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

Ubipass 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

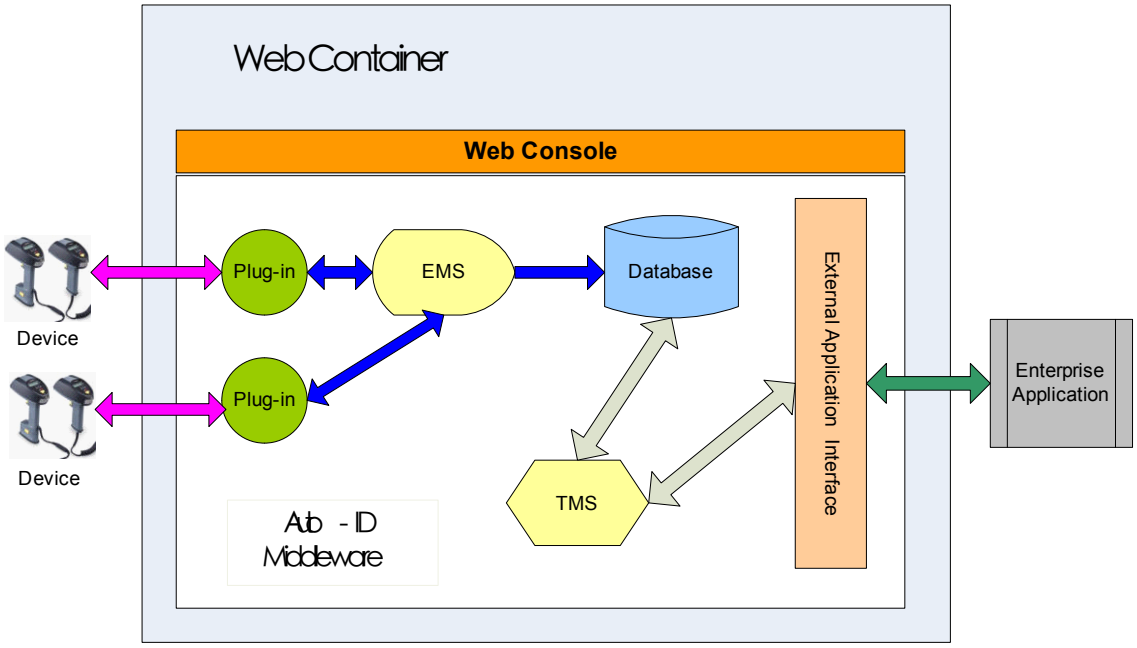
- Alien Reader Interface Guide & Java Developer Guide
- SAP Auto-ID Infrastructure 2.1 Interim Version
- SAP AII-DC Interface Description 1.0
- PML Core Specification, http://www.epcglobalinc.com/standards_technology/Secure/v1.0/PML_Core_Specification_v1.0.pdf
- EPC Tag Data Standards Version 1.1 Rev.1.24,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

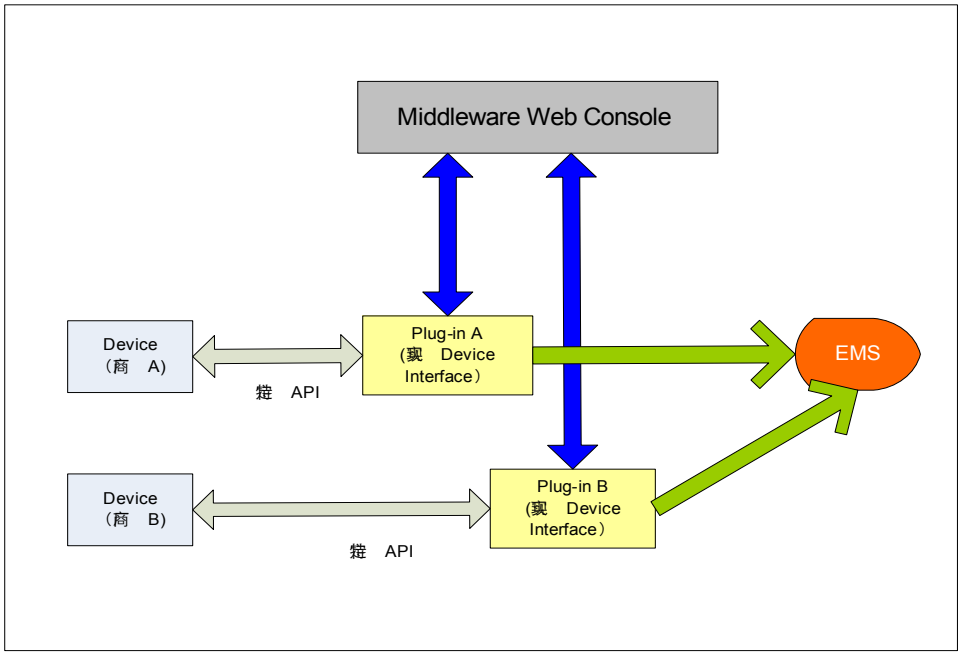
Middleware的体系结构如下图所示：



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.ubipass.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.ubipass.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.ubipass.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.ubipass.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 描述 eventID 事件类型	无	设置事件类型
setTagID	名称 描述 tagID 标签ID号	无	设置Tag的ID号
setReaderID	名称 描述 readerID Reader的ID	无	设置产生该消息的reader ID
setEventTime	名称 描述 eventTime 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。

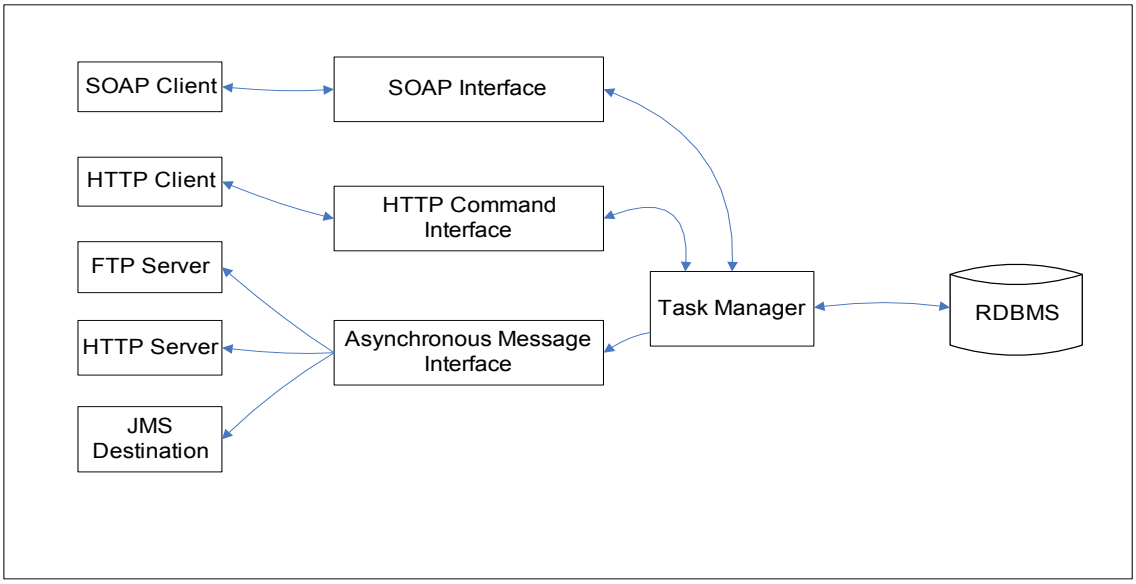
5.2. EMS配置

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