator 建立的 session 中，才能执行任务控制方法。

17. void startTask(String taskName, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

启动一个任务。

18. void stopTask(String taskName, String sessionId) throws InvalidSessionException, InvalidParameterException, AccessDeniedException, SystemOperationException;

停止一个任务。

9.2 Asynchronous Message Interface

Middleware 支持异步地以事件触发方式发送包含 Tag 数据的消息给外部系统。消息可以是以文件形式写到本地目录下或 FTP 到另一台机器，HTTP POST 发送 XML 或发送到定义好的 JMS 的 Queue 或 Topic 中。(Note: 通过 POST 方法向 HTTP Server 发送 XML 时，HTTP Server 会返回 HTTP 200 OK response 通知已接收到 XML 消息)

异步消息接口由 TMS 的 Task Manager 控制。用户通过 Web Console 完成 Task 的配置及手工启动|停止。Task 支持以下消息触发模式：

- 事件触发：当 Device 检测 Tag 加入或 Tag 移出时，把 Tag 信息生成消息发出；
- 打包：用于支持货品打包。**By Empty 模式**：当 Device 检测不到任何 Tag 时，则认为一整包货品已经过了 Device；**By Idle 模式**：定义相邻包通过 Device 的最小空闲间隔，当超过 Idle Period Device 还没检测到新的 Tag 时，则认为一整包货品已经过了 Device；**By Number 模式**：定义整包货品的数量，当读完指定数量的 Tag 后，就把 Tag 信息封装在 XML 文档中送出；
- 打印触发：指 Bar Code 的 Reader 读到物品种类的条码时，发送把写标签的 PML；

9.3 HTTP Command Interface

外部系统可以连接到 Middleware 的 HTTP Command Interface Server 通过 POST 方法发送 XML Command 来指示 Printer 打印标签及写 Tag。HTTP Server 要求 Authentication，发送 XML 命令的外部系统传送 user name 和 password 进行身份验证(user 一定为 Operator 或 Administrator)。当验证错误，Server 返回 Status Code 401 。对于 Operator 他只能向他所能控制的 Device 发指令，否则返回 Status Code 403。

HTTP POST 发送 XML Command 的格式参考 SAP AII-DC Interface Description 1.0: 3.1 Command Message。

下面给出外部系统请求 RFID Printer: Writer_Device 写 EPC 码 3074024220403B8000000008 的 XML Command:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Command.xsd">
    <WriteTagData readerID="Writer_Device">
        <Item>
          <FieldList>
                <Field name="EPC">3074024220403B8000000008</Field>
            </FieldList>
        </Item>
    </WriteTagData>
</Command>
```

下面给出外部系统请求 Fixed Barcode Printer: Barcode_Printer 打印 GTIN 码 00037000567394 的 XML Command:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Command xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Command.xsd">
    <WriteTagData readerID="Barcode_Printer">
        <Item>
          <FieldList>
                <Field name="GTIN">00037000567394</Field>
            </FieldList>
        </Item>
    </WriteTagData>
</Command>
```

HTTP Command Interface Server 在接收到 XML Command 后应立即返回 HTTP 200 OK response，通知已接收到 XML 字符串。当 Printer 正确执行了写 Tag 操作后，Interface Server 向外部系统通过 HTTP POST 返回如下的 XML(外部系统 IP，Port 及 Path 是预先配置好的):

```
<?xml version="1.0" encoding="UTF-8" ?>
<pmlcore:Sensor xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pmluid:ID>DEVICE_CONTROLLER_NAME</pmluid:ID>
    <pmlcore:Observation>
        <pmlcore:DateTime>2004-11-12T02:00:00.762+01:00</pmlcore:DateTime>
        <pmlcore:Command>IN</pmlcore:Command>
        <pmlcore:Tag>
            <pmluid:ID>3074024220403B8000000008</pmluid:ID>
            <pmlcore:Data>
                <pmlcore:XML>
                    <EPCStatus>WS</EPCStatus>
                </pmlcore:XML>
            </pmlcore:Data>
        </pmlcore:Tag>
        <pmlcore:Data>
            <pmlcore:XML>
                <ReaderID>Writer_Device</ReaderID>
            </pmlcore:XML>
        </pmlcore:Data>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

The element EPCStatus may have the following values:

- 'WS' to indicate that the tag was written and verified successfully
- 'WU' to indicate that the tag was written, but not verified successfully

10 License Key Management

Middleware 需要正确的 License Key 才能启动。License Key 是根据 User Name 和 Expiration Date 加密计算出来的。

11 Web Console

Web Console 是对 Middleware 进行配置、管理及监控的页面，所有 Web 页面都要支持 internationalization，根据 Web Client 的 Locale 设置自动选择语言显示。初始配置支持中文(ZH-CN)和英文(EN)。

11.1 用户登陆页面

输入用户名密码进行登陆。要提供修改密码的链接。当前使用 BASIC Authentication，如有需要，可支持 DIGEST Authentication。

11.2 用户修改密码页面

供用户修改自己的密码。

11.3 Device 状态页面

检索系统配置的 Device 的当前状态列表，在有权限的情况下还可以选择特定 Device，执行启动/停止及显示 Device 当前检测到的 Tag 列表的操作。从已有的 Device List 中选择，并对其修改或删除，也可添加新的 Device。

11.4 Device 配置页面－添加/编辑

只可由 Administrator 访问。需要设置 Device 的名称、ID、连接类型(IP 或串口)、Device 类型(Reader/Printer)、IP 地址&端口号或串口名&波特率、可选的连接所需的 Login 名及密码、启动类型(Startup Type: Automatic or Manual)、命令字符串(用在 XML 消息中)、设备描述。同时要给出 Device 的 Plug-in，对于 Middleware 已默认支持的 Device，只要从 dropdown list 中选择即可；否则要在 dropdown list 中选择 User Customized，并额外提供其对应的 Device Plug-in 对应的类名(Class Name)。Device 的名称、ID、IP 或串口名为唯一。Device 名只可包括字母数字、下划线和空格。

11.5 Device Group 管理界面

只可由 Administrator 访问。列出所有的 Device Group，选择后可以编辑或删除，也可以添加一个新组。

11.6 Device Group 管理界面-添加/编辑

只可由 Administrator 访问。设置组名称及描述，绑定该组可以管理的 Device。组的描述可以为空，组名在系统中应为唯一，只可包括字母数字、下划线和空格。一个 Device 至少要归属于一个 Device Group。

11.7 用户帐号管理界面

只可由 Administrator 访问。列出所有帐号，选择后可以编辑或删除，也可以添加一个新的帐号。

11.8 用户帐号管理界面-添加/编辑

只可由 Administrator 访问。设置帐号名称、口令(至少 6 个字符)、所控制的 Device and/or Device Group、Role（Viewer/Operator/Administrator/ExternalUser）、帐号描述。用户名应为唯一，只可包括字母数字、下划线和空格。

11.9 RFID Reader View Tag 页面

在有权限的情况下显示所选的 RFID Reader 此时检测到的 Tag 列表。

11.10 Reader Monitoring 页面

在有权限的情况下显示所选的 Reader，实时动态地显示从监控开始后 Reader 所检测到的所有 Tag 的信息。

11.11 Print Tag 页面

Operator 在有权限的情况下选择特定 Printer，给出 Tag ID 的范围，支持一个一个经确认的 Item Print 及无需用户确认的自动 Batch Print。

11.12 Middleware Server 管理页面

只可由 Administrator 访问。配置 Middleware 的名字（可能用到 Notification Message XML 中），配置接口服务器：SOAP Interface, Asynchronous Message Interface 的启动模式(Startup Type: Automatic or Manual)。选择特定的接口服务器执行启动/停止操作。

11.13 HTTP Command Interface 管理页面

只可由 Administrator 访问。HTTP Command Interface 可同时和多个支持 Command XML 的外部系统通讯，对每个系统要指定其 IP、Port、Path 及外部 HTTP Server 是否要求 Authentication，如是还要提供 Logon user name 和 password。Middleware 使用它们建立 HTTP 连接回送操作的结果的 XML。当前使用 BASIC Authentication，如有需要，可支持 DIGEST Authentication，也可支持 HTTPS。

11.14 Task 管理页面

只可由 Administrator 访问。显示定义的 Task 的名称、状态及启动模式（自动或手动）。选择特定的任务修改配置或删除它，也可添加新的任务。执行任务的启动/停止操作(要求 Task 对应的 Device 一定已启动)。

11.15 Task 管理页面-添加/编辑

只可由 Administrator 访问。设置任务的名称、执行模式（自动或手动）、自动任务的启动模式（自动或手动）、自动任务的触发模式或手动任务的事件模式（ADD、REMOVE、READ）、Device Group、Reader 及消息目的地。

此外要定义任务对应的 Notification Message 的 XML 格式设置：消息类型(FULL/SIMPLIFIED)、命令字符串、ReaderID (deviceID/deviceName)、Top Level ID (Middleware Name/Device Group Name)。

自动任务的触发模式可为检测到新 Tag(Tag Detected)、检测到 Tag 移出(Tag Removed)、Packing By Empty、Packing By Idle、Packing By Number 和 Printing。当触发模式为 Packing By Idle，要定义空闲间隔（相邻托盘通过 Device 的最小时间间隔）；当触发模式为 Packing By Number，要定义整包货品需读到的 Tag 的数量。

一个任务要定义一个或多个消息目的地（最多 10 个），这样消息可同时发送到多个目的地。消息目的地可为本地文件目录、FTP Server、HTTP Server 或 JMS Queue/Topic。对文件目录，指定目录名；对 FTP，指定 FTP Server 的 IP、用户名、密码及目录；对 HTTP，指定 Server 的 IP、

Port、Path、Logon user name 和 password；对 JMS，指定 ConnectionFactory & Destination 的 JNDI 名及 Destination 的类型(topic or queue)。

11.16 手动打包页面

只可由 Operator 访问。Operator 选择可执行的手动任务，页面实时显示按照任务定义的设备读到的 Tag ID，用户可以点击 'Pack' 把这个读取周期的 Tag 信息生成 XML 发送给消息目的地，用户可以点击 'Clear' 开始一个新的读取周期。

11.17 License Key Management 页面

此页面显示系统相关的 License Key 信息，包括 Expiration Date、User Name、License Key (用户需要 User Name 和 Expiration Date 来得到他的 License Key)。用户可修改 User Name 和 License Key 以安装新的 License。

12 Development and Test Environment

12.1 Operating System

Windows 2000/XP, Linux 或 Unix

12.2 Database Server

Commercial Server: Oracle, MS SQL Server;
Open source/freeware: MySQL, MaxDB, PostgreSQL;

12.3 JDK/JVM

J2SE 5.0

12.4 Application Server

Apache Tomcat 5.5 or JBoss 4