

Creating a RFID Data Integration Framework for Enterprise Information System

Xiaoyong Su, Chi-Cheng Chu, B.S. Prabhu, Rajit Gadh
University of California, Los Angeles
UCLA-WINMEC, RFID Lab
420 Westwood Plaza, UCLA, Los Angeles, CA 90095

Abstract

High volume, high speed, and heterogeneous inbound and outbound data format are characteristics of RFID-based Automatic Identification Data Capture infrastructure. Due to these characteristics, integrating and utilizing RFID data in current Enterprise Information Systems is challenging. A “store and forward” and rule-based subscription integration methodology to streamline RFID data integration is proposed in this paper. Application Level Event (ALE) subscription, an EPCGlobal RFID data integration standard, is analyzed against our proposed and implemented approach. Both integration mechanisms are implemented via a system termed the WinRFID Data Collector. A comparison between the integration methods is presented. From our research methodology, we conclude that the proposed approach is an effective way to process RFID data in a heterogeneous enterprise application environment. Technical limitations of the proposed approach are also presented and analyzed.

1 Introduction

Enterprises utilize advanced computing technologies to achieve efficiency, reduce costs, and increase productivity in their operations through the use of various enterprise applications such as Supply Chain Management (SCM) [18] systems, Customer Relationship Management (CRM) [19] systems, and Manufacturing Execution systems [20] [26]. Such enterprise applications generate and aggregate information about raw materials, products, equipment, shipments and personnel (collectively referred to as business objects), through their identity which is typically a number unique to each business object. The methodology of assigning an identity to a business object and automatically capturing the identity of the business object as it moves within and across an enterprise is called Automatic Identification and Data Capture (AIDC).

AIDC improves the efficiency and productivity of an enterprise application because it uses a unique identity to simplify the description of a business object and speed up information capture. For example, consider a Radio Frequency Identification (RFID) tag with an identity number (say “A00F3937ECBF56AE”) attached to a laptop that is being manufactured (assembled, packaged, inspected, checked-out). A RFID reader reads this tag at each inspection step and the identity number is used to lookup and retrieve detailed information about the laptop from a database. During this process the number only needs to be exchanged between enterprise applications. AIDC reduces the time required to gather information about the business object when compared to manual information entry and reduces human input errors.

1.1 AIDC Integration in Enterprise Application

Identification data captured by AIDC devices such as a RFID reader usually do not contain complete information about the business object or activity associated with the business object. In the above example, the laptop was identified by the number A00F3937ECBF56AE but not by its exact name or type. The process of obtaining information associated with the ID, such as product name, date of manufacture, expiration date, production factory, supplier name, customer name, is defined as Data Processing. Data Processing is typically performed by a software application called middleware, which performs the following:

- (i) Formats data – decodes the un-structured data (string or byte array) into structured data based on a predetermined coding scheme. For example, the number “A00F3937ECBF56AE” represents a 64-bit SGTIN number defined in EPCglobal, an organization that defines RFID standards and tag data standards [16]. The SGTIN number consists of six fields:
 - a. bit 0 to bit 1 is Header which indicates a 64-bit SGTIN number.
 - b. bit 2 to bit 4 is Filter Value which is used for identifying the tag types.
 - c. bit 5 to bit 18 is Company Prefix Index which is used to look up a Company Prefix registered to EPCglobal.
 - d. bit 19 to bit 38 is Item Reference which is used identify the type of an item.
 - e. bit 39 to bit 63 is Serial Number which is an unique number of the item in same type.Based on the SGTIN coding scheme, the number “A00F3937ECBF56AE” can be decoded into: Header = “2”, Filter Value = “2”, Company Prefix = “0052472”, Item Reference = “180315”, Serial Number = “4607652”.
- (ii) Transforms data – the structured data may still not be usable by the enterprise application because the enterprise application requires the data in a different form. For example, a SCM system may not require all fields contained in A00F3937ECBF56AE. It typically requires three fields: Company Prefix Index, Item Reference, and Serial Number. In this case, the extra data field contained in the number is discarded.
- (iii) Dispatches data – the middleware usually is not the final user/consumer of the identification data. The data needs to be sent to the enterprise application.
- (iv) Generates event – A single instance of captured data does not contain enough information to describe a business event such as fulfilling a particular order. The middleware shall have ability to aggregate a set of data combining with time and location to indicate a business event. For example, if the RFID tag number “A00F3937ECBF56AE” is read at dock door number 4 of warehouse B in company A, it may indicate, by way of a predefined rule, that this shipment is company C. Upon reading the RFID tag a transaction record called a business event is generated. Its corresponding process, such as sending a notification to company C, is then triggered.

1.2 RFID Integration Challenges

As RFID emerges as a new AIDC technology, the volume of data and the speed of data are significantly increased. This is due to the following:

- (i) An RFID system reads data 10-100 times faster than traditional barcode scanning system [22]. This requires that the data processing system be fast enough to process (formatting, transformation, dispatching, and association) such high data rates which traditional barcode based systems do not have.
- (ii) The ability to read at much greater distances and in region volumes larger than traditional barcode scanners results in greater amounts of data generated. For example, an RFID reader may be able to read all the tagged items within a carton in a fraction of a second provided they are within its antenna’s read range. This is evidenced by the seven terabytes of data generated daily by the Wal-Mart RFID trial [24]. This amount of data demands that the RFID based AIDC system be capable of capturing high volumes of data without loss.
- (iii) Most RFID tags have user memory to which data can be written and read. For example, the temperature of a shipment can be recorded on an attached tag at each shipping point, from the production point to the shelf of a retail store. To process the data in memory, actions such as comparing, searching, and pattern matching may be required. Memory-based tags increase the complexity of the data processing.

The increased data volume, data capture speed, and complexity of the data present a significant challenge to data processing and integration. To ensure lossless data capture and to provide intelligently processed data to the enterprise application, the “store and forward” approach is used in this research. High speed and high volume data is buffered in the local cache of the data capturing module – the Reader Coordinator (RC).

This buffered data is then processed by the data processing module – Data Collector (DC). This process is shown in Figure 1.

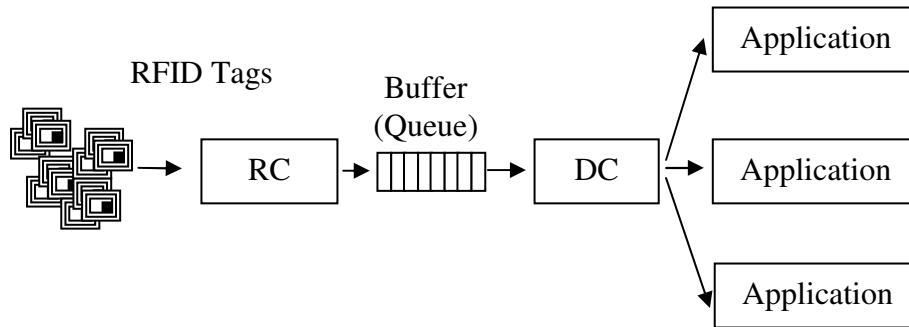


Fig. 1 Buffering via Queue between RC and DC

The DC collects identification data from the buffer, also defined as Data Source. It is the middleware which formats, transforms and dispatches data to the enterprise application.

2 Literature Review

2.1 Data Integration in Enterprise Application

The availability of multiple independent, heterogeneous data sources is a necessary part of an enterprise application. Different data sources need to be combined, aggregated, and provided in a unified data format to the user. As the data processing and integration middleware, the DC collects, formats, and aggregates RFID data from the data source and sends the data to the enterprise application. To understand the characteristics of the processing of RFID data, existing data integration methods were researched and analyzed.

Data integration is a process of providing formatted and transformed data to the enterprise application. Data integration has three roles to play as follows:

- (i) Data Source – It is the originator of data. It has various forms such as Database [11], XML documents [4], and Message Queue [3]
- (ii) Data Consumer - It is an application that requests data.
- (iii) Data Server - It is an application that provides data. It receives queries from Data Consumers and aggregates data from various Data Sources based on the queries condition. The result of the query is formatted, transformed and the returned to the Data Consumer.

As indicated in Figure 1, the DC is Data Server, Buffered data is the Data Source and Application is the Data Consumer.

Various data integration methods are utilized by the Data Server. Relational database query is the most researched data integration method. Various approaches exist for optimizing data integration such as uniformed query interface and adaptive execution [3], semantic integration [4], federated database system [5] [6], and global schemas with constraints [7]. However, in the RFID-based AIDC infrastructure, high volume data results due to the ability of the RFID reader to capture data at high speed data. Database operation involves frequent disk I/O operations, which slow down the speed of database functions such as inserting data and updating data. The performance of traditional databases typically can not keep up with the rate at which RFID data is generated. Because of this reason, we do not use database integration methods.

Recently Web Services for enterprise application integration [8] [10] [11] [12] [13] [14] has become a widely researched topic. A web service is based on standard protocols (XML based document format with

HTTP transport protocol) and provides a small autonomous unit of business function which is distributed over the internet to facilitate aggregation in a heterogeneous enterprise environment. It provides a platform independent integration mechanism since standard protocols are used. Using Web Services, an integration bus could be established to conquer the “spaghetti” [12] puzzle that traditional approaches usually entail.

Web Services for data integration has gained attention from researchers. For example, Zhu *et al.* propose a service-oriented data integration architecture (SODIA) [9], which provides a unified view of data from various data sources. The topic that they address in their research is that the data sources are unknown at design time. They use semantically described Web Service for data processing. SODIA can provide a dynamically unified data view from various antonymous, heterogeneous data sources. Hansen *et al.* use an aggregation model (data integration) [8] for the global provisioning system. In their research, Web Service is used to extract and integration business data in the heterogeneous business systems. Our thesis is that Web Service is the most suitable approach for a heterogeneous environment where various data sources and various applications are present. However, although the Web Service approach solves the challenge of integrating heterogeneous data sources with heterogeneous applications, it is not a real-time data notification integration approach. Since one of the advantages of RFID is that it provides real-time visibility because of its non-line-of-sight functionality, to utilize RFID effectively a real-time notification integration approach needs to be investigated.

2.2 Data Subscription - The Application Level Event Interface

Application Level Event (ALE) is an RFID data oriented processing and integration specification that defines how to accumulate, filter, and group EPC data and how to send (defined as “report” in ALE) the results to the client – the enterprise application which subscribes data to ALE. ALE interface [15] was proposed by EPCGlobal, a joint venture set up by Uniform Code Council (UCC), which licensed the Electronic Product Code (EPC) technologies developed by the Auto-ID Center, and European Article Number (EAN) International, the bar code standards body in Europe, to process EPC data and deliver data to enterprise applications. Two types of data delivery approaches are defined in ALE. One is based on “event cycle” model which returns data periodically. The other is the “poll” or “immediate” model which returns the data synchronously. Figure 2 shows the ALE data reporting mechanism.

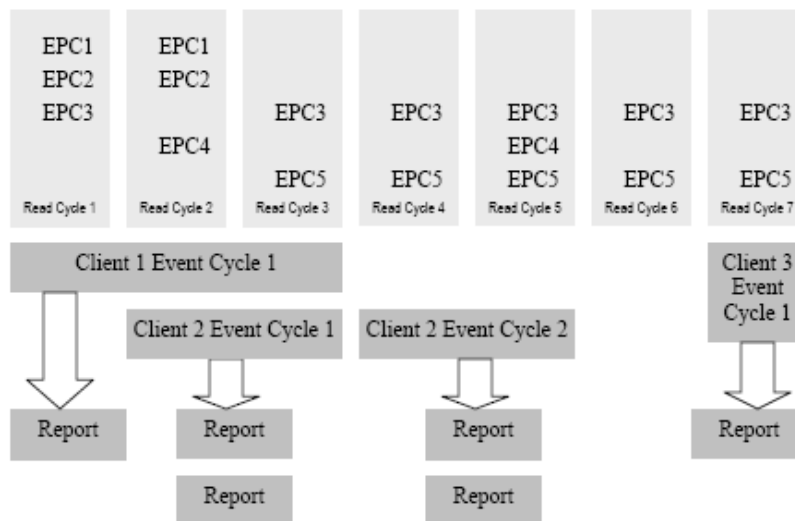


Fig. 2 ALE Reporting [16]

In the ALE, the “read cycle” is the smallest unit of interaction with a data source (such as reader). It may contain a set of EPC data as the result. The “event cycle” is the superset of the “read cycle”. It contains one or more “read cycles”. When the client subscribes to the ALE, it defines boundary conditions such as read

duration, report repeat period, triggers, and where to report etc. The boundary condition, along with filtration condition, grouping condition and output formats form ECSpec [16]. The ECSpec is registered to the ALE and is used as the criteria for data processing and dispatching.

The “poll” or “immediate” operation provides an interaction between the client application and the ALE. The “poll” operation executes an ECSpec that has already been registered on the ALE. The “immediate” operation needs the client to provide an ECSpec along with the request. Both operations will block the “event cycle” operation of the requesting client.

ALE currently serves as the standard EPC data integration interface. Most EPC-compliant middleware are claimed to have implemented the ALE specification. Since ALE is based on the EPC, it supports only the EPC-enabled applications. The data acquisition is “call-response” model or “call-once-response periodically” model. The ALE has several drawbacks which limit its applications. First, it is not suitable for the hybrid enterprise identification system where multiple identification technologies are used. Second, many applications are relying on ID only, without an encoding scheme. In this case ALE subscription process only increases the system complexity. Third, the ALE subscription is not an event driven process and therefore not suitable for real-time applications. To solve the limitations of ALE and to provide real-time data notification to the enterprise application, we propose a new subscription method which will be described later (called the WinRFID subscription mechanism) in this research.

3 Architecture

In the WinRFID network, identification data captured by the RC is stored in local cache in the form of message queue. This data needs to be aggregated and transformed before being delivered to the enterprise application. The DC is a data processing and integration middleware which takes charge of data formatting, data transformation and sending data to enterprise application. The reasons for separating the data processing and integration module (DC) from the data capturing module (RC) are:

- (i) The identification data capture process should capture data rapidly. As was discussed before, the data operations (formatting, transforming, and dispatching) are time consuming. If too many data operations are conducted in the RC, the RC may not be able to capture all the data coming in especially when it manages a large number of readers.
- (ii) Since the DC is an independent process, by running it on highly powerful CPUs it can handle complicated data processing without compromising the performance of the data capture module.
- (iii) By separating the functionality, the RFID network can be made more flexible and scalable because additional functional modules can be installed on demand.

Figure 3 shows the architecture of the DC. The DC collects data from multiple Data Sources which contain the identification data captured by the RC. The enterprise applications subscribe to the DC by sending ALE or WinRFID subscription condition – a set of rules which are used for data processing to the DC.

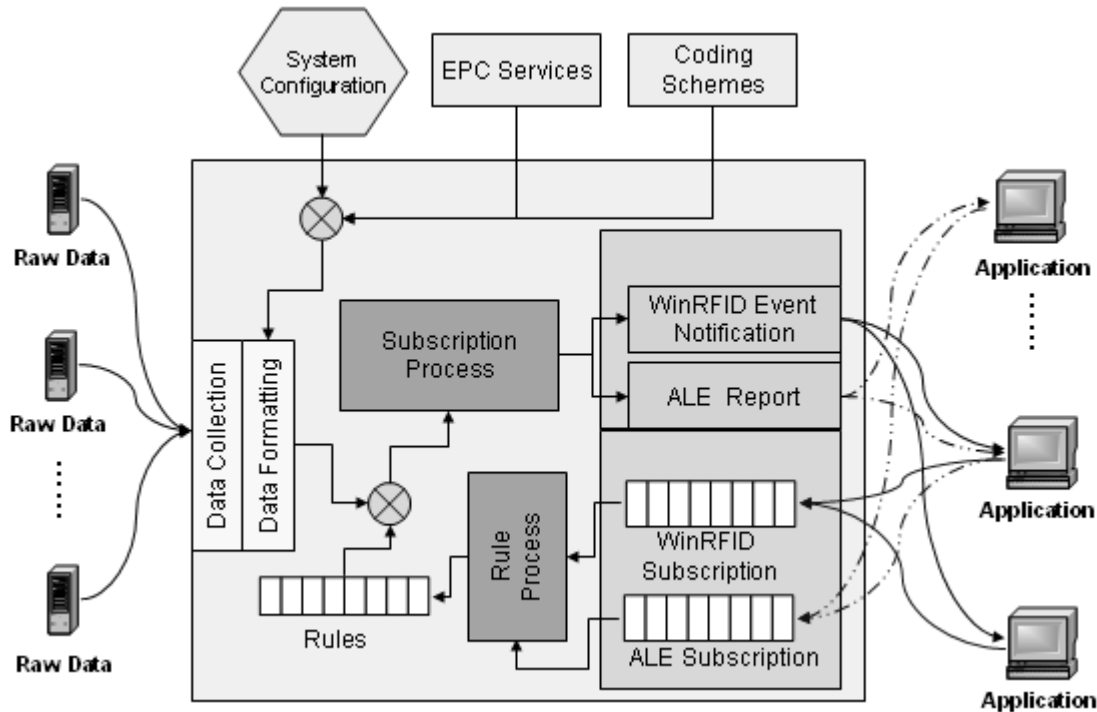


Fig. 3 Data Collector Architecture

Four major function modules are defined in the DC.

- (i) Data Collection and Formatting Module – this module extracts raw data from caches generated by RCs. The formatting module compares the pattern of the raw data, converts the raw data to the formatted data structure based on the coding protocol of the ID, and presents the formatted data as XML before sends the data to subscription/reporting module.
- (ii) Subscription/Reporting Module – Two types of subscriptions and reporting are proposed or implemented in the DC. They are:
 - 1) ALE subscriptions/reporting, which is the implementation of ALE specification. Data aggregation and reporting are based on “Event Cycle” defined in ECSpec.
 - 2) WinRFID subscription/reporting, which provides real-time reporting or “pushing”. The WinRFID subscription data processing is based on filtration patterns – a set of conditions which define how to filter irrelevant data and data format templates – pre-defined formats that guide the DC to transform data into desired format.

Two arrays of subscriptions are maintained in the DC.
- (iii) Subscription Processing Module – This is the core processor of the DC. The formatted data is processed against the subscription conditions and the rules.
- (iv) Rules Processing Module – Generates rules based on subscription conditions.

The Subscription Processing Module and Rule Processing are bound in both WinRFID Subscription and ALE Subscription. The data processing relies on the subscription condition which is defined by the client application (the enterprise application). Since the client decides how to process data and how to return data, the subscription based data processing is suitable for heterogeneous application environments where various applications are present.

4 Data Collection and Formatting

In the WinRFID-supported automatic identification application environment, the identification data captured by RC is stored in a local cache. The cache could be database, XML document, and message queue etc. Since the RCs are distributed across the enterprise physically and logically, the caches are spread across the enterprise as well. Caches are defined as Data Source. Relationships of the Data Source (DS) and the DC are represented by Figure 4. They could be “one to one”, which means one DC collects data from only one Data Source; “one to many”, which means one DC collects data from more than one Data Source; and “many to one”, which means more than one DC can collect data from one Data Source.

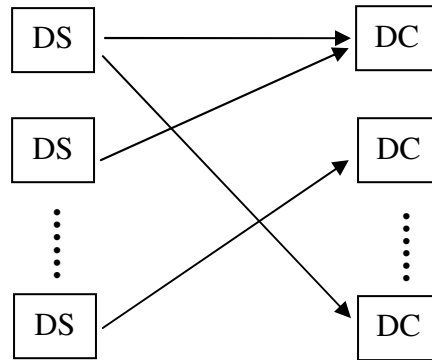


Fig. 4 The relationships between the Data Sources and the Data Collector
(DS represents Data Source and DC represents Data Collector)

The data stored in the DS is in the native formats (the data structure to accommodate the identification data) defined in WinRFID. It contains information including ID (RFID tag id), time (when the data is captured), location (where the data is captured) and content (non-structure data, system event and exception etc.). The data can be represented as a tuple (id, time, location, user_data). Once the data is extracted from the Data Source, the content (in form of string, number, or byte array) of the data is decoded into XML-based formatted data using the coding scheme defined in RC. The following xml represents an instance of formatted data,

```
<?xml version="1.0" encoding="utf-8" ?>
<Tag>
  <timestamp>2009-07-16T19:20:12+01:00</timestamp>
  <reader>rdr1.abc.com</reader>
  <id value="A00F3937ECBF56AE">
    <company_prefix>0052472</company_prefix>
    <item_reference>180315</item_reference>
    <serial_number>4607652</serial_number>
  </id>
  <user_data/>
</Tag>
```

5 Subscription and Reporting

Once the data is formatted, it is processed against the subscriptions. The DC supports both “push” data dispatching model – WinRFID subscription – and “pull” data dispatching model – ALE subscription. The WinRFID subscription pushes the data to the client application when the incoming data satisfies the subscription conditions and has been processed. The ALE subscription executes the subscription based on the time interval or trigger defined by the subscription conditions and returns the data periodically.

5.1 ALE Subscription and Reporting

The ALE subscription and reporting module is the implementation of the EPCglobal Application Level Event Specification [15]. The output is reported in the format defined by the subscription.

In ALE specification the filtration conditions are sets of “include” and “exclude” patterns. The EPC pattern structure [16] defined in the EPC Tag Data Standards is used to present the patterns. For example, “urn:epc:pat:gid-96:20.300.*” indicates all the General Identifier (GID) tags that has company id 20 (Domain Manager ID) and product id 200 (Object Class) belong to this category. Only the tags that satisfy both include patterns and exclude patterns will be reported to the client. Within the “include” patterns the result is the logical OR of the each pattern, within the “exclude” patterns the result is the logical AND of each pattern. The report could be a set of individual EPC tags or the summary of the accumulated result. The report can be grouped based on the specification. The grouping conditions are also a set of patterns represented by the pattern structures that defined in the EPC Tag Data Standards.

In terms of the mathematical format, the following relationship exists.

$$\text{Report}(I, E, G) = \{ F(I, E, T_i)|_{g_1}, F(I, E, T_i)|_{g_2}, \dots, F(I, E, T_i)|_{g_n} \} \quad (1)$$

Where T_i is the input EPC tag ids within an event cycle, I is a set of include patterns, E is a set of exclude patterns, and G is a set of group conditions. $F(I, E, T_i)|_{g_l}$ indicates that the tags that satisfy the filtration conditions are grouped together in group g_l .

$$F(I, E, T_i)|_{g_j} = g(T_i |_{((T_i \in i_1) \dots (T_i \in i_m)) \& (T_i \notin e_1) \& \dots \& (T_i \notin e_n)}) \quad (2)$$

Where i_m is a one include pattern, e_n is a one exclude pattern, function g is the group operation under group condition g_i . An example of subscription and reports of the ALE subscription/report can be found in Appendix 1.

5.2 WinRFID Subscription and Reporting

(1). Rule Based Subscription

The WinRFID subscription is based on the data types defined. It is rule-based and contains a set of “include” and “exclude” conditions. The client defines the subscription condition with composite rules. The rules are semantically organized to XML encapsulated format. The subscription is an event-based subscription, or “push” subscription. Data collected from a Data Source is checked against the client subscription conditions. If the data satisfies the conditions for a particular subscription, it is pushed to the client data processing module indicated in the URL. The rule based data checking is based on the boolean logic operation of a set of conditions. For example, the following conditions compose a rule to filter data:

```
<Conditions BL="AND">
  <Condition Expression = "TAG.id.company_prefix &eq 0052472"/>
  <Condition Expression = "TAG.id.item_reference &eq 180315" />
</Conditions>
```

TAG is a predefined data type. An instance of TAG is tag in this example. It represents the following logic: if the company prefix is equal to 0052472 and item reference is equal to 180315, then the tag satisfies the condition.

(2). Data Transformation

A significant challenge of integrating RFID data to enterprise information system is that diverse client applications usually require different data format. It is desired that RC can provide data format based on request from client. Extensible Stylesheet Language Transform (XSLT) and XQuery [17] can transform XML-structured data to another form of data based on pattern and template (or transformation rules). RC utilizes this feature of XSLT to allow a client control the output data format. The pattern and template that

XSLT engine used to transform XML data is called XSL stylesheet. The following is an example of a XSL stylesheet.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <Event>
      <xsl:for-each select="Tag">
        <tag_id><xsl:value-of select="id"/></tag_id>
        <location><xsl:value-of select="reader"/></xsl:choose>
        <time_stamp><xsl:value-of select=" timestamp"/></time_stamp>
      </xsl:for-each>
    </Event>
  </xsl:template>
</xsl:stylesheet>
```

Taking the formatted data example in section 4 as input, this XSL stylesheet is used transforms the formatted data into another form of XML format (or any other types). After the transformation, the following XML data results:

```
<Event>
  <tag_id>A00F3937ECBF56AE</tag_id>
  <location> dockdoor1.bulding1</location>
  <time_stamp>2009-07-16T19:20:12+01:00</time_stamp>
</ Event>
```

(3). Subscription Processing

Subscription is defined by the condition in XML form. Since data filtration, data transformation, and data delivery are three major functions in the subscription process, three sections exist in the subscription condition respectively. They are:

1. General definition: This section contains the definition. It includes data type and URL which indicate the location of the data processing module of the client application.
2. This section contains transformation rules. It could either be external such that only the pointer of the XSLT schema file is embedded or internal such that the whole transformation schema is embedded.
3. This section contains the conditions used for filtering data.

The following XML code is an example of a subscription.

```
<WinRFIDSubscription type="TAG"
receiverURL="http://localhost//process.aspx">
  <Transformation type="EXTERNAL">
    <value>http://translateserver.abc.com/Tramforms/tag.xslt</value>
  </Transformation>
  <SubscriptionCondition>
    <Conditions BL="AND">
      <Condition Express=" TAG.id.company_prefix &eq 0052472 "/>
    </Conditions>
  </SubscriptionCondition>
</WinRFIDSubscription>
```

The data that satisfies the subscription conditions will be sent to the URL indicated by receiverURL in the code. The data is formatted into XML before being sent to the data processing module of the client application. The following is example code of the data processing module of the client application. It is written in C# script language and hosted by the Microsoft Internet Information Service (IIS).

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.Web" %>
<%@ Import Namespace="System.Xml" %>

<%
    try
    {
        byte[] data = Request.BinaryRead(Request.TotalBytes);
        string s = System.Text.Encoding.Unicode.GetString(data);

        XmlDocument doc = new XmlDocument();
        doc.LoadXml(s);

        TAG tag = new TAG(doc.DocumentElement);

        //Pseudo code
        Process(tag);
    }
    catch (Exception ex){ Response.Write(ssss);}
%>
```

The client can format the XML data transferred from the DC into correct format because it defines desired data fields in the transformation condition.

6 Measure the Data Integration Capability of the DC

As mentioned earlier, a message-based “Store and Forward” data capture and process methodology is used. High volume data captured by the RC is buffered as locally cached data. The DC then reads the data from the local cache and then formats, transforms, and delivers the data to the subscribers. Although the performance of the DC would not affect the data capture capability, it is important to know the data integration capability of the DC so that the data processing can be balanced in the infrastructure by adding or removing DCs.

In the experiment, the DC runs on Dell Precision 650 workstation equipped with 2.0 GHz Xeon Processor and 1.0 GB RAM. The operating system is Windows Server 2003. The captured data is buffered in a message queue. Each record of data is a message. Four scenarios which cover all the possible combination of data processing are tested:

- (i) Data cache only without client subscriptions. In this scenario, the data is stored in a database.
- (ii) One WinRFID subscription. In this scenario besides being stored in the database, the data is formatted, transformed and delivered to the client based on the subscription conditions.
- (iii) One ALE subscription. In this scenario besides being stored to database, the data is formatted, grouped and delivered to the client based on the ALE ECSpec.
- (iv) One WinRFID subscription and one ALE subscription. In this scenario, both WinRFID subscription and ALE subscription exist.

For each scenario, the speed of data processing (i.e. how many reads per second) and the CPU usage are evaluated.

Figure 5 shows the number of messages that can be processed per second by the DC in a four-minute duration window. The average process speed of the DC is 100 messages per second. It was observed that both WinRFID and ALE subscription affect the process speed. In the evaluation, the average process speed was 107 messages/second under the condition of “no subscription”, 97 messages/second under the condition of “one WinRFID subscription”, 102 messages/second under the condition of “one ALE subscription”, and 90 messages/second under the condition of “one WinRFID subscription and one ALE subscription”.

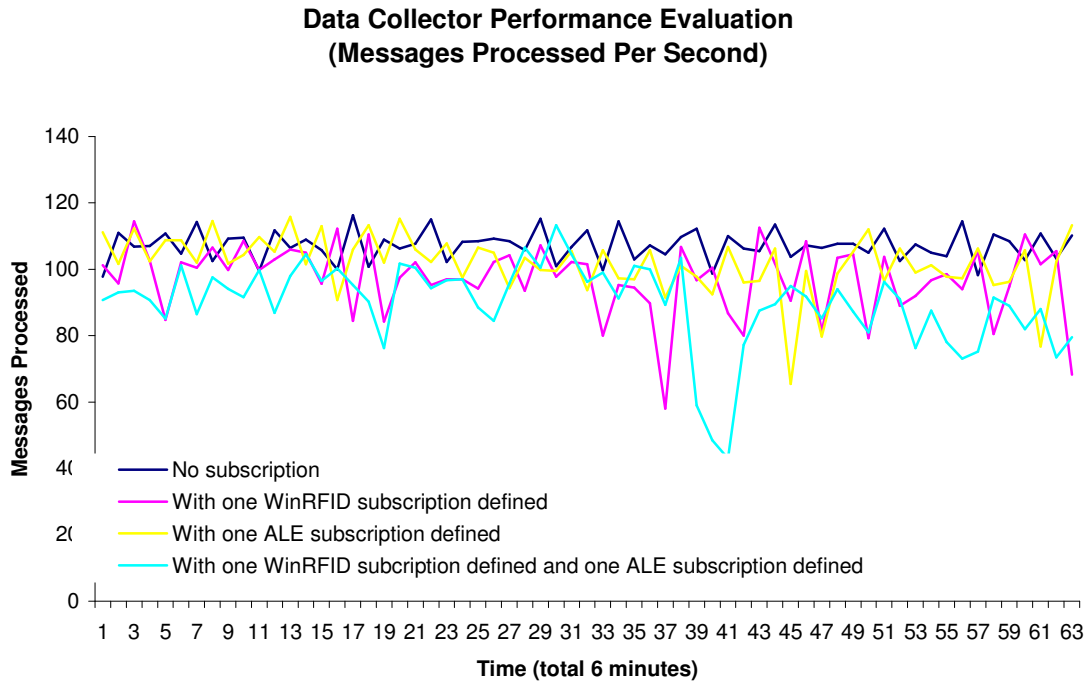


Fig. 5 Data Collector Performance Evaluation
(Messages Processed Per Second)

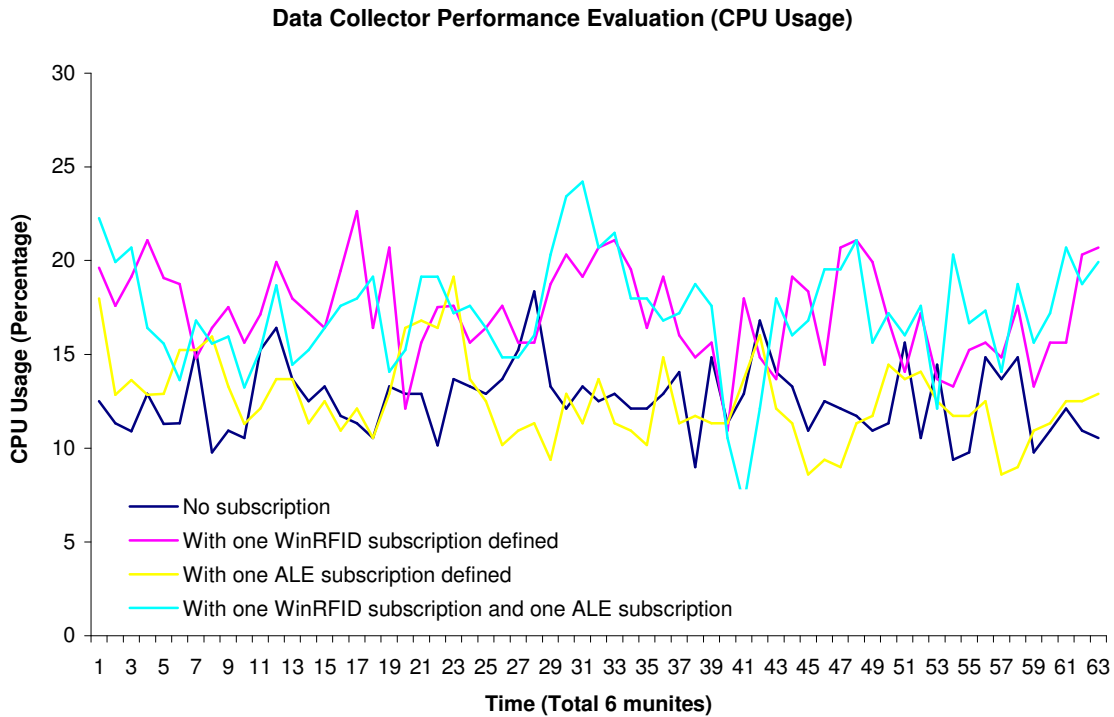


Fig. 6 Data Collector Performance Evaluation
(CPU Usage)

Figure 6 shows the CPU usage in a four-minute window. The average CPU usage is 12.6% under the condition of “No subscription”, 17.3% under the condition of “One WinRFID subscription”, 12.8% under the condition of “One ALE subscription”, and 17.4% under the condition of “One WinRFID subscription and One ALE subscription”.

From the experiment, we observed that the data processing takes a significant amount of time. More actions on the data entail fewer messages-processed-per-second and higher CPU usage. If the data capturing is combined with data processing in a single process, lossless data capture is difficult to achieve due to data processing blocking data capture. Experimental results prove that the buffering between the data capturing module and the data processing module is necessary. The results can be used as a reference for the design and use of the DC in the following manners:

- 1) In most cases, the high volume data capture at high speed happens within a short period. The buffering and processing approach used in DC is able to handle such data.
- 2) If high volume and high speed data capturing are required for an extended time period, multiple DCs should be utilized.
- 3) Data transformation in the subscription affects the processing speed. Complicated transformations should not be put in every subscription.

7 Summary

Identification technologies are increasingly being used to connect the business logic operation of an enterprise with the physical business objects in the enterprise I.T. systems to improve enterprise productivity and reduce the cost of operation. However, challenges of application integration exist as new and advanced AIDC technologies such as RFID are introduced. This is because RFID brings in substantially higher volumes of data with much higher speed of data capture, resulting in large data rates

that must be managed by the enterprise applications. To effectively utilize RFID data, we use a subscription-based data integration methodology and implementation of rule-based data processing – DC. The advantage of the subscription based approach is that it provides a generic way for supporting diverse enterprise applications. Each enterprise application defines its own subscription conditions using filtration criteria, event triggers, data transformation schema and the URL where the data will be processed.

Two subscription mechanisms are implemented in the DC. One is the ALE subscription which provides data to the client periodically. The other is a new subscription approach that provides real-time data processing and notification. This subscription also performs data transformation which allows a client to define the data format. The subscription condition generation and process, data reporting mechanisms and data representation format have been presented and discussed.

Experiments conducted on data integration capability of the DC indicate that data processing is the bottleneck when utilizing RFID for enterprise applications. We use “store and forward” buffering to eliminate the possible performance degradation of data capture affected by the data processing. It is observed that the data transformation is a significant factor which slows down the data processing. The data transformation in the WinRFID subscription causes a 5% percent CPU usage increase. At the same time 10% fewer messages are processed.

Although data capturing is not affected by data processing, the average data processing speed of 100 messages per second is still incapable of providing real-time data processing and delivery in a high volume and high speed data capturing environment. Several issues require further research specifically the following:

1. In most cases, the high volume data captured at high speed occurs within a short period. The buffering and processing approach used in the DC is able to handle such data.
2. If high volume and high speed data capturing are required for an extended time period, multiple DCs should be utilized.
3. Data transformation in the subscription affects the processing speed. Complicated transformations should not be used in every subscription.

Reference:

- [1] Hurt Geihs, Middleware Challenge Ahead, Computer, Volume 34, Issue 6, June 2001, pp 24-31
- [2] Wolfgang Emmerich, Nima Kaveh, Component technologies: Java beans, COM, CORBA, RMI, EJB and the CORBA component model, In the proceedings of the 24th International Conference on Software Engineering, Orlando, Florida, 2002
- [3] Zachary G. Ives, Daniela Florescu, Marc Friedman, An Adaptive Query Execution System for Data Integration, ACM SIGMOD Record, Volume 28, Issue 2, June 1999, pp 299-310
- [4] S. Bergamaschi, S. Castano and M. Vincini, Semantic integration of semistructured and structured data sources, ACM SIGMOD Record, Volume 28, Issue 1, March 1999, pp 54-59
- [5] R.J. Bayardo, W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, D. Woelk, InfoSleuth: agent-based semantic integration of information in open and dynamic environments, In the proceeding of the 1997 ACM SIGMOD international conference on Management of data, Tucson, Arizona, 1997
- [6] Joseph M. Hellerstein, Michael Stonebraker, and Rick Caccia, Independent, Open Enterprise Data Integration, IEEE Bulletin, Data Engineering, Volume 22, Issue 1, 1999, pp43-49

Accepted to International Journal of Internet Protocol Technology
(Accepted Sep 24, 2009)

- [7] Andrea Cali, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini, Data Integration under Integrity Constraints, In the proceeding of 14th international conference on Advanced Information Systems Engineering (CAiSE), Toronto, Canada, May 27-31, 2002
- [8] Mark Hansen, Stuart Madnick, Michael Siegel, Data Integration using Web Services, MIT Sloan School of Management Working Paper, Feb 10th, 2003
- [9] Fujun Zhu, Mark Turner, Ioannis Kotsiopoulos, Keith Bennett, Michelle Russell, David Budgen, Pearl Brereton, John Keane, Paul Layzell, Michael Rigby and Jie Xu, Dynamic Data Integration Using Web Services, In the proceeding of IEEE International Conference on Web Services, July 6-9, 2004
- [10] Yan Zhu, Christof Bornbovd, Alejandro P. Buchmann, Data Transformation for Warehousing Web Data, In the Proceeding of Third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems(WECWIS'01), 2001
- [11] L. M. Haas, R. J. Miller, B. Niswonger, M. Tork Roth, P. M. Schwarz, E. L. Wimmers, Transformation Heterogeneous Data with Database Middleware: Beyond Integration, IEEE Data Engineering Bulletin, 1999.
- [12] Christoph Bussler, The Role of Semantic Web Technology in Enterprise Application Integration, IEEE Data Engineering Bulletin, 2003
- [13] Michael Carey, Michael Blevins, Pal Takacsi-Nagy, Integration, Web Services Style, IEEE Data Engineering Bulletin, 2002
- [14] Mike P. Papazoglou, Service -Oriented Computing: Concepts, Characteristics and Directions, In the Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03), 2003
- [15] EPCGlobal, The Application Level Events (ALE) Specification, Version 1.0, EPCGLobal Ratified Specification, September 15, 2005
- [16] EPCGlobal, EPC Generation 1 Tag Data Standards Version 1.1 Rev.1.27, Standard Specification, 10 May, 2005
- [17] James Clark, XSL Transformations (XSLT), Version 1.0, W3C Recommendation 16 November 1999, <http://www.w3.org/TR/xslt>
- [18] Benita M. Beamon, Supply chain design and analysis: Models and methods, International Journal of Production Economics, Vol. 55 pp. 281-294, 1998
- [19] M. Stone, N. Woodcock, M. Wilson, Managing the Change from Marketing Planning to Customer Relationship Management, International Journal of Strategic Management: Long Range Planning, Volume 29, Number 5, October 1996, pp. 675-683(9)
- [20] Ronelle Russell, Manufacturing Execution Systems: Moving to the Next Level, Pharmaceutical Technology, January 2004, pp 38-50
- [21] N Erasala, DC Yen, TM Rajkumar, Enterprise Application Integration in the electronic commerce word, Computer Standards & Interface, Vol 25, Issue 2, May 2003
- [22] Andy Winans, Sink or Swim? As you dive into the depths of RFID, be sure to avoid drowning in data, RFID Journal, Dec. 12, 2005, <http://www.rfidjournal.com/article/articleview/2009/1/82/>

Accepted to International Journal of Internet Protocol Technology
(Accepted Sep 24, 2009)

[23] Evan Schuman, Will Users Get Buried Under RFID Data? Ziff Davis Internet, November 9, 2004, <http://www.eweek.com/article2/0,1895,1722063,00.asp>

[24] B. S. Prabhu, Xiaoyong Su, Harish Ramamurthy, Chi-Cheng Chu, Rajit Gadh, "WinRFID – A Middleware for the enablement of Radio Frequency Identification (RFID) based Applications", Invited chapter in *Mobile , Wireless and Sensor Networks: Technology, Applications and Future Directions*, Rajeev Shorey, Chan Mun Choon, Ooi Wei Tsang, A. Ananda (eds.), John Wiley, December 2005 .

[25] B.S. Prabhu, Xiaoyong Su, Charlie Qiu, Harish Ramamurthy, Peter Chu, Rajit Gadh, "WinRFID – Middleware for Distributed RFID Infrastructure", International Workshop on Radio Frequency Identification (RFID) and Wireless Sensors, Indian Institute of Technology, Kanpur, India, November 11-13, 2005.

[26] Zhekun Li, Rajit Gadh, and B. S. Prabhu, "Applications of RFID Technology and Smart Parts in Manufacturing", *Proceedings of DETC04: ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference* September 28-October 2, 2004, Salt Lake City, Utah USA.

APPENDIX 1

ALE subscription sample:

```
<?xml version="1.0" encoding="utf-8"?>
<ale:ECSPec xmlns:ale="urn:epcglobal:xsd:1"
  xmlns:epcglobal="urn:epcglobal:xsd:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:epcglobal:ale:xsd:1 ale:xsd"
  schemaVersion="1.0"
  creationDate="6/12/2006 9:04:03 PM"
  includeSpecInReports="False"
  specName="dock door processor">
  <logicalReaders>
    <logicalReader>rdr1.abc.com</logicalReader>
    <logicalReader>rdr2.abc.com</logicalReader>
  </logicalReaders>
  <boundarySpec>
    <startTrigger>http://epcserver1.abc.com/start1</startTrigger>
    <stopTrigger>http://epcserver1.abc.com/stop1</stopTrigger>
    <repeatPeriod unit="MS">5000</repeatPeriod>
    <duration unit="MS">1000</duration>
    <staleSetInterval unit="MS">2000</staleSetInterval>
  </boundarySpec>
  <reportSpecs>
    <reportSpec reportName="dock door results" reportIfEmpty="False"
reportOnlyOnChange="True">
      <reportSet set="CURRENT" />
      <filterSpec>
        <includePatterns>
          <pattern>urn:epc:pat:sgtin-96:3.0100110.*.*</pattern>
        </includePatterns>
        <excludePatterns>
          <pattern>urn:epc:pat:sgtin-96:3.0100110.213300.[100-
2000]</pattern>
        </excludePatterns>
      </filterSpec>
      <groupSpec>
        <pattern>urn:epc:pat:sgtin-96:X.X.X.*</pattern>
      </groupSpec>
      <output includeEPC="True" includeRawHex="True" />
    </reportSpec>
    <reportSpec reportName="All data" reportIfEmpty="False"
reportOnlyOnChange="True">
      <reportSet set="CURRENT" />
      <output includeEPC="True" />
    </reportSpec>
  </reportSpecs>
</ale:ECSPec>
```

ALE report example:

```
<?xml version="1.0" encoding="utf-8"?>
<ale:ECReports xmlns:ale="urn:epcglobal:xsd:1"
```

Accepted to International Journal of Internet Protocol Technology
(Accepted Sep 24, 2009)

```
    xmlns:epcglobal="urn:epcglobal:xsd:1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:epcglobal:ale:xsd:1 ale:xsd"
    schemaVersion="1.0"
    creationDate="6/12/2006 10:23:07 PM"
    specName="dock door processer"
    date="6/12/2006 10:23:07 AM"
    ALEID="virtualserver.abc.com"
    totalMilliseconds="24000"
    terminationCondition="TRIGGER">
<reports>
  <report reportName="dock door results">
    <group name="urn:epc:tag:sgtin-96:3.0100110.213200.*">
      <groupList>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213200.122233</tag>
        </member>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213200.126262</tag>
        </member>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213200.122666</tag>
        </member>
      </groupList>
    </group>
    <group name="urn:epc:tag:sgtin-96:3.0100110.213122.*">
      <groupList>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213122.126622</tag>
        </member>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213122.234683</tag>
        </member>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213122.166222</tag>
        </member>
      </groupList>
    </group>
  </report>
  <report reportName="All data">
    <group>
      <groupCount>
        <count>6</count>
      </groupCount>
      <groupList>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213200.122233</tag>
        </member>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213200.126262</tag>
        </member>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213200.122666</tag>
        </member>
        <member>
          <tag>urn:epc:tag:sgtin-96:3.0100110.213122.126622</tag>
        </member>
      </groupList>
    </group>
  </report>
</reports>
```

```
</member>  
<member>  
  <tag>urn:epc:tag:sgtin-96:3.0100110.213122.234683</tag>  
</member>  
<member>  
  <tag>urn:epc:tag:sgtin-96:3.0100110.213122.166222</tag>  
</member>  
</groupList>  
</group>  
</report>  
</reports>  
</ale:ECReports>
```