UAI-IOT Framework: a method of Uniform interfaces to Acquire Information from heterogeneous enterprise information systems

Hongyu Li\(^1\), Ye Tian\(^1\)*, Yang Liu\(^1\), Tingli Li\(^1,2\), Wei Mao\(^1\)
\(^1\)Computer Network Information Center, Chinese Academy of Sciences
\(^2\)University of Chinese Academy of Sciences
4, South 4\(^{th}\) Street, Zhongguancun, Haidian district, Beijing 100190, China
{lihongyu,tianye,liu-yang,litingli,mao}@cnnic.cn

Abstract—Enterprise information systems store great amount of information, which describes details of the objects in Internet of things. Due to these complex systems lack of uniform interface, users could hardly get the valuable information. We propose the UAI-IOT framework, which provides a method of uniform interfaces to actively acquire information from heterogeneous enterprise systems. To be compatible with these systems, the framework obtains information from log files and transformed the log information into semantic events, which could be acquired by any user through the interfaces of information service. Besides, integrating UAI-IOT framework and RNS would make contribution to share information among organizations. We also introduce a use case and make a comprehensive discussion.

Keywords—heterogeneity; log; Internet of things; information acquiring

I. INTRODUCTION

The Internet of Things (IOT) is a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols. [1] The primary purpose of IOT is to share the information of objects, which reflects the manufacture, transportation, consumption and other details of people’s life. Using the sharing information in IOT to feedback could effectively guide people to make decisions. At present, a large amount of such information is stored in heterogeneous enterprise information systems. However, lacking of uniform interfaces caused the valuable information of enterprise systems hardly be acquired by users.

In order to make the information easily be acquired, some researches have been done [2][3][4][5]. Nowadays, enterprises have to reconstruct their information systems and integrate these systems with middleware, so that they could bridge these systems to existing IOT architectures which provide interfaces for users to get information. However, not only would such a work costs a lot of efforts, but also information could be acquired only by specified interface of different IOT architectures.

Motivated by such challenges, we propose an effective method of uniform interfaces to acquire information from heterogeneous enterprise information systems, which is named UAI-IOT Framework. In order to avoid reconstructing information systems, we propose a novel solution from the perspective of the log file, because of its following features:

- **Real time** - Along with the operations of systems, log will be automatically generated and recorded in specified types of documents.
- **Rich** - Log files contain lots of information about the system logic which could reflect the state of the objects.
- **Semi-Structured** - Most information in log files are recorded by a uniform format as defined in the source code, so that it is easily to be parsed.

As depicted above, rich information makes log files could be regarded as the data source. Compared with the heterogeneous format of log files in different information systems, the complexity of reconstructing systems is much higher, which appears not only in the various programming languages and compiling tools, but also software architectures. In order to discover and deal with the business logic, we propose to handle the log information by Complex Event Processing (CEP) [6] technology, which could efficiently transform such information into semantic events. Finally, the framework publishes these events through information service, which provides uniform interfaces based on EPCIS specification [7] for any users. Besides, Integrated with Resource Name Service (RNS)[8][9], UAI-IOT Framework could contribute greatly for sharing information among organizations.

The reminder of this paper is structured as follows. In section 2, we present related work. Section 3 introduces the design of the framework. Data sharing mechanism is presented in Section 4. Section 5 introduces a use case and Section 6 makes a discussion and analysis. Section 8 presents the conclusion.

II. RELATED WORK

To make users easily acquire information of heterogeneous enterprise information systems, EPCglobal proposes a series of specifications [10]. Although they have specified uniform interfaces to exchange messages, enterprise systems could not actively publish information to EPC-network, but passively rely on driven by RFID events. That makes enterprises have to reconstruct information...
systems to integrate with RFID middleware so as to connect with IOT. Many researches focus on the integration of middleware and enterprise information systems, in order to reduce the complexity.

S. Kim [2] proposes a description method that can define not only the condition syntaxes of business rules as BESpec (Business Event Specification), but also the corresponding event refinement specifications. Although this method simplify the integration, users have to understand the structures of ECSpecs[11] to define the BESpec. Y. Liu [3] designs a four layers integration framework, which eases the event processing procedure by modeling the business process in the four layers correspondingly. But it doesn’t provide a uniform interface to obtain the event data. M. Li [4] provides a new method to communicate the enterprise information systems with document interpretation, which decreases the difficulty of reconstruction. Because of the heterogeneity of documents, it would tightly couple the systems and the middleware. Besides, one of the EU FP7 projects - AspireRFID[5] which tremendously reduce efforts to the integration, provides APDL[12] and connector[13], respectively makes business logic definition easily and offering a uniform interface for data exchanging. However, it also needs the enterprise information systems to reconstruct, so as to be compatible with the interface.

### III. DESIGN

In order to easily access heterogeneous enterprise information systems without reconstruction, we use Log Monitor to actively capture data from log files of the systems. With the operations on these systems, Log Monitor obtains new generated log records, and sends these record data to Event Generator. Receiving information of log files, Event Generator handle the data with CEP based on rule engine, which would eventually transform these raw data into semantic events. Each semantic event reflects the status change of single or multiple objects in the enterprise scenario. Then, these events would be sent to Information Service and stored in the repository. To provide better management of the semantic events, Information Service supports five kinds of extension mechanism. After that, users or applications could acquire the information through the uniform query interface of Information Service.

Figure 1 shows overview of the UAI-IOT Framework, which is designed to comprise three modules – Log Monitor, Event Generator and Information Service.

#### A. Log Monitor

To make the UAI Framework be compatible with various enterprise information systems, Log Monitor plays an important role, which is responsible for monitoring and capturing the information of log files. According to the operating system environment of information systems, we could achieve the goal of monitoring log files by specified programming language APIs. If detecting some new log records generated through APIs, Log Monitor would capture these information and send them to Event Generator.

The semi-structured of log records make it easy to be processed and filtered. According to the normal structure of log records, we define the log entity as follows:

\[ \text{Log} = \{\text{Date/Time, Level, Class, Method, Message}\} \]

“Date/Time” indicates the detail date and time when log information is recorded. “Level” represents the level of the log information which is specified in source code. “Class, Method” indicates the name of the source code class and method which records log information. “Message” indicates the detail information of log records which reflects the operations on system.

Each log record could be abstract as the definition above. Although the “Level” field and “Class, Method” field of some special log records would be null, after filtered and parsed in Event Generator, each log record would be the data source of basic event and complex event.

![Figure 1. Framework overview.](image)

### B. Event Generator

Event Generator is designed to transform the log information into semantic events. Receiving information from Log Monitor, Event Handler would filter the data stream and transform the data into log entity reports. Due to the features of log information, we choose CEP to process and correlate the log reports, which could process multiple streams of data continuously and identify meaningful events in real-time. CEP Manager passes log reports to Rule Engine, which is the key component of CEP. Rules of the engine are described according to the language defined by Drools. In order to make applications easily operate the rules, framework provides each rule a Restful interface. With the
uniform structured URI of Restful interface, applications could get the information of rules over HTTP.

CEP is the pivotal technology to implement semantic event generating, so that we will introduce the core concept of CEP - event and the crucial module of Event Generator - Drools Rule Engine in next two parts.

1) Event of CEP

Event is the basic definition in complex event processing. Generally, an event could be defined as an activity in the system and categorized into basic event and complex event. A Basic Event could be defined as below. “id” is the unique identifier of each event, “t” is the time when the event occurs, “a” is a set of event attributes.

E=E (id, t, a)

The data source of our framework is log files, so that a log entity is an instance of a basic event. The “id” field is the unique ID generated by the program, “t” is equals the “Date/Time” field of log entity as defined before, “a” is a set of event attributes including the “Level”, “Class.Method” and “Message” of each log entity.

A Complex Event could be defined as below. “id” is the unique ID of each event, “t” is the time when the last basic event occurs, “l” is the location that event occurs, “b” is the business step of the event, “a” is a set of event attributes.

E=E (id, t, l, b, a)

With the temporal logic and business rules, a set of basic events could be aggregated into a complex event via the event constructor. Complex events have much more semantic information than basic events, that’s why they are more helpful to the business decision.

2) Drools Rule Engine

Based on the definition of events, we need a reasoning rule engine to detect events and a set of syntax to describe the inference logic of rules.

As figure 2 shows the overview of Drools rule system, Drools stores rules in Production Memory. And the facts that the Inference Engine matches against are kept in the Working Memory. Actually, from the perspective of Drools, the event in CEP is a kind of special type of facts. Working Memory would assert the facts which may be modified or retracted. When facts and rules happen to be confliction, Agenda manages the execution order of these conflicting rules. Drools implement the Pattern Matching by the RETE algorithm [14]. Besides, in order to optimize the implementation, Drools not only enhances RETE algorithm with the implementation called ReteOO [15], but also combines the advantages of Leaps algorithm [16]. We will not introduce the details of the algorithms, since we focus on how we process the stream of log files by CEP model supported by Drools 5.0 open source software [17].

The most valuable information of a complex event stores in the “business step” field. According to the rules regulated in Production Memory, Drools engine effectively matches the pattern of basic events to infer the value of “business step”. As a result of expressing the complex logic in Drools rule language, it facilitates the framework decoupling with business circumstance.

C. Information Service

The primary vehicle for data exchanging between end users and the framework is Information Service (IS), which consists of Repository, Query Interface and Capture Interface. Event Generator would send the semantic events to Capture Interface, and the data of events would be stored in Repository. Then, any user could acquire information through the Query Interface.

In order to seamlessly connect to the most popular IOT architecture - EPC network, the design of IS follows EPCIS specification [7]. However, the five types of events – EPCIS Event, Object Event, Aggregation Event, Quantity Event, Transaction Event, which introduced in the specification, could not meet requirements in the scenario of heterogeneous enterprise information system. We design and implement five kinds of extensions in different dimensions.

- Event Type extension, which has the greatest impact on the system. Before handling new type of event, capture interface should receive a XML message as shown blow, which contains field names and corresponding field type of the new event. When the parse module of capture interface discriminates this kind of extension by the special node “<mode>” in XML message, repository would initialize new event tables in database.

- Event Field extension, which extends the basic elements of each event. Each event field defines a name and a data structure which specifies how to store information in the repository. We could implement event field extension by inserting new records in the table named “evnetype_extension”, which is designed specifically for field extension on each type of event.

- Vocabulary Type extension, which relates to each event field. The value of specified event field should be in the range of corresponding vocabulary elements. Each vocabulary type extension follows the extension of event filed, with initializing new vocabulary tables in database.

- Master Data Attribute extension. Master data attribute is a pair of “name-value” to describe the
element of vocabulary and it could be easily extended by inserting new corresponding attributes.

- **Vocabulary Element extension**, which specifies value ranges of the event field. It could be extension by inserting new records in corresponding "eventfield_vocabulary" table.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<epcis: EPCISDocument
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
creationDate="2008-03-06T11:42:15.016+01:00"
schemaVersion="1.0">
  <EPCISBody>
    <EventList>
      <extension>
        <Mode>create</Mode>
        <EventName>NewEventType</EventName>
        <eventTime>timestamp</eventTime>
        <eventTimeZoneOffset>String</eventTimeZoneOffset>
        <parentID>String</parentID>
        <childIDs>String</childIDs>
        <action>String</action>
        <bizStep>VocType</bizStep>
        <disposition>VocType</disposition>
        <transaction>VocType</transaction>
        <bizLocation>VocType</bizLocation>
      </extension>
    </EventList>
  </EPCISBody>
</epcis: EPCISDocument>
```

To meet different requirements of users, the Query Interface of IS could be divided into Query Control interface, which manages subscriptions to scheduled queries, and Query Callback Interface, which consumes data immediately. As the sample message mentioned above, both of the interfaces bind with SOAP over HTTP via a WSDL web services description. Applications and users could communicate with IS by the uniform format HTTP messages.

IV. DATA SHARING

The design of UAI-IOT Framework makes users easily get information from heterogeneous enterprise information systems through the interface of IS. Therefore, data stored in the repository of IS could be shared among different systems. For the purpose of sharing information across IS of different organizations, the framework need to be integrated with a name service, which provides a mapping relationship between an object identifier and an IP address of the server which stores data of the object in IOT. Any user who attempts to acquire information of an object could request the name service, and would get an identifier to addressing the IS according to name service afterwards. Using this server identifier would make users easily obtain information of objects through the interface of IS.

However, many countries and organizations have developed distinctive standards to name their own IOT resources, while the corresponding name service could only deal with their special naming schemes, such as ONS [18], uID [19], ID@URI [20], etc. To be compatible with such complex identifier standards, a two-stage identifier resolving mechanism was proposed, which is named Resource Name Service [8][9]. Motivated by sharing information across a higher level, we integrate the UAI-IOT Framework with RNS.

A. RNS Preliminary

To counter the heterogeneity problem of current identifications in IOT, RNS proposes a Two-stage object identification structure. The first stage is the Standard Identifier (SID), designed to define the naming rule. The second stage is Resource Identifier (RID), used to uniquely identify the resource.

Figure 3 shows the architecture of RNS. The SID name servers connect to each other by DHT in logic, and participate together for sharing the indexing information of resource identifiers with others, which each node only maintains a part of peer information. The RID name server is the same as hierarchical tree structure of DNS. This hybrid mode not only guarantees the efficiency of the frequent RID resolutions, but also ensures the equitability and fairness of the SID resolutions.

![Figure 3. Resource name service architecture.](image)

If an object identifier is the initial input of RNS, a naming scheme description will be returned after the first phase of SID resolution. With the help of such a description, the initial identifier can be transferred to an extended URI for the second phase of RID resolution. According to the types defined in service domain, users may get an IP address of the corresponding information server by a recursive processing.

B. Integrating UAI-IOT and RNS

In the internal network of enterprises, a variety of information systems, such as ERP (Enterprise Resource Planning) system, WMS (Warehouse Management System), CRM (Customer Relationship Management) system and SCM (Supply chain management) system, etc., manage the data of accounting, manufacturing, sales and service. The UAI-IOT Framework leads an easy way to share information among these systems.

In order to share the information for a higher level among organizations, we integrate RNS with UAI-IOT Framework.
to maintain the compatibility between heterogeneous object identifiers, as figure 4 shows. To complete the integration, the object identifier field of each semantic event should name in accordance with the two-stage structure. As events generating, each new object identifier would be registered in RNS name server, so that mapped with single or multiple information servers which store event data about the object.

With the integration, no matter what kind of resource identifier are taken as the original input, after the first stage of SID solution, a naming scheme description is known. Then all various RID can be translated into a unified resource name. So there is no need for an ordinary user to care about exactly what kind of naming schema is adapted for practical resource. Consequently, users would address the IS through requesting RNS the object identifier, and get the information from interfaces of IS as demands.

As Figure 5 shows below, the company has three facilities, four warehouses, and seven locations to put stuff.

ADempiere would generate a text log file each day when there are some operations on the system. There are nine log levels could be chosen in the system preference: “OFF; SEVERE; WARNING; INFO; CONFIG; FINE; FINER; FINEST; ALL”. To meet the requirement of our experiment purpose, we choose the option “ALL” to get all the information from log files.

V. USE CASE

Directed by the design of the framework, a use case about the integration of UAI-IOT Framework and ADempiere would be introduced below. ADempiere is one of the most popular open source ERP systems, with its comprehensive function, stable performance and well-applicability.

A. ERP initialization

In order to simulate the real situation of ERP system, we construct a virtual company called “CNNIC Fashion” which is an apparel company that manufactures a variety of clothes.
C. Running a business transaction

Finishing the initial configuration, we simulate a process to stock new cloth. ADempiere would process business steps, along with the operations of different users: merchandiser making purchase orders, warehouse keeper receipting cloth from the supplier and storing in LOCD1A location, financial officer signed up the delivery, receiving invoice and paid money.

Log Monitor captures log information stream from the log file “client2013-05-12_0.log” in real-time, encapsulates the information to log reports, and then sends these data to Event Generator.

Receiving the log data, event generator would process the information and generate events according to the rules. Rule engine matches the events so that contextualizes the raw data into semantic information. Finally, events data would be stored in the information service with the capture interface. The information and structure of the event would be as the sample message showed below. End users and applications could get the information through the uniform query interface.

```
<EPCISBody>
  <EventList>
    <TransactionEvent>
      <eventTime>2013-04-20T18:55:04Z</eventTime>
      <eventTimeZoneOffset>+08:00</eventTimeZoneOffset>
      <RNSID>cnnicfashion.niot.cn/32768:4369</RNSID>
      <action>ADD</action>
      <bizStep>cnnicfashion.niot.cn/bizstep/stocking</bizStep>
      <Disposition>cnnicfashion.niot.cn/disp/in_progress</Disposition>
      <bizLocation>cnnicfashion.niot.cn/location/LOCD1A</bizLocation>
    </TransactionEvent>
  </EventList>
</EPCISBody>
```

VI. DISCUSSION AND ANALYSIS

In this section we will have a discussion around the comparison among characteristic of different frameworks, and give a comprehensive analysis of the framework in the IOT architecture.

Regarding to the evaluation of artifacts and software mechanisms that comprise our framework, we present a comparison between AspireRFID and UAI-IOT Framework. We choose four different aspects to evaluate the frameworks, as shown in Table 1.

- **Compatible with heterogeneous network.** The bottom layer of Internet of things architecture generates large-volume raw data, the source may be RFID network, WSN network, enterprise information systems or the others. AspireRFID is designed to integrate with RFID network, while UAI-IOT Framework focuses on information systems in the internal network of enterprises.

- **Complexity of integration with enterprise information systems.** To the perspective of integration, enterprise systems have to modify the source code to connect with the modules of AspireRFID. It would give rise to high complexity of the integration and cost much more efforts. However, the UAI-IOT Framework just need to adapt the log files of the systems. Compared with the heterogeneous format of log files, the complexity of reconstructing systems is much higher, appearing in the various programming languages, compiling tools and software architectures, etc.

- **Scalability of various applications.** In order to provide convenience to communicate with applications, both of the frameworks have uniform interfaces. Aspire RFID follows the EPCIS specification, while UAI-IOT Framework supports five kinds of extensions on EPCIS, which would enables itself to exchange message with more applications.

- **Information sharing mechanism.** Aspire RFID is compatible with EPCglobal specifications so that it could integrate with ONS to sharing information across the EPC network. In order to solve the problem of identifier heterogeneity, UAI-IOT Framework which could integrate with RNS is able to share information among organizations across a higher level.

| TABLE I. COMPARISON BETWEEN ASPIRE RFID AND UAI-IOT |
|----------------|-----------------|
|                | AspireRFID | UAI-IOT |
| Compatible     | RFID       | Enterprise Intranet |
| Complexity     | High       | Low      |
| Scalability    | Low        | High     |
| Information sharing mechanism | Low | High |

From high-level perspective of the IOT architecture, we make an analysis on the functionality between EPC network architecture and the architecture of integration UAI-IOT Framework with RNS. From bottom to top as Figure 6 shown,

- **Enterprise Information Systems and EPC-based RFID network** are both the data source of the architecture. Enterprise information systems generate the business data stream recorded by log files, while EPC-based RFID network generate RFID data stream.

- **Log Monitor and Filtering&Collection Module** are responsible to clean and filter the raw data in respective architecture.
- Event Generator and Capturing Applications would process the source data and transform them into application level events.
- Information Service and EPCIS store the events information and provide both query interface and capture interface to exchange information with applications.
- RNS and ONS are the name service in corresponding architectures. ONS resolve identifiers only named by EPCglobal specification, while RNS provide a two-stage resolving mechanism to be compatible with various specifications.

As the discussion and analysis above, UAI-IOT Framework is a better choice to integrate enterprise information systems with IOT architecture. Although there are some differences between UAI-IOT architecture and EPC architecture, the modules on each level of the architecture are functionally equivalent.

VII. CONCLUSION

The heterogeneity of enterprise information systems brings an enormous challenge for data sharing in Internet of things. To make users easily acquire information from these systems, this paper proposed the UAI-IOT Framework.

UAI-IOT Framework could actively capture information from the log files to reduce the heterogeneity of enterprise systems. Instead of reconstructing the systems to add interfaces for data sharing, the framework not only reduces the expenditure, but also extends the lifecycle of these systems. The CEP module makes UAI-IOT Framework efficiently transform the log data into semantic events and decouple with business logics. Besides, the IS provides users and applications uniform interfaces to get data. To enable information sharing between different organizations, we explored a mechanism of integrating our framework and the RNS which can offer us a compatible name service.

ACKNOWLEDGMENT

This work is supported by the Public IOT Identity Service Platform project of Foshan Administration of Science & Technology under grant 2011BY100363.

REFERENCES