EIOP - An E-Commerce Interoperability Platform

Yusuf Tambag
Kara Harp Okulu, Dekanlık, OBS Şube
Ankara, Turkey
yusuf@kho.edu.tr

ABSTRACT

Interoperability has become one of the big problems of e-commerce since it was born. A number of B2B standards like ebXML, UDDI, RosettaNet, xCBL, etc. emerged recently to solve the interoperability problem.

Currently, there exists many B2B standards each provide competing and complementary solutions to B2B interoperability. So, there is a need for serving implementation of these standards from a single, central store to ease the use and management of the implementations. This paper presents EIOP, an E-commerce Interoperability Platform. EIOP is designed to provide a central store for implementations of e-commerce specifications to be able to use and configure these implementations from a single, central point. It defines the term *EIOP Component* which corresponds to plug&play e-commerce applications that are stored in the EIOP.

Categories and Subject Descriptors: K.4.4 [Computers and Society]: Electronic Commerce

General Terms: Standardization

Keywords: E-commerce, interoperability, ebXML, UDDI, ebIOP

1. INTRODUCTION

Currently, many specifications appeared that provide different solutions for various e-commerce problems. As a result, companies tend to support more than one specification. By supporting more than one specification, a company will be able to combine complementary aspects of the specifications to provide better services and the availability of the services supported by the company will be increased (i.e., if implementation of a specification is unavailable, the company may use the implementation provided by another specification that provide solution for the same problems). So, there is a need for an engine that will store and serve implementations of these specifications from a single, central point. The engine should also be easily customizable to enable addition/removal of these implementations. EIOP aims to provide implementation of such an engine.

EIOP is extension of ebIOP [1] which provides such an engine for ebXML.

2. SYSTEM ARCHITECTURE

EIOP contains a server that listens to a port and a web based user interface from which one can manage (add/remove/change/list) and use the EIOP Components over the web. Figure 1 shows the system architecture. EIOP contains one or more EIOP Components which are implementations of e-commerce problems or specifications. EIOP receives messages with the help of its JAXM (Java XML Messaging) compliant message handler over the HTTP or TCP/IP. EIOP server (which resides most probably inside the company firewall) receives messages with the help of its TCP/IP compliant message handler. The JAXM compliant message handler routes the messages coming from the other EIOP servers (which are outside the firewall) to the messaging handler that resides in EIOP server using TCP/IP.

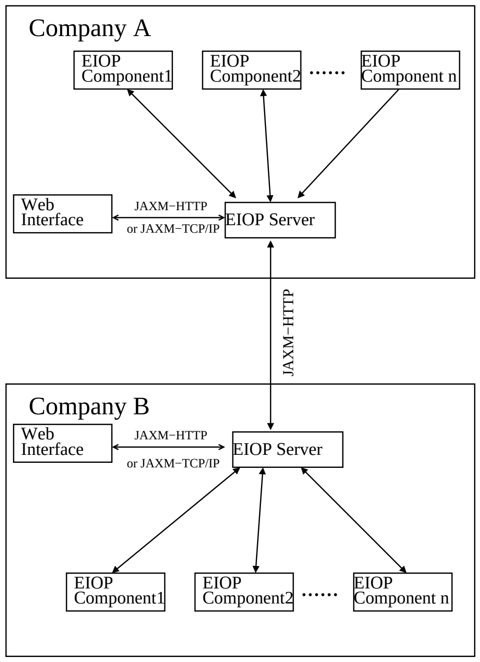


Figure 1: EIOP Architecture.

EIOP communicates with the web interface and EIOP's of other parties using a standard messaging service called EIOP MS (EIOP Messaging Service). EIOP MS defines a set of SOAP Header and body elements with in the SOAP Envelope. Figure 2 shows an example request message. As shown in the figure, the header part contains message identification and processing data that explains how the message should be processed at the receipent. The body part contains the information about the attachments, and the first attachement of the message contains the EIOP request or response message. The message shown in the figure requests the list of components on the EIOP Server.

EIOP messages are processes by a messaging handler named

```
<SOAP:Envelope xmlns:SOAP=
 "http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP:Header>
  <pmh:MessageData message.type='platform.info'
    processor.class='metu.ceng.e102830.iop.
    platform.components.pinfo.
    PlatformInfoMessageProcessor'>
    <pmh:MessageId>Msg001</pmh:MessageId>
  </pmh:MessageData>
 </SOAP:Header>
 <SOAP:Body>
  <pmb:Manifest>
   <pmb:Reference id='Attachment1'/>
  </pmb:Manifest>
 </SOAP:Body>
</SOAP:Envelope>
--boundaryValue
Content-ID: <Msg001@sender.com>
Content-Type: text/xml; charset='UTF-8'

<PlatformRequest>
 <PlatformServerRequest crq:request.type='LIST'>
 </PlatformServerRequest>
</PlatformRequest>
```

Figure 2: An example EIOP Message.

EIOP MSH. When a message is received by the EIOP MSH, the header part of the message is processed to find a suitable message processor for the message type. Currently the EIOP has two types of messages namely *platform.info* and *platform.component*. The first message type requests information or modification from the EIOP (e.g., list/add/edit/remove EIOP Components). The other type of message is targeted to an EIOP Component and it is further processed by the component itself.

2.1 EIOP Component

An EIOP component is an e-commerce application which is an implementation of an e-commerce problem or specification. In EIOP, the GUI that is used to manage the components in ebIOP [1] is maintained as an option. Figure 3 shows this GUI. Specific user interfaces of the components could be launched by using the *Components* menu as shown in the figure.

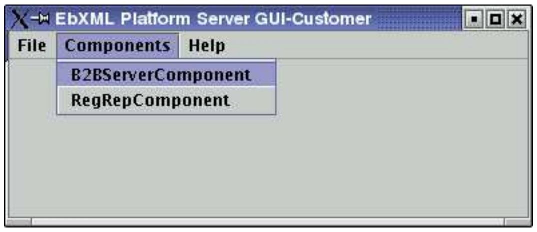
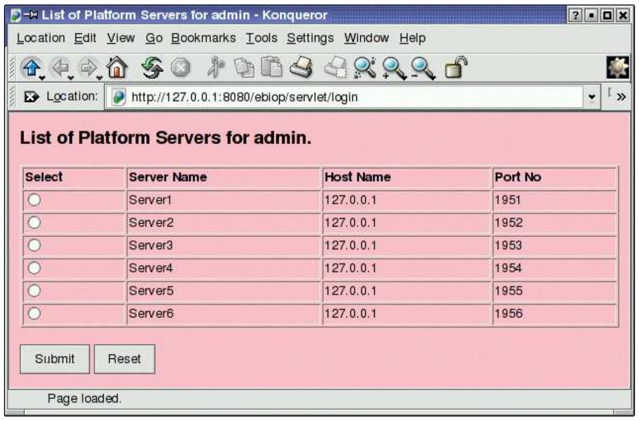


Figure 3: EbIOP GUI.

One can also use EIOP Components over the web interface as shown in Figure 4. As shown in the figure, a user may access more than one EIOP Server from the same interface.

EIOP Components that resides in an EIOP Server is provided in the configuration file of the EIOP Server. Each EIOP Component must implement two methods namely *initialize()* and *processMessage(PlatformMessage)*. The *initialize()* method is called by the EIOP Server when after the server reads its configuration file to initialize the component. This method should read the configuration file of the component and make necessary initialization specific to


List of Platform Servers for admin - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://127.0.0.1:8080/ebiop/servlet/login
Submit
Reset
Page loaded.

List of Platform Servers for admin.

Select	Server Name	Host Name	Port No
○	Server1	127.0.0.1	1951
○	Server2	127.0.0.1	1952
○	Server3	127.0.0.1	1953
○	Server4	127.0.0.1	1954
○	Server5	127.0.0.1	1955
○	Server6	127.0.0.1	1956

Figure 4: Web Interface.

the component. The *processMessage(PlatformMessage)* method is called by the EIOP MSH of the EIOP Server. When a message targeted to a component is received by the MSH, the component is located from the EIOP Server and *processMessage(PlatformMessage)* method is called by the MSH. The message is further processed by the component and result (if any) is returned to the MSH.

EIOP Server communicates with EIOP Servers of other parties and with the web interface using a standard language called EIOP ML. EIOP ML is an XML based language and contains two message types namely *PlatformRequest* and *PlatformResponse*. User requests are converted into a *PlatformRequest* message by the web interface and sent as an attachment in a SOAP message to the EIOP Server for processing.

After the message is processed, the result is returned as a *PlatformResponse* message to the web interface and the message is converted into suitable format for requesting device (e.g., WML for PDA's, HTML for web browsers, etc.) by a suitable XSL document. The conversion process is done by a suitable *ResponseProcessor* at the web interface. According to the type of the response message and the requesting device, suitable *ResponseProcessor* is loaded and response document is converted to the desired format.

3. CONCLUSIONS

EIOP provides a common platform for e-commerce applications to make businesses easily integrate and use e-commerce applications from a single central point. EIOP achieves this by providing a common base class namely *EIOPComponent* which is an e-commerce application or implementation of an e-commerce specification.

The platform could easily be customized by making a small change in the configuration. The implementors could replace or add/remove EIOP Components to/from the EIOP easily by using the web interface or by modifying the configuration. The software is currently being developed using JDK 1.4.2 with generics and jakarta tomcat 5.0.18. After the EIOP core is completed, ebXML and UDDI specifications will be integrated into the EIOP as EIOP Components to compare the two standards.

4. REFERENCES

[1] Tambag, Y., Cosar, A., "Managing business lifecycle using ebIOP", Journal of Software Practice and Experience, Volume 33, Issue 13 (p. 1217-1227).