COMPARING DOCUMENT OBJECT MODEL (DOM) AND SIMPLE API FOR XML (SAX) IN PROCESSING XML DOCUMENT IN LEAVE APPLICATION SYSTEM

Juliana Wahid

Faculty of Information Technology

Universiti Utara Malaysia

2005
ACKNOWLEDGEMENTS

In the Name of Allah, the Most Gracious, the Most Merciful.

I would like to thank:

The Faculty of Information Technology for the financial support under the faculty grant,

My husband for all the support and encouragement he has given me over the years,

Finally my colleagues for all their support and assistance.
ABSTRAK

Kewujudan produk Pengurusan Aliran Kerja (WFMS) yang banyak di dalam pasaran membolehkan pengguna menggunakan produk tertentu untuk memenuhi keperluan aplikasi yang khusus. Walau bagaimanapun, ciri-ciri sedia ada WFMS tidak dapat menyokong kebolehoperasian antara produk WFMS. Kegagalan untuk kebolehoperasian ini boleh menyebabkan berlakunya pulau-pulau otomasi proses yang tidak serasi di dalam organisasi atau pun di antara organisasi.


Penyelidikan ini dijalankan adalah sebagai langkah untuk mengukuhkan serta memperbaiki dapatan daripada penyelidikan yang telah dijalankan sebelum ini. Sebagai tambahan kepada penyelidikan sedia ada, penyelidikan ini akan menggunakan satu lagi piawaian daripada XML iaitu Simple API for XML (SAX) sebagai ganti kepada piawaian DOM di dalam aplikasi system cuti yang telah dibangunkan.

Pada peringkat akhir penyelidikan ini, ujian dijalankan ke atas aplikasi system cuti yang menggunakan piawaian DOM dan juga SAX. Antara item ujian yang dilihat ialah masa antara penghantaran permohonan cuti dengan kelulusan. Penggunaan SAX menunjukkan masa antara penghantaran permohonan cuti dengan kelulusan adalah lebih rendah berbanding penggunaan DOM. Masa yang lebih rendah menunjukkan penggunaan piawaian SAX dapat meningkatkan prestasi aplikasi.
ABSTRACT

The Extensible Markup Language (XML) consists of two Application Programming Interface (API) i.e. Document Object Model (DOM) and Simple Application of XML (SAX). Previous research has used the DOM API in the Leave Application prototype. In this research, the DOM API based code that developed in the prototype will be changed to SAX API based code.

The performance measurement then is carried out to evaluate the used of DOM API and SAX API in the prototype. The evaluation process took place within in each API and also combination of both APIs.

From the evaluation process, it was concluded that the use of combination of SAX based code in Domino Workflow environment and DOM based code in Microsoft Exchange 2000 Server produces the lowest processing time that will increase the performance of the leave Application system.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE PAGE</td>
<td>i</td>
</tr>
<tr>
<td>DISCLAIMER</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Problem Statement</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.5 Summary of Contributions</td>
<td>2</td>
</tr>
<tr>
<td>1.6 Thesis Outline</td>
<td>3</td>
</tr>
<tr>
<td>CHAPTER 2 LITERATURE REVIEW</td>
<td></td>
</tr>
<tr>
<td>2.1 Extensible Markup Language (XML)</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Document Object Model (DOM)</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Simple API of XML (SAX)</td>
<td>8</td>
</tr>
<tr>
<td>2.4 DOM versus SAX</td>
<td>10</td>
</tr>
<tr>
<td>CHAPTER 3 IMPLEMENTATION AND EVALUATION</td>
<td></td>
</tr>
<tr>
<td>3.1 Analyze and Design the System</td>
<td>12</td>
</tr>
<tr>
<td>3.1.1 Domino Workflow Environment</td>
<td>13</td>
</tr>
<tr>
<td>3.1.2 Microsoft Exchange 2000 Server Environment</td>
<td>16</td>
</tr>
<tr>
<td>3.2 Observe and Evaluate the System</td>
<td>20</td>
</tr>
<tr>
<td>3.2.1 Within each APIs</td>
<td>20</td>
</tr>
<tr>
<td>3.2.2 Combination of APIs</td>
<td>21</td>
</tr>
<tr>
<td>CHAPTER 4 DISCUSSION AND CONCLUSION</td>
<td></td>
</tr>
<tr>
<td>4.1 Discussion</td>
<td>29</td>
</tr>
<tr>
<td>4.2 Conclusion</td>
<td>30</td>
</tr>
<tr>
<td>4.3 Future Direction</td>
<td>31</td>
</tr>
<tr>
<td>4.3.1 Use only one public folder for SAX based code in Microsoft Exchange 2000 Server</td>
<td>31</td>
</tr>
<tr>
<td>4.3.2 Implement both SAX based code and DOM based code in the prototype system that involved bigger XML document</td>
<td>31</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>32</td>
</tr>
<tr>
<td>APPENDIX A: “IMPORT XML USING SAX” AGENT’S SCRIPT</td>
<td>33</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 2.1: W3C Properties of DOM (Source: W3Schools, 2001)........ 7
Table 2.2: W3C Methods of DOM (Source: W3Schools, 2001)........ 8
Table 3.1: DOM based code Environment Processing Time Result...... 24
Table 3.2: SAX based code Environment Processing Time Result..... 25
Table 3.3: SAX based code Domino and DOM based code Exchange
   Environment Processing Time Result............................ 26
Table 3.4: DOM based code Domino and SAX based code Exchange
   Environment Processing Time Result............................ 27
Table 3.5: Total Processing Time for Each Setup...................... 27
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Staff.xml</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>The Tree Diagram for Staff.xml</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>XML Tree Structure (Source: Brownell, 2002)</td>
<td>7</td>
</tr>
<tr>
<td>2.4</td>
<td>XML Events Structure (Source: Brownell, 2002)</td>
<td>8</td>
</tr>
<tr>
<td>2.5</td>
<td>SAX Application Diagram</td>
<td>9</td>
</tr>
<tr>
<td>2.6</td>
<td>DOM Framework (Source: Hunter et al., 2000)</td>
<td>10</td>
</tr>
<tr>
<td>2.7</td>
<td>SAX Framework (Source: Hunter et al., 2000)</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Parts that Modified for SAX Implementation</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>The Differentiation of Lotus Script in Leave Form DOM and Leave Form SAX</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>Two Addition Folders in Microsoft Exchange 2000 Server</td>
<td>16</td>
</tr>
<tr>
<td>3.4</td>
<td>Agents in “MessageAttachment” and “MessageAttachment (SAX)” Public Folders</td>
<td>17</td>
</tr>
<tr>
<td>3.5</td>
<td>Extraction of “Cuti.xml” File to “parsefile.txt” File</td>
<td>18</td>
</tr>
<tr>
<td>3.6</td>
<td>Agent in the “Agent” public Folder</td>
<td>19</td>
</tr>
<tr>
<td>3.7</td>
<td>DOM based code Environment Setup</td>
<td>20</td>
</tr>
<tr>
<td>3.8</td>
<td>SAX based code Environment Setup</td>
<td>21</td>
</tr>
<tr>
<td>3.9</td>
<td>SAX based code Domino and DOM based code Exchange Environment Setup</td>
<td>21</td>
</tr>
<tr>
<td>3.10</td>
<td>DOM based code Domino and SAX based code Exchange Environment Setup</td>
<td>22</td>
</tr>
<tr>
<td>3.11</td>
<td>Event Log Entry in Domino Workflow Environment</td>
<td>23</td>
</tr>
<tr>
<td>3.12</td>
<td>Agent Event Log in Microsoft Exchange 2000 Server</td>
<td>23</td>
</tr>
</tbody>
</table>
If not otherwise indicated, the author is the source of the figures.
LIST OF ABBREVIATIONS

ACL  Access Control List
AD  Active Directory
API  Application Programming Interface
BASIC  Beginner’s All Purpose Symbolic Instruction Code
CDO  Collaboration Data Object
CEC  Digital’s Applied Research Center
CORBA  Common Object Request Broker Agent
DLLs  Dynamic Link Libraries
DMA  Document Management Alliance
DOM  Document Object Model
DSS  Decision Support System
E-Business  Electronic Business
E-Commerce  Electronic Commerce
EAI  Enterprise Application Integration
ENC  European Networking Center
FSO  File System Object
GUI  Graphical User Interface
ID  Identification Data
IE  Information Engineering
IPSD  Interactive, Process-Oriented System Development
IPSI  Integrated Publication and Information Systems Institute
ISST  Institute for Software Engineering and Systems Engineering
IT  Information Technology
Mentor  Middleware for Enterprise-Wide Workflow Management
MIS  Management Information System
MoU  Memorandum of Understanding
OpenGL  Open Graphic Library
PDF  Portable Data Format
R5  Release 5
RDBMS  Relational Database Management Systems
RTF  Rich Text File
SAX  Simple API XML
SD  System Development
SMTP  Simple Mail Transfer Protocol
SQL  Structured Query Language
TP  Transaction Processing
URLs  Unified Resource Locators
UUM  Universiti Utara Malaysia
VBScript  Microsoft Visual Basic Scripts Edition
WFMC  Workflow Management Coalition
WFMS  Workflow Management System
XML  Extensible Markup Language
CHAPTER 1 INTRODUCTION

1.1 Background

Previous research by Juliana (2003) has shown that interoperability between Domino Workflow and Microsoft Exchange 2000 Server can be achieved using Microsoft Exchange Connector for Lotus Notes and Document Object Model (DOM) of Extensible Markup Language (XML).

As stated by Laddad (2000) and Skonnard (2000), Extensible Markup Language (XML) standard consists of two kinds of Application Programming Interfaces (APIs), i.e. Document Object Model (DOM) and Simple Application XML (SAX). DOM is more likely to be used in information exchanging and there are more examples of programming from various languages such as Java, VBScript and LotusScript on DOM (Juliana, 2003).

1.2 Problem statement

The used of DOM as stated by Hunter et al. (2000) enables programmers to create documents and parts of documents, navigate through the document, move, copy and remove parts of the document, add or modify attributes. However, these document manipulation might have a high price in terms of flexibility and resource
consumption (Brownell, 2002). The use of SAX can perhaps minimize the lengthy time spent in the prototype system developed previously using DOM because as stated by Bourret (2000) and Birbeck et al. (2001) the SAX-based code uses much less memory where it buffers only one row of data at a time while the DOM-based code buffers the entire document. Furthermore, the SAX-based code is faster because it does not have to spend time building a DOM tree.

1.3 Objective

The main objective of this study is to process the XML document in the prototype system using SAX API. Then, a performance evaluation between the same prototype systems, i.e. Leave Application System with different XML API will be carried out.

1.4 Summary of Contributions

In general, this study will provide better performance (in terms of time) of Leave Application system between two environments, i.e. Domino Workflow and Microsoft Exchange 2000 Server.
1.5 Thesis Outline

This section presents summaries of chapters as outline in this thesis.

The first chapter describes the background of the study that using DOM of XML in achieving workflow interoperability. The next section presents the need to use SAX in this study.

Chapter 2 first describes the overview of XML. The next two sections present each of the XML API i.e. DOM and SAX. The last section of this chapter provides the pro and cons between the two APIs.

Chapter 3 first discusses the research methods use in this research. The next section presents the implementation of the change in the API in the Leave Application system from DOM to SAX. The last section presents the performance evaluation process that carried out in Leave Application System that using different API of XML.

Finally, Chapter 4 discusses the findings of the study and followed by future work that can be implemented.
CHAPTER 2 LITERATURE REVIEW

2.1 Extensible Markup Language (XML)

Extensible Markup Language (XML) is a new standard that was produced by World Wide Web Consortium (W3C) in late 1998. It is a set of syntax rules and guidelines for defining text-based markup languages. XML languages have a number of uses including:

- Exchanging information
- Defining document types
- Specifying messages

XML is a markup language for documents containing structured information. Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays (for example, content in a section heading has a different meaning from content in a footnote, which means something different than content in a figure caption or content in a database table, etc.). Almost all documents have some structure.

XML contains several advantages. Some of it that listed by Connolly (2002) includes:

- Support for the integration of data from multiple sources.
- Open standard and platform/vendor independent.
- Ability to describe data from a wide variety of applications.
- Extensibility; which allows users to define their own tags to meet their own particular requirements.
- XML can provide more (and better) facilities for browser presentation and performance.

According to Sills et al. (2002), in an XML document, the data are stored in a hierarchical fashion as shown in Staff.xml in Figure 2.1.

```xml
<?xml version="1.0" ?>
<recommendations count="1">
  <STAFFLIST>
    <STAFF branchNo="B005">
      <NAME>Yazid</NAME>
      <STAFFNO>SL21</STAFFNO>
      <POSITION>Manager</POSITION>
    </STAFF>
    <STAFF branchNo="B003">
      <NAME>Juliana</NAME>
      <STAFFNO>SG37</STAFFNO>
      <POSITION>Assistant</POSITION>
    </STAFF>
  </STAFFLIST>
</recommendations>
```

Figure 2.1: Staff.xml

A hierarchy is also referred to as a tree in data structures. Conceptually, the data stored in the Staff.xml can be represented as a tree diagram as shown in Figure 2.2.
As the preceding views briefly about XML, let look at the Application Programming Interface (API) on XML. An Application Programming Interface (API) is a set of interfaces and classes used to expose particular functionality to a variety of applications (Brownell, 2002). An XML APIs generally fall into two categories, i.e. tree-based and event-based. The following section will describe the tree-based API of XML which known as Document Object Model (DOM).

2.2 Document Object Model (DOM)

The XML Document Object Model (DOM) is a programming interface for XML documents. It defines the way an XML document can be accessed and manipulated. As a W3C specification, the objective for the XML DOM has been to provide a standard programming interface to a wide variety of applications. The XML DOM is designed to be used with any programming language and any operating system. With

Figure 2.2: The Tree Diagram for Staff.xml

Each rectangle is a node in the tree. STAFF node happens to be a child node of the STAFFLIST node and parent node of its NAME, STAFFNO and POSITION nodes.
the XML DOM, a programmer can create an XML document, navigate its structure, and add, modify, or delete its elements.

A DOM implementation presents an XML document as a tree structure, or allows client code to build such a structure from scratch. It then gives access to the structure through a set of objects, which provided well-known interfaces. Figure 2.3 shows some XML text and its transformation to tree structure.

![Figure 2.3: XML Tree Structure (Source: Brownell, 2002)](image)

In order to extract the XML document to tree structure, XML parser will be used. Example of XML parser is MSXML; which is built-in parser in Internet Explorer. Once we have a DOM tree or document object, we can access the parts of our XML document through its properties and methods. Example of DOM properties is shown in Table 2.1 and example of DOM methods is shown in Table 2.2.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Returns a NamedNodeMap containing all attributes for this node</td>
</tr>
<tr>
<td>childNodes</td>
<td>Returns a NodeList containing all the child nodes for this node</td>
</tr>
<tr>
<td>FirstChild</td>
<td>Returns the first child node for this node</td>
</tr>
<tr>
<td>LastChild</td>
<td>Returns the last child node for this node</td>
</tr>
<tr>
<td>nextSibling</td>
<td>Returns the next sibling node. Two nodes are siblings if they have the same parent node</td>
</tr>
<tr>
<td>nodeName</td>
<td>Returns the nodeName, depending on the type</td>
</tr>
<tr>
<td>nodeType</td>
<td>Returns the nodeType as a number</td>
</tr>
<tr>
<td>nodeValue</td>
<td>Returns, or sets, the value of this node, depending on the type</td>
</tr>
<tr>
<td>ownerDocument</td>
<td>Returns the root node of the document</td>
</tr>
<tr>
<td>parentNode</td>
<td>Returns the parent node for this node</td>
</tr>
</tbody>
</table>
Table 2.2: W3C Methods of DOM (Source: W3Schools, 2001)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appendChild(newChild)</td>
<td>Appends the node newChild at the end of the child nodes for this node</td>
</tr>
<tr>
<td>cloneNode(boolean)</td>
<td>Returns an exact clone of this node. If the boolean value is set to true, the cloned node contains all the child nodes as well</td>
</tr>
<tr>
<td>hasChildNodes()</td>
<td>Returns true if this node has any child nodes</td>
</tr>
<tr>
<td>insertBefore(newNode,refNode)</td>
<td>Inserts a new node, newNode, before the existing node, refNode</td>
</tr>
<tr>
<td>removeChild(nodeName)</td>
<td>Removes the specified node, nodeName</td>
</tr>
<tr>
<td>replaceChild(newNode,oldNode)</td>
<td>Replaces the oldNode, with the newNode</td>
</tr>
</tbody>
</table>

The following section will describe the event-based API of XML which known as Simple API of XML (SAX).

2.3 Simple API of XML (SAX)

SAX is a product of collaboration on the XML-DEV mailing list. According Birbeck et al. (2001), SAX uses a parser that reads from the input XML document and notifies the application of interesting events as shown in Figure 2.4. A SAX parser uses some predefined callbacks to notify an application of parsing events. These callbacks are defined as methods of several standard SAX interfaces to be implemented by the application.

![Figure 2.4: XML Events Structure (Source: Brownell, 2002)](image-url)
The components of a simple SAX application are shown in the Figure 2.5.

In the diagram:

- The Application is the "main program": the code that we write to start the whole process.
- The Document Handler is code that we write to process the contents of the document.
- The Parser is an XML Parser that conforms to the SAX standard.

The job of the application is to create a parser; to create a document handler; to tell the parser what document handler to use (by calling the parser's setDocumentHandler() method); and to tell the parser to start processing a particular input document (by calling the parse() method of the parser). The job of the parser is to notify the document handler of all the interesting things it finds in the document, such as element start tags and end tags. The job of the document handler is to process these notifications to achieve whatever the application requires.
As the preceding two sections describe each of XML API, the next section will look at the pro and cons in using both of them.

2.4 DOM versus SAX

In reviewing both of DOM and SAX, diagrammatically, DOM as shown in Figure 2.6 will parse the document into the DOM tree and then use the DOM to navigate around the document.

![DOM Framework](source)

Figure 2.6: DOM Framework (Source: Hunter et al., 2000)

Whereas the SAX approach as shown in Figure 2.7 tell the parser to raise events whenever it find something (data).

![SAX Framework](source)

Figure 2.7: SAX Framework (Source: Hunter et al., 2000)
If our XML document were 20 mega bytes large, it would be very inefficient to construct and traverse an in-memory parse tree just to locate one piece of contextual information. In the other hand, an event-based interface; which is SAX would allow us to find it in a single pass using very little memory. The using of DOM will not only take up space (memory) but also time in building the tree. However, as SAX only allows a view of one bit of the document at a time, it is extremely useful using DOM for random-access applications.

As objectives of this research is to minimize the time consumed in the interoperability between Domino Workflow and Microsoft Exchange 2000 Server in Leave Application prototype, the advantages of SAX is most probably highlighted. According to Hunter et al. (2000), Birbeck et al. (2001), Brownell (2002), and Sills et al. (2002), SAX parsing is faster than DOM parsing. From this point of view the next chapter will describe the code changing process from DOM code-based to SAX code-based in the Leave Application prototype system to realize the evaluation of performance in both environment.
CHAPTER 3 IMPLEMENTATION AND EVALUATION

The System Development (SD) approach by Nunamaker et al. (1991) has been adopted in this research. The stages in the SD approach are as follows:

- Construct a conceptual framework
- Develop a system architecture
- Analyze and design the system
- Build the prototype system
- Observe and evaluate the system

However as this research will only change the DOM based code to SAX based code, not all stages in SD will be applied. The stages that will be focusing are:

- Analyze and design the system
- Observe and evaluate the system

3.1 Analyze and Design the System

The modification of the system will focuses only on several part of overall development by Juliana (2003), i.e. “Create Forms and Views” on Domino Workflow environment,
“Create the Public Folders” part and ‘Create Agents” part on the Microsoft Exchange 2000 Server environment as shown in Figure 3.1.

Figure 3.1: Parts that Modified for SAX Implementation

3.1.1 Domino Workflow Environment

There were two main modifications in the “Creates Forms and Views” part. The first one is on the Leave Form. The original Leave Form will be saved in two new form named Leave Form DOM and Leaved Form SAX. Both Leave Form DOM and Leave Form SAX will be change in terms of their name to Leave Form whenever a testing is carried out. The Leave Form DOM is actually the original form, so there
was no modification on it. The Leave Form SAX was modified in terms of its scripting in Queryclose event. Figure 3.2 shows the part of LotusScript in the Queryclose event for both Leave Form DOM and Leave Form SAX. Minor change is only on the word “Message Attachment”. It will change to “MessageAttachment1” in Leave Form SAX to make sure that different public folder in Microsoft Exchange 2000 server that contains SAX-based code will parse the document that sent to that folder.

Figure 3.2: The Differentiation of Lotus Script in Leave Form DOM and Leave Form SAX

The second modification is on the agent side. The agent that supports the XML parsing on DOM is “Import XML using DOM” agent. To facilitate the used of SAX, new agent was created, known as “Import XML using SAX”. This agent contains one event handler that consists of several lines of script:
Public EventHandler(Session s, Database db) throws SAXException, IOException, Exception{
    com.ibm.xml.parsers.SAXParser parser = new com.ibm.xml.parsers.SAXParser();
    //Register the event handler and the error handler
    parser.setDocumentHandler(this);
    parser.setErrorHandler((ErrorHandler)this);
    //Parse the document to create the events.
    parser.parse("file:///C:/Cuti1.xml");
}

This event handler will call up other event such as:

- startDocument()
- endDocument()
- startElement()
- characters()
- endElement()

The overall script of this agent can be found in Appendix A.

Now, we will look at the modification on the Microsoft Exchange 2000 Server environment that took place on “Create the Public Folders” part and “Create Agents” part.
3.1.2 Microsoft Exchange 2000 Server Environment

In this environment, the first modification is on the “Create the Public Folders” part. In the previous prototype system, three public folders were created. In this research, two more public folders, i.e. “Agent” and “MessageAttachment (SAX)” were created to support the SAX-based code as shown in Figure 3.3. The “MessageAttachment (SAX)” public folder is known as “MessageAttachment1” in Domino Workflow environment.

![Figure 3.3: Two Addition Folders in Microsoft Exchange 2000 Server](image)

The second modification is on the “Create Agents” part. As the DOM based-code environment uses an agent called “parsing xml document” that resided on the “MessageAttachment” public folder, the SAX based-code agent that called “parsing
xml file” was created. This agent was resided on the “MessageAttachment (SAX)” public folder. These two agents are shown in Figure 3.4 within their respective public folder.

The “parsing xml file” agent (can be found in Appendix B) that triggered when a new item is posted in the “MessageAttachment (SAX)” public folder contains several tasks that justified the functions of the “MessageAttachment (SAX)” public folder. The tasks are as follows:
- Receives an e-mail that contains “Cuti.xml” file from Domino Workflow environment

- Save the “Cuti.xml” file to C directory

- Run “parse.exe” file (a Visual Basic Application that is a main SAX code that can be found in Appendix C) to extract “Cuti.xml” file to “parsefile.txt” and save the “parsefile.txt” to C directory (Example of “Cuti.xml” file together with its extraction that is “parsefile.txt” is shown in Figure 3.5).

- Create new message in the “Agent” public folder (as a way of activating another agent)

![Cuti.xml File]

```xml
<?xml version='1.0' ?>
<recommendations count='1'>
  <document form='Leave Form'>
    <nok>
      <text>000</text>
    </nok>
    <napemohon>
      <text>admin</text>
    </napemohon>
    <lawatan>
      <text>8A</text>
    </lawatan>
    <jab>
      <text>STM</text>
    </jab>
    <bakicuti>
      <number>28</number>
    </bakicuti>
    <StDate>
      <datetime>11/10/2004</datetime>
    </StDate>
    <EDate>
      <datetime>11/10/2004</datetime>
    </EDate>
  </document>
</recommendations>
```

![Parsefile.txt]

Figure 3.5: Extraction of “Cuti.xml” File to “parsefile.txt’ File
As mentioned earlier, two public folders were created. As the preceding describes agent that involved in the “MessageAttachment (SAX)” public folder, now we will look at the agent that involved in the “Agent” public folder. Figure 3.6 shows the agent that called “create form” that resided in the “Agent” public folder.

Figure 3.6: Agent in the “Agent” public Folder

The “create form” agent that triggered when a new item is posted in the “Agent” public folder contains task that justified the function of the “Agent” public folder. The task is to open the “parsefile.txt” file and assign its data to ‘form approval’ form in the “Leave Application” public folder. The whole script of this agent can be found in Appendix D.
3.2 Observe and Evaluate the System

This section will describe the performance measure of both DOM and SAX in terms of their responsiveness, which evaluate how quickly a given task of processing time i.e. XML document parsing and leave form creating, can be accomplished by both APIs. The measure was carried out in two category of measurement i.e. within each APIs and combination of APIs.

3.2.1 Within each APIs

This measurement was carried out either using DOM based code or SAX based code in both Domino Workflow and Microsoft Exchange 2000 Server environment of the Leave Application System as shown in Figure 3.7 and Figure 3.8.

![Figure 3.7: DOM based code Environment Setup](image-url)
3.2.2 Combination of APIs

This measurement was carried out using combination of DOM based code and SAX based code in both Domino Workflow and Microsoft Exchange 2000 Server environment of the Leave Application System as shown in Figure 3.9 and Figure 3.10.
Both Domino Workflow and Microsoft Exchange 2000 Server user were set up to be an applicant and approver of the Leave Application System. As stated by Joines et al. (2003) same test must be run at least three times, the leave application process was carried out ten times for each setup mentioned above. For each leave application process that also involved the approval part, the processing time was obtained from time start and time end in event log file in both Domino Workflow and Microsoft Exchange 2000 Server environment. Figure 3.11 shows the example of event log in Domino Workflow environment. While Figure 3.12 shows the example of agent event log in public folders in Microsoft Exchange 2000 Server.
Figure 3.11: Event Log Entry in Domino Workflow Environment

Figure 3.12: Agent Event Log in Microsoft Exchange 2000 Server Environment
According to Joines et al. (2003) the performance measure demands repeatable and reliable results. From this point of view this research will consider only repeatable results of processing time. If the times that captured are repeatable in more then one specific times then the average of the times will be calculated.

The processing time for leave application process with the first setup i.e. DOM based code environment is shown in Table 3.1.

Table 3.1: DOM based code Environment Processing Time Result

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Applicant Environment</th>
<th>Approver Environment</th>
<th>DO MINO</th>
<th>MICROSOF T EXCHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Agent &quot;import XML using DOM&quot; Processing Time (Second)</td>
<td>Total (Second)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Exchange</td>
<td>Domino</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Domino</td>
<td>Exchange</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Exchange</td>
<td>Domino</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Domino</td>
<td>Exchange</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Exchange</td>
<td>Domino</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>Domino</td>
<td>Exchange</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>Exchange</td>
<td>Domino</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Domino</td>
<td>Exchange</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>Exchange</td>
<td>Domino</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>Domino</td>
<td>Exchange</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

From Table 3.1 above, the repeatable processing time for “import XML using DOM” agent is 6 second. The repeatable processing time for “parsing xml document” agent is 3 and 4 second. So the average processing time for “parsing xml document” agent is 3.5 second. The total processing time for DOM based code environment is 9.5 second.
The processing time for leave application process with the second setup i.e. SAX based code environment is shown in Table 3.2.

**Table 3.2: SAX based code Environment Processing Time Result**

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Applicant Environment</th>
<th>Approver Environment</th>
<th>DOMINO Agent &quot;import XML using SAX&quot; Processing Time (Second)</th>
<th>MICROSOFT EXCHANGE Agent &quot;parsing xml file&quot; Processing Time (Second)</th>
<th>Agent &quot;create form&quot; Processing Time (Second)</th>
<th>Total (Second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exchange</td>
<td>Domino</td>
<td>16</td>
<td>2</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>Domino</td>
<td>Exchange</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Exchange</td>
<td>Domino</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Domino</td>
<td>Exchange</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Exchange</td>
<td>Domino</td>
<td>12</td>
<td>5</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Domino</td>
<td>Exchange</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>Exchange</td>
<td>Domino</td>
<td>9</td>
<td>2</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>Domino</td>
<td>Exchange</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>Exchange</td>
<td>Domino</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>Domino</td>
<td>Exchange</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>14</td>
</tr>
</tbody>
</table>

From Table 3.2 above, the repeatable processing time for “import XML using SAX” agent is 5 second. The repeatable processing time for “parsing xml file” agent is 2 second and the repeatable processing time for “create form” agent is 3 second. The total processing time for SAX based code environment is 10 second.

The processing time for leave application process with the third setup i.e. SAX based code Domino and DOM based code Microsoft Exchange 2000 Server environment is shown in Table 3.3.
Table 3.3: SAX based code Domino and DOM based code Exchange Environment Processing Time Result

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Applicant Environment</th>
<th>Approver Environment</th>
<th>DOMINO Agent “importXML using SAX” Processing Time (Second)</th>
<th>MICROSOFT EXCHANGE Agent “parsing xml document” Processing Time (Second)</th>
<th>Total (Second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exchange</td>
<td>Domino</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Domino</td>
<td>Exchange</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Exchange</td>
<td>Domino</td>
<td>9</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>Domino</td>
<td>Exchange</td>
<td>5</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Exchange</td>
<td>Domino</td>
<td>6</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>Domino</td>
<td>Exchange</td>
<td>6</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>Exchange</td>
<td>Domino</td>
<td>8</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>Domino</td>
<td>Exchange</td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>Exchange</td>
<td>Domino</td>
<td>6</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>Domino</td>
<td>Exchange</td>
<td>8</td>
<td>4</td>
<td>12</td>
</tr>
</tbody>
</table>

From Table 3.3 above, the repeatable processing time for “import XML using SAX” agent is 5 and 6 second. So the average processing time for “import XML using SAX” agent is 5.5 second. The repeatable processing time for “parsing xml document” agent is 3 second. The total processing time for SAX based code Domino and DOM based code Microsoft Exchange 2000 Server environment is 8.5 second.

The processing time for leave application process with the forth setup i.e. DOM based code Domino and SAX based code Microsoft Exchange 2000 Server environment is shown in Table 3.4.
From Table 3.4 above, the repeatable processing time for “import XML using DOM” agent is 5 and 8 second. So the average processing time for “import XML using DOM” agent is 6.5 second. The repeatable processing time for “parsing xml file” agent is 2 second and the repeatable processing time for “create form” agent is 3 second. The total processing time for DOM based code Domino and SAX based code Microsoft Exchange 2000 Server environment is 11.5 second.

From the above results, Table 3.5 summarizes the total processing time for four different setups mentioned above.

Table 3.5: Total Processing Time for Each Setup

<table>
<thead>
<tr>
<th>Setup</th>
<th>Total Processing Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOM based code Environment</td>
<td>9.5</td>
</tr>
<tr>
<td>SAX based code Environment</td>
<td>10</td>
</tr>
<tr>
<td>SAX based code Domino and DOM based code Exchange Environment</td>
<td>8.5</td>
</tr>
<tr>
<td>DOM based code Domino and SAX based code Exchange Environment</td>
<td>11.5</td>
</tr>
</tbody>
</table>
As for now this chapter has describes the implementation of SAX in the Leave Application prototype and evaluation process of both APIs. The result obtained in the evaluation process will be discussed in next chapter.
CHAPTER 4 DISCUSSION AND CONCLUSION

This chapter will discuss the result obtained. From that discussion, a conclusion will be stated and suggestion of future direction work will be presented.

4.1 Discussion

Result shown in Table 3.5 show that combination of SAX based code in Domino Workflow environment and DOM based code in Microsoft Exchange 2000 Server produces the lowest processing time of Leave Application process.

The statement that stated in Chapter 2 which is “SAX parsing is faster than DOM parsing”, can be proven by observing the processing time in Domino Workflow environment. Within APIs implementation that referring to Table 3.1 and Table 3.2, processing time for SAX based code is 5 second while processing time for DOM based code is 6 second. By referring Table 3.3 and 3.4 for combination of APIs implementation, processing time for SAX based code is 5.5 second while processing time for DOM based code is 6.5 second.
However in the Microsoft Exchange 2000 Server environment, DOM based code produces the lowest processing time then the SAX based code. This happens because the implementation of SAX based code in the Microsoft Exchange 2000 Server uses two public folders that drag the processing times.

### 4.2 Conclusion

Overall, the objectives of this research as follows:

1. to process the XML document in the prototype system developed using SAX API
2. to carry out performance evaluation for the prototype using different XML API has been achieved. Both objectives were achieved in the process of implementation and evaluation in Chapter 3.

As stated in Chapter 1 that the contribution of this research is to provide better performance, which is low in response time, the use of combination of SAX based code Domino and DOM based code Microsoft Exchange 2000 Server that has the lowest processing time will fulfill the contribution for this research.
4.3 Future Direction

There are number of aspects that future work can be done from this research:

4.3.1 Use only one public folder for SAX based code in Microsoft Exchange 2000 Server

The processing time of SAX based code can possibly be minimizes by planning the implementation of SAX based code in one public folder. If this can be achieved, the totally SAX based code environment will produces the lowest processing time then DOM based code environment.

4.3.2 Implement Both SAX based code and DOM based code in the prototype system that involved bigger XML document

The size of XML document that used in this research is only 1 kilo bytes. Perhaps a bigger XML document can show more relevant of different processing time between SAX based code and DOM based code.
REFERENCES


import lotus.domino.*;
import org.xml.sax.*;
import java.io.*;
import java.util.*;
public class JavaAgent extends AgentBase{

Session s;
Database db;

public void NotesMain(){
    try {
        EmbeddedObject attachment;

        s = getSession();
        AgentContext agentContext = s.getAgentContext();
        db = agentContext.getCurrentDatabase();

        // we will be working with a document available from the All
        // by Form and Subject view - open it up
        View v = db.getView( "Message from Exchange" );
        db.updateFTIndex(true);
        if (v == null) {
            System.out.println( "Unable to open view" );
            return;
        }

        lotus.domino.Document mailDoc = v.getFirstDocument();
        //while (mailDoc != null) {
        //lotus.domino.Document mailDoc = v.getLastDocument();
        // Get ahold of the Body field in the mail document
        RichTextItem body = (RichTextItem)mailDoc.getFirstItem("Body");
        System.out.println( "jumpa attachment" );
        Vector attachments = body.getEmbeddedObjects();
        if (attachments.size() > 0) {
            attachment = (EmbeddedObject)body.getEmbeddedObjects().elementAt(0);
            attachment.extractFile("c:\" + attachment.getSource());
        } else {
            System.out.println( "No attachments found in the
document with key 'Memo'\devcon2k.xml attached" );
            return;
        }

        // parse the XML contained in the attached file
        //Instantiate an event handler
    }
}

APPENDIX A: “IMPORT XML USING SAX” AGENT’S SCRIPT
DocumentHandler handler = new EventHandler(s, db);

if (v.getNextDocument(mailDoc) == null){
    mailDoc.remove(true);
} else {
    mailDoc = v.getNextDocument(mailDoc);
    lotus.domino.Document delmailDoc = v.getPrevDocument(mailDoc);
    delmailDoc.remove(true);
}

} catch(Exception e) {
    e.printStackTrace();
}

} //end main
} //end class JavaAgent

//Methods of this class are listeners for document events
// and error events. Note that HandlerBase implements
// the ErrorHandler interface.

class EventHandler extends HandlerBase{
    private String NoPekerja=null;
    private String NamaPemohon=null;
    private String Jawatan=null;
    private String Jabatan=null;
    private String bakicuti_1=null;
    private String MulaBercuti=null;
    private String TamatCuti=null;
    private String Hari=null;
    private String AlamatBercuti=null;
    private String Telefon=null;
    private String TarikhMohon=null;
    private String pelulus=null;
    private String kelulusan=null;
    private String reason=null;
    private String accumulator;

    public EventHandler(Session s, Database db) throws SAXException, IOException, Exception{

        com.ibm.xml.parsers.SAXParser parser = new com.ibm.xml.parsers.SAXParser();
        //Register the event handler and the error handler
        parser.setDocumentHandler(this);
        parser.setErrorHandler((ErrorHandler)this);
        //Parse the document to create the events.
        parser.parse("file:///C:/Cuti1.xml");
    }
View v1 = db.getView("Leave form by all field");

    if (v1 == null) {
        System.out.println("Unable to open view");
        return;
    }

// create a Vector to act as a key for the lookup, and add two elements
Vector key = new Vector();
    key.addElement("Leave Form");
    key.addElement(this.getNoPekerja());
    key.addElement(this.getNamaPemohon());
    key.addElement(this.getJawatan());
    key.addElement(this.getJabatan());
    key.addElement(new java.lang.Double(this.getbakicuti_1()));
    key.addElement(this.getMulaBercuti());
    key.addElement(this.getTamatCuti());
    key.addElement(this.getHari());
    key.addElement(this.getAlamatBercuti());
    key.addElement(this.getTelefon());
    key.addElement(this.getTarikhMohon());
    System.out.println("Processing in progress...");
    lotus.domino.Document doc = v1.getDocumentByKey(key);

    if (doc == null) {
        System.out.println("document belum ada");
        lotus.domino.Document newDoc = db.createDocument();
        newDoc.replaceItemValue("Form", "Leave Form");

        newDoc.replaceItemValue("NoPekerja", this.getNoPekerja());
        newDoc.replaceItemValue("NamaPemohon", this.getNamaPemohon());
        newDoc.replaceItemValue("namatext", this.getNamaPemohon());
        newDoc.replaceItemValue("Jawatan", this.getJawatan());
        newDoc.replaceItemValue("Jabatan", this.getJabatan());
        newDoc.replaceItemValue("Bakicuti_1", new java.lang.Double(this.getbakicuti_1()));
        newDoc.replaceItemValue("bakicutitext", this.getbakicuti_1());
        newDoc.replaceItemValue("MulaBercuti", s.createDateTime(this.getMulaBercuti()));
        newDoc.replaceItemValue("mulabercutitext", this.getMulaBercuti());
        newDoc.replaceItemValue("TamatCuti", s.createDateTime(this.getTamatCuti()));
        newDoc.replaceItemValue("tamatcutitext", this.getTamatCuti());
        newDoc.replaceItemValue("Hari", new java.lang.Double(this.getHari()));
        newDoc.replaceItemValue("haritext", this.getHari());
        newDoc.replaceItemValue("AlamatBercuti", this.getAlamatBercuti());
        newDoc.replaceItemValue("Telefon", this.getTelefon());
newDoc.replaceItemValue("TarikhMohon", s.createDateTime(this.getTarikhMohon()));
newDoc.replaceItemValue("tarikhmohontext", this.getTarikhMohon());
newDoc.replaceItemValue("pelulus", this.getpelulus());
newDoc.replaceItemValue("NewProcessNameOS", "Leave");
newDoc.replaceItemValue("NewJobNameOS", this.getNamaPemohon());
newDoc.replaceItemValue("NewJobPriorityOS", "2. Medium");
newDoc.replaceItemValue("MailStatusOS", "2");
newDoc.replaceItemValue("InitiateOS", "yes");

//newDoc.replaceItemValue( "ExternalInitiator", "" );

newDoc.save();

System.out.println( "run agent backgrounder" );
    Agent agent1 = db.getAgent("(OS Domino Workflow Backgrounder)" );
    agent1.run();
    System.out.println( "run agent claim" );
    Agent agent3 = db.getAgent("(claim)" );
    agent3.run();
    System.out.println( "run agent reroute" );
    Agent agent2 = db.getAgent("(reroute)" );
    agent2.run();
    System.out.println( "run agent claim lagi" );
    Agent agent4 = db.getAgent("(claim)" );
    agent4.run();

    //mail withdoclink to the approver
    System.out.println( "run agent ke user" );
    Agent agent = db.getAgent("create mail with doclink to be sent to notes user");
    agent.run();

} else{
    //mailDoc.remove(true);
    System.out.println( "document dah ada" );
    doc.replaceItemValue("kelulusan", this.getkelulusan() );
    doc.replaceItemValue("reason", this.getreason() );

    doc.save();

    if (xmlDoc.getDocumentElement().getElementsByTagName("kelulusan").item(0).getFirstChild().getFirstChild().getNodeValue()==No){

// doc.replaceItemValue( "reason", 
xmlDoc.getDocumentElement().getElementsByTagName("reason").item(0).getFirstChild().getFirstChild().getNodeValue() );

//System.out.println(xmlDoc.getDocumentElement().getElementsByTagName("kelulusan").item(0).getFirstChild().getFirstChild().getNodeValue());
//System.out.println(xmlDoc.getDocumentElement().getElementsByTagName("reason").item(0).getFirstChild().getFirstChild().getNodeValue());

//
//doc.save();
//
//}
System.out.println( "run agent path completed" );
Agent agent3 = db.getAgent("(routing path completed)");
agent3.run();
doc.save();
//mail withdoclink to the applicant
System.out.println( "run agent ke applicant" );
Agent agent = db.getAgent("create mail with doclink to be sent to notes user applicant");
agent.run();

System.out.println( "Processing complete..." );

}

public void startDocument(){//handle startDocument event
System.out.println("Start Document");
}//end startDocument()

public void endDocument(){//handle endDocument event
System.out.println("End Document");
}//end endDocument()

//handle startElement event displaying attributes
public void startElement(String name,AttributeList atts){
System.out.println("Start element: " + name);
if (atts != null) {
    int len = atts.getLength();
    //process all attributes
    for (int i = 0; i < len; i++) {
        String attName = atts.getName(i);
        String type = atts.getType(i);
        String value = atts.getValue(i);
        System.out.println("Attribute: " + attName

37
+ " . Value = " + value + ", Type = " + type);
} // end for loop on attributes
} // end if
} // end start element

// handle characters event
public void characters(char[] ch, int start, int length) {
    // accumulator.append(ch, start, length);
    accumulator = new String(ch, start, length);
    System.out.println(new String(ch, start, length));
} // end characters()

// handle endElement event
public void endElement(String name) {
    if (name.equals("NoPekerja")) {
        NoPekerja = accumulator.toString().trim();
    } // end if
    if (name.equals("NamaPemohon")) {
        NamaPemohon = accumulator.toString().trim();
    } // end if
    if (name.equals("Jawatan")) {
        Jawatan = accumulator.toString().trim();
    } // end if
    if (name.equals("Jabatan")) {
        Jabatan = accumulator.toString().trim();
    } // end if
    if (name.equals("bakicuti_1")) {
        bakicuti_1 = accumulator.toString().trim();
    } // end if
    if (name.equals("MulaBercuti")) {
        MulaBercuti = accumulator.toString().trim();
    } // end if
    if (name.equals("TamatCuti")) {
        TamatCuti = accumulator.toString().trim();
    } // end if
    if (name.equals("Hari")) {
        Hari = accumulator.toString().trim();
    } // end if
    if (name.equals("AlamatBercuti")) {
        AlamatBercuti = accumulator.toString().trim();
    } // end if
    if (name.equals("Telefon")) {
        Telefon = accumulator.toString().trim();
    } // end if
    if (name.equals("TarikhMohon")) {
        TarikhMohon = accumulator.toString().trim();
    } // end if
    if (name.equals("pelulus")) {
        pelulus = accumulator.toString().trim();
    } // end if
    if (name.equals("kelulusan")) {
        kelulusan = accumulator.toString().trim();
    } // end if
    if (name.equals("reason")) {
        reason = accumulator.toString().trim();
    } // end if
    System.out.println("End element: " + name);
} // end endElement()

public String getNoPekerja() {
    return NoPekerja;
}
APPENDIX B: “PARSING XML FILE” AGENT’S SCRIPT

<SCRIPT RunAt=Server Language=VBScript>

,'
',-----------------------------------------------------------------------------------
Option Explicit
',-----------------------------------------------------------------------------------
', Global Variables
',-----------------------------------------------------------------------------------
Dim g_bstrDebug ' Debug String
',-----------------------------------------------------------------------------------
', CONSTANTS
',-----------------------------------------------------------------------------------
Dim g_Const_MBX
Dim g_Const_Directory
', Enter the display name of the mailbox which you want have send the report to
  g_Const_MBX = "Administrator"
', Enter the directory where you want to save all attachments
  g_Const_Directory = "c:\"
',-----------------------------------------------------------------------------------
', EVENT HANDLERS
',-----------------------------------------------------------------------------------
', DESCRIPTION: This event is fired when a new message is added to the folder
Public Sub Folder_OnMessageCreated
  Dim objSession ' Session
  Dim objInfoStores
  Dim objInfoStore
  Dim objFields
  Dim objRootFolder
  Dim objMsg ' new message
  Dim objMsgprofile ' new message
  Dim objMsgNew ' new message
  Dim objMsgs ' collections of new message
  Dim objMsgsprofile ' collections of new message
  Dim objCurrentMsg ' Current message
  Dim objCurrentMsgs ' collection of message
  Dim objFolders
</SCRIPT>
' Initialize objects
Set objSession = Nothing
Set objInfoStores = Nothing
Set objInfoStore = Nothing
Set objFields = Nothing
Set objRootFolder = Nothing
Set objMsg = Nothing
Set objMsgNew = Nothing
Set objMsgprofile = Nothing
Set objMsgs = Nothing
Set objMsgsprofile = Nothing
Set objCurrentMsg = Nothing
Set objFolders = Nothing
Set objFolder = Nothing
Set objFolderCur = Nothing
Set objFolderprofile = Nothing
Set objAttachment = Nothing
Set objAttachments = Nothing
Set objxmlDoc = Nothing
Set strEntryID = Nothing
Set subj = Nothing

' Clear error buffer
Err.Clear

' Get session informationen
Set objSession = EventDetails.Session

' No errors detected ?
If Err.Number = 0 Then

' Write some logging
Call DebugAppend(objSession.CurrentUser & " SendAtt - Processing startet", False)

' Get current folder
Set objFolderCur =
objSession.GetFolder(EventDetails.FolderID, Null)

Set objCurrentMsgs = objFolderCur.Messages

int countmsg = 0
For Each objCurrentMsg In objCurrentMsgs
    countmsg = countmsg + 1
    Call DebugAppend("Msg " & countmsg, False)
    ' Set current message to read
    objCurrentMsg.Unread = False
    objCurrentMsg.Update
    ' Remember subject of arrived message
    Call DebugAppend("New message with subject: <" &
    objCurrentMsg.Subject & "> arrived", False)

    ' Get attachment of the message
    On Error Resume Next
    Set objAttachments = objCurrentMsg.Attachments
    ' No errors detected?
    ' If Not objAttachments Is Nothing Then
        ' Attachments found, write logging
        Call DebugAppend("Attachments found", False)
    False)
    ' Extract all attachments
    For Each objAttachment In objAttachments
        ' Check if attachment is a file or link
        ' Note that CDO 1.2x does not support
        to save OLE objects
        ' and embedded messages to the file
        system
        If (objAttachment.Type = 2) Then
            ' Save attachment to the
            ' filesystem, write logging
            Call DebugAppend("Save attachments to filesystem", False)
            ' Write attachment to the
            filesystem
            objAttachment.WriteToFile(g_Const_Directory & objAttachment.Name)
            End If
    Exit For
Next

' Delete attachments
' On Error Resume Next
' objAttachments.Delete
' End If

Dim wShell
Set wShell = CreateObject("Shell.Application")
Call DebugAppend(Now, False)
wShell.Open "C:\parse.exe"
Call DebugAppend(Now, False)
objCurrentMsg.Update
objCurrentMsg.delete

Call DebugAppend(Now, False)
' Write some logging, without the folder name
Call DebugAppend("SaveAtt - Processing finished", False)

Set objInfoStores = objSession.InfoStores
For Each objInfoStore in objInfoStores
  If objInfoStore.Name = "Public Folders" Then
    Exit For
  End If
Next

Call DebugAppend("lepas", False)
' Write some logging
Call DebugAppend(objSession.CurrentUser & " SendAtt - Processing startet", False)

' Get Agent folder
Set objFields = objInfoStore.Fields
strEntryID = objFields.Item(&H66310102).Value
Set objRootFolder = objSession.GetFolder( strEntryID, objInfoStore.ID)
Set objFolders = objRootFolder.Folders
Set objFolder = objFolders.GetFirst()
Do While objFolder.Name <> "Agent"
  Set objFolder = objFolders.GetNext()
Loop

Set objMsgs = objFolder.Messages
Set objMsgNew = objMsgs.Add
objMsgNew.unread = False
objMsgNew.Update
Next
Else
    ' Check for any possible sys errors
    Call DebugAppend("Undefined Error detected", True)
End If
Call DebugAppend(Now, False)
    ' Clear objects
    Set objSession = Nothing
    Set objInfoStores = Nothing
    Set objInfoStore = Nothing
    Set objFields = Nothing
    Set objRootFolder = Nothing
    Set objMsg = Nothing
    Set objCurrentMsg = Nothing
    Set objMsgNew = Nothing
    Set objFolders = Nothing
    Set objFolder = Nothing
    Set objFolderCur = Nothing
    Set objAttachment = Nothing
    Set objAttachments = Nothing
    Set objxmlDoc = Nothing
    Set strEntryID = Nothing
    Set objMsgProfile = Nothing
    Set objMsgs = Nothing
    Set objMsgsProfile = Nothing
    Set objFolderProfile = Nothing
    Set wShell = Nothing

    ' Write results to the Scripting Agent log
    Script.Response = g_bstrDebug
End Sub

' DESCRIPTION: This event is fired when the timer on the folder expires
Public Sub Folder_OnTimer
    ' Not used
End Sub

' DESCRIPTION: This event is fired when a message in the folder is changed
Public Sub Message_OnChange
    ' Not used
End Sub

' DESCRIPTION: This event is fired when a message is deleted from the folder
Public Sub Folder_OnMessageDeleted
End Sub
Private Sub DebugAppend(bstrParm, boolErrChkFlag)
    If boolErrChkFlag = True Then
        If Err.Number <> 0 Then
            g_bstrDebug = g_bstrDebug & bstrParm & " - " & cstr(Err.Number) & " " & Err.Description & vbCrLf
            Err.Clear
        End If
    Else
        g_bstrDebug = g_bstrDebug & bstrParm & vbCrLf
    End If
End Sub

</SCRIPT>
APPENDIX C: “PARSE.EXE” SCRIPT

Sub Main()

Dim reader As New SAXXMLReader ' Reads the XML document
    Dim contentHandler As New ContentHandlerImpl ' Receives parsing events
    ruang = ""
    Set reader.contentHandler = contentHandler ' They work together
    reader.parseURL ("C:\Cuti.xml") ' Parse the document

    Exit Sub

End Sub

Option Explicit
Implements IVBSAXContentHandler

Private Sub IVBSAXContentHandler_characters(strChars As String)
    Dim fs As Object
    Dim a As Object
    strChars = Replace(strChars, vbCrLf, vbCrLf)
    Form1.Text1.Text = Form1.Text1.Text & strChars & vbCrLf
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set a = fs.CreateTextFile("c:\parsefile.txt", True)

    a.WriteLine (Trim(Replace(Form1.Text1.Text, "Text1", "")))
End Sub

Private Property Set IVBSAXContentHandler_documentLocator(ByVal RHS As MSXML2.IVBSAXLocator)
End Property

Private Sub IVBSAXContentHandler_endDocument()
End Sub

Private Sub IVBSAXContentHandler_endElement(strNamespaceURI As String, strLocalName As String, strQName As String)
End Sub

Private Sub IVBSAXContentHandler_endPrefixMapping(strPrefix As String)
End Sub

Private Sub IVBSAXContentHandler_ignorableWhitespace(strChars As String)
End Sub
Private Sub IVBSAXContentHandler_processingInstruction(strTarget As String, strData As String)
End Sub

Private Sub IVBSAXContentHandler_skippedEntity(strName As String)
End Sub

Private Sub IVBSAXContentHandler_startDocument()
End Sub

Private Sub IVBSAXContentHandler_startElement(strNamespaceURI As String, strLocalName As String, strQName As String, ByVal oAttributes As MSXML2.IVBSAXAttributes)
    Dim i As Integer
    Dim char As String
    For i = 0 To (oAttributes.length - 1)
        char = char & " " & _
            oAttributes.getLocalName(i) & "=" & oAttributes.getValue(i) _
            & ""
    Next

    If strLocalName = "qu" Then
        Err.Raise vbObjectError + 1, "ContentHandler.startElement", _
            "Found element <qu>"
    End If
End Sub

Private Sub IVBSAXContentHandler_startPrefixMapping(strPrefix As String, strURI As String)
End Sub
APPENDIX D: “CREATE FORM” AGENT’S SCRIPT

<SCRIPT RunAt=Server Language=VBScript>

',
',
', '-----------------------------------------------------------------------------------
Option Explicit
',
', '-----------------------------------------------------------------------------------
', ' Global Variables
', '-----------------------------------------------------------------------------------
Dim g_bstrDebug ' Debug String
',
', '-----------------------------------------------------------------------------------
', ' CONSTANTS
', '-----------------------------------------------------------------------------------
Dim g_Const_MBX
Dim g_Const_Directory
',
', ' Enter the display name of the mailbox which you want have send the report to
g_Const_MBX = "Administrator"
',
', ' Enter the directory where you want to save all attachments
g_Const_Directory = "c:\xml"
',
', '-----------------------------------------------------------------------------------
', ' EVENT HANDLERS
', '-----------------------------------------------------------------------------------
',
', ' DESCRIPTION: This event is fired when a new message is added to the folder
Public Sub Folder_OnMessageCreated
    Dim objSession ' Session
    Dim objInfoStores
    Dim objInfoStore
    Dim objFields
    Dim objRootFolder
    Dim objMsg ' new message
    Dim objMsgprofile ' new message
    Dim objMsgNew ' new message
    Dim objMsgs ' collections of new message
    Dim objMsgsprofile ' collections of new message
    Dim objCurrentMsg ' Current message
    Dim objCurrentMsgs ' collection of message
    Dim objFolders
</SCRIPT>
Dim objFolder ' Current folder
Dim objFolderCur ' Current folder
Dim objFolderprofile ' Current folder
Dim objAttachment ' Attachment object
Dim objAttachments ' Attachment collection
Dim objxmlDoc
Dim strEntryID
Dim intCounter
Dim countmsg
Dim subj

' Initialize objects
Set objSession = Nothing
Set objInfoStores = Nothing
Set objInfoStore = Nothing
Set objFields = Nothing
Set objRootFolder =Nothing
Set objMsg = Nothing
Set objMsgNew = Nothing
Set objMsgprofile = Nothing
Set objMsgs = Nothing
Set objMsgsprofile = Nothing
Set objCurrentMsg = Nothing
Set objFolders = Nothing
Set objFolder = Nothing
Set objFolderCur = Nothing
Set objFolderprofile = Nothing
Set objAttachment = Nothing
Set objAttachments = Nothing
Set objxmlDoc = Nothing
Set strEntryID = Nothing
Set subj = Nothing

' Clear error buffer
Err.Clear

' Get session informationen
Set objSession = EventDetails.Session

' No errors detected ?
If Err.Number = 0 Then

  ' Write some logging
  Call DebugAppend(objSession.CurrentUser & " SendAtt - Processsing startet", False)

  ' Get current folder
Set objFolderCur = objSession.GetFolder(EventDetails.FolderID, Null)

Set objCurrentMsgs = objFolderCur.Messages

int countmsg = 0
For Each objCurrentMsg In objCurrentMsgs
    countmsg = countmsg + 1
    Call DebugAppend("Msg " & countmsg, False)
    ' Set current message to read
    objCurrentMsg.Unread = False

    objCurrentMsg.Update
    ' Remember subject of arrived message
    Call DebugAppend("New message with subject: <" & objCurrentMsg.Subject & ">") arrived", False)

    ' Get attachment of the message
    On Error Resume Next
    Set objAttachments = objCurrentMsg.Attachments
    ' No errors detected?
    ' If Not objAttachments Is Nothing Then

    ' Attachments found, write logging
    Call DebugAppend("Attachments found", False)

    ' Extract all attachments
    For Each objAttachment In objAttachments
        ' Check if attachment is a file or link
        ' Note that CDO 1.2x does not support
        ' and embedded messages to the file system
        If (objAttachment.Type = 1) Or (objAttachment.Type = 2) Then
            ' Save attachment to the filesystem, write logging
            Call DebugAppend("Save attachments to filesystem", False)
        End If
        ' Write attachment to the filesystem
        objAttachment.WriteToFile(g_Const_Directory & objAttachment.Name)
    Next
    Exit For

    End If
End For
Next

' Delete attachments
'On Error Resume Next
'objAttachments.Delete
'

objCurrentMsg.Update
objCurrentMsg.delete

' Write some logging, without the folder name
Call DebugAppend("SaveAtt - Processing finished", False)

Set objInfoStores = objSession.InfoStores
For Each objInfoStore in objInfoStores
   If objInfoStore.Name = "Public Folders" Then
      Exit For
   End If
Next

Call DebugAppend("lepas", False)
' Write some logging
Call DebugAppend(objSession.CurrentUser & " SendAtt - Proccessing startet", False)

Dim t1, t2
Dim nokak, napemohon, Jawatan, jab, bakicuti, StDatetext, EDatetext, DurationDay, alamat, notel, datetext, approver, maildisplay2, approval
Dim f, fso
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.OpenTextFile("C:\parsefile.txt", 1)
Call DebugAppend(Now, False)
f.ReadLine
f.ReadLine
nokak = f.ReadLine
f.ReadLine
napemohon = f.ReadLine
f.ReadLine
Jawatan = f.ReadLine
f.ReadLine
jab = f.ReadLine
f.ReadLine
bakicuti = f.ReadLine
f.ReadLine
StDatetext = f.ReadLine
f.ReadLine
EDatetext = f.ReadLine
f.ReadLine
DurationDay = f.ReadLine
f.ReadLine
alamat = f.ReadLine
f.ReadLine
notel = f.ReadLine
f.ReadLine
datetext = f.ReadLine
f.ReadLine
approver = f.ReadLine
f.ReadLine
maildisplay2 = f.ReadLine
f.ReadLine
approval = f.ReadLine
Call DebugAppend(Now, False)
Call DebugAppend("nokak: " & nokak, False)
Call DebugAppend("napemohon: " & napemohon, False)
Call DebugAppend("Jawatan: " & Jawatan, False)
Call DebugAppend("jab: " & jab, False)
Call DebugAppend("bakicuti: " & bakicuti, False)
Call DebugAppend("StDatetext: " & StDatetext, False)
Call DebugAppend("EDatetext: " & EDatetext, False)
Call DebugAppend("DurationDay: " & DurationDay, False)
Call DebugAppend("alamat: " & alamat, False)
Call DebugAppend("notel: " & notel, False)
Call DebugAppend("datetext: " & datetext, False)
Call DebugAppend("approver: " & approver, False)
Call DebugAppend("maildisplay2: " & maildisplay2, False)
Call DebugAppend("approval: " & approval, False)
Call DebugAppend(Now, False)

' Get Leave Application folder
Set objFields = objInfoStore.Fields
strEntryID = objFields.Item(&H66310102).Value
Set objRootFolder = objSession.GetFolder( strEntryID, objInfoStore.ID)
Set objFolders = objRootFolder.Folders
Set objFolder = objFolders.GetFirst()
Do While objFolder.Name <> "Leave Application (SAX)"
    Set objFolder = objFolders.GetNext()
Loop

Set objMsgs = objFolder.Messages
intCounter=0
For Each objMsg In objMsgs
    If objMsg.Fields("nokak").Value(0)= nokak Then
        If objMsg.Fields("napemohon").Value(0)=napemohon Then
            If objMsg.Fields("Jawatan").Value(0)= Jawatan Then
                If objMsg.Fields("jab").Value(0)= jab Then
                    If objMsg.Fields("bakicuti").Value(0) = bakicuti Then
                        If objMsg.Fields("StDatetext").Value(0)= StDatetext Then

If objMsg.Fields("EDatetext").Value(0)= EDatetext Then
  If objMsg.Fields("DurationDay").Value(0)= DurationDay Then
    If objMsg.Fields("alamat").Value(0)= alamat Then
      If objMsg.Fields("notel").Value(0)= notel Then
        If objMsg.Fields("datetext").Value(0)= datetext Then
          Call DebugAppend("masuk", false)
          intCounter=intCounter + 1
          objMsg.Unread = False
          objMsg.Fields.Add "approver", 8, approver
          objMsg.Fields.Add "maildisplay2", 8, maildisplay2
          objMsg.Fields.Add "approval", 8, approval
          If approval="Yes" Then
            objMsg.Subject="Your leave application (" &
            objMsg.Fields("day").Value(0) & objMsg.Fields("month").Value(0)
            & objMsg.Fields("year").Value(0) & "." & objMsg.Fields("eday").Value(0) &
            objMsg.Fields("emonth").Value(0) & objMsg.Fields("eyear").Value(0) & ") has been
            APPROVED"
          End If
          If approval="No" Then
            objMsg.Subject="Your leave application ( ) has been
            REJECTED"
          End If
          objMsg.Fields.Add "reason", 8, reason
          End If
          objMsg.Update

          ' Error detected ?
          If Not Err.Number = 0 Then
            ' Error changing message class
            Call DebugAppend("Error - Could not change message
class", True)
          Else

            ' Message class successfully changed
            Call DebugAppend("Message class successfully
changed", True)
          End If
        End If
      End If
    End If
  End If
End If
Next
Call DebugAppend(Now, False)
If intCounter = 0 Then
  Call DebugAppend(intCounter, False)

  ' Create new message
  Call DebugAppend(Now, False)
  Set objMsgNew = objMsgs.Add
  objMsgNew.Unread = False
  objMsgNew.Type = "IPM.Post.form approval"
  objMsgNew.Subject = "Leave Application from " & napemohon
  objMsgNew.Update
  objMsgNew.Fields.Add "nokak", 8, nokak
  objMsgNew.Fields.Add "napemohon", 8, napemohon
  objMsgNew.Fields.Add "Jawatan", 8, Jawatan
  objMsgNew.Fields.Add "jab", 8, jab
  objMsgNew.Fields.Add "bakicuti", 8, bakicuti
  objMsgNew.Fields.Add "stdatelabel", 8, StDatetext
  objMsgNew.Fields.Add "edatelabel", 8, EDatetext
  objMsgNew.Fields.Add "StDate", 7, StDatetext
  objMsgNew.Fields.Add "StDatetext", 8, StDatetext
  objMsgNew.Fields.Add "EDate", 7, EDatetext
  objMsgNew.Fields.Add "EDatetext", 8, EDatetext
  objMsgNew.Fields.Add "harilabel", 8, DurationDay
  objMsgNew.Fields.Add "DurationDay", 8, DurationDay
  objMsgNew.Fields.Add "addresslabel", 8, alamat
  objMsgNew.Fields.Add "phonelabel", 8, notel
  objMsgNew.Fields.Add "date", 7, datetext
  objMsgNew.Fields.Add "datetext", 8, datetext
  objMsgNew.Fields.Add "approver", 8, approver
  objMsgNew.Fields.Add "maildisplay2", 8, maildisplay2
  objMsgNew.Fields.Add "form", 8, "form approval"
  objMsgNew.Fields.Add "reason", 8, "-
  objMsgNew.Update
  Call DebugAppend(Now, False)
  ' Error detected
  If Not Err.Number = 0 Then
    ' Error changing message class
    Call DebugAppend("Error - Could not change message class", True)
    Else
    ' Message class successfully changed
    Call DebugAppend("Message class successfully changed", True)
  End If
  End If
f.Close

objCurrentMsg.Update
objCurrentMsg.delete

Next
Else
  ' Check for any possible sys errors
  Call DebugAppend("Undefined Error detected", True)
End If

' Clear objects
Set objSession = Nothing
Set objInfoStores = Nothing
Set objInfoStore = Nothing
Set objFields = Nothing
Set objRootFolder = Nothing
Set objMsg = Nothing
Set objCurrentMsg = Nothing
Set objMsgNew = Nothing
Set objFolders = Nothing
Set objFolder = Nothing
Set objFolderCur = Nothing
Set objAttachment = Nothing
Set objAttachments = Nothing
Set objxmlDoc = Nothing
Set strEntryID = Nothing
Set objMsgProfile = Nothing
Set objMsgs = Nothing
Set objMsgsProfile = Nothing
Set objFolderProfile = Nothing

' Write results to the Scripting Agent log
Script.Response = g_bstrDebug

End Sub

' DESCRIPTION: This event is fired when the timer on the folder expires
Public Sub Folder_OnTimer
  ' Not used
End Sub

' DESCRIPTION: This event is fired when a message in the folder is changed
Public Sub Message_OnChange
  ' Not used
End Sub

' DESCRIPTION: This event is fired when a message is deleted from the folder
Public Sub Folder_OnMessageDeleted

End Sub
Private Sub DebugAppend(bstrParm, boolErrChkFlag)
    If boolErrChkFlag = True Then
        If Err.Number <> 0 Then
            g_bstrDebug = g_bstrDebug & bstrParm & " - " & cstr(Err.Number) & " " & Err.Description & vbCrLf
            Err.Clear
        End If
    Else
        g_bstrDebug = g_bstrDebug & bstrParm & vbCrLf
    End If
End Sub

</SCRIPT>
Agent agent script: create form
This agent run when a new item is posted in Agent folder.

<SCRIPT RunAt=Server Language=VBScript>
Option Explicit

'-----------------------------
' Global Variables
'-----------------------------

Dim g_bstrDebug ' Debug String

'-----------------------------
' CONSTANTS
'-----------------------------

Dim g_Const_MBX
Dim g_Const_Directory

' Enter the display name of the mailbox which you want to have send the report to
g_Const_MBX = "Administrator"

' Enter the directory where you want to save all attachments
g_Const_Directory = "c:\xml\"

'-----------------------------
' EVENT HANDLERS
'-----------------------------

' DESCRIPTION: This event is fired when a new message is added to the folder
Public Sub Folder_OnMessageCreated
    Dim objSession ' Session
    Dim objInfoStores
    Dim objInfoStore
    Dim objFields
    Dim objRootFolder
    Dim objMsg ' new message
    Dim objMsgProfile ' new message
    Dim objMsgNew ' new message
    Dim objMsgs ' collections of new message
    Dim objMsgProfile ' collections of new message
    Dim objCurrentMsg ' Current message
    Dim objCurrentMsgs ' collection of message
    Dim objFolder ' Current folder
    Dim objFolderCur ' Current folder
    Dim objFolderProfile ' Current folder
    Dim objAttachment ' Attachment object
    Dim objAttachments ' Attachment collection
    Dim objxmlDoc
    Dim strEntryID
</SCRIPT>
Dim intCounter
Dim countmsg
Dim subj

' Initialize objects
Set objSession = Nothing
Set objInfoStores = Nothing
Set objInfoStore = Nothing
Set objFields = Nothing
Set objRootFolder = Nothing
Set objMsg = Nothing
Set objMsgNew = Nothing
Set objMsgprofile = Nothing
Set objMsgs = Nothing
Set objMsgsprofile = Nothing
Set objCurrentMsg = Nothing
Set objFolders = Nothing
Set objFolder = Nothing
Set objFolderCur = Nothing
Set objFolderprofile = Nothing
Set objAttachment = Nothing
Set objAttachments = Nothing
Set objxmlDoc = Nothing
Set strEntryID = Nothing
Set subj = Nothing

' Clear error buffer
Err.Clear

' Get session informationen
Set objSession = EventDetails.Session

' No errors detected?
If Err.Number = 0 Then

' Write some logging
Call DebugAppend(objSession.CurrentUser & " SendAtt - Processing startet", False)

' Get current folder
Set objFolderCur = objSession.GetFolder(EventDetails.FolderID, Null)

Set objCurrentMsgs = objFolderCur.Messages

int countmsg = 0
For Each objCurrentMsg In objCurrentMsgs
    countmsg = countmsg + 1
    Call DebugAppend("Msg " & countmsg, False)
' Set current message to read
objCurrentMsg.Unread = False
objCurrentMsg.Update
' Remember subject of arrived message
Call DebugAppend("New message with subject: <" & 
objCurrentMsg.Subject & "> arrived", False)

' Get attachment of the message
On Error Resume Next
Set objAttachments = objCurrentMsg.Attachments

' No errors detected?
' If Not objAttachments Is Nothing Then

' Attachments found, write logging
Call DebugAppend("Attachments found", False)

' Extract all attachments
For Each objAttachment In objAttachments

' Check if attachment is a file or link
' Note that CDO 1.2x does not support to save OLE
' and embedded messages to the file system
If (objAttachment.Type = 1) Or (objAttachment.Type = 2) Then

' Save attachment to the filesystem,
Call DebugAppend("Save attachments to filesystem", False)

' Write attachment to the filesystem
objAttachment.WriteToFile(g_Const_Directory & objAttachment.Name)
End If
Next

' Delete attachments
' On Error Resume Next
' objAttachments.Delete
' End If

objCurrentMsg.Update
objCurrentMsg.delete

' Write some logging, without the folder name
Call DebugAppend("SaveAtt - Processing finished", False)
Set objInfoStores = objSession.InfoStores
For Each objInfoStore in objInfoStores
    If objInfoStore.Name = "Public Folders" Then
        Exit For
    End If
Next

Call DebugAppend("lepas", False)
' Write some logging
Call DebugAppend(objSession.CurrentUser & " SendAtt - Proccessing startet", False)

Dim t1, t2
Dim nokak, napemohon, Jawatan, jab, bakicuti, StDatetext, EDatetext, DurationDay, alamat, notel, datetext, approver, maildisplay2, approval

Dim f, fso
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.OpenTextFile("C:\parsefile.txt", 1)
Call DebugAppend(Now, False)
f.ReadLine
f.ReadLine
nokak = f.ReadLine
f.ReadLine
napemohon = f.ReadLine
f.ReadLine
Jawatan = f.ReadLine
f.ReadLine
jab = f.ReadLine
f.ReadLine
bakicuti = f.ReadLine
f.ReadLine
StDatetext = f.ReadLine
f.ReadLine
EDatetext = f.ReadLine
f.ReadLine
DurationDay = f.ReadLine
f.ReadLine
alamat = f.ReadLine
f.ReadLine
notel = f.ReadLine
f.ReadLine
datetext = f.ReadLine
f.ReadLine
approver = f.ReadLine
f.ReadLine
maildisplay2 = f.ReadLine
f.ReadLine
approval = f.ReadLine
Call DebugAppend(Now, False)
Call DebugAppend("nokak: " & nokak, False)
Call DebugAppend("napemohon: " & napemohon, False)
Call DebugAppend("Jawatan: ", Jawatan, False)
Call DebugAppend("jab: ", jab, False)
Call DebugAppend("bakicuti: ", bakicuti, False)
Call DebugAppend("StDatetext: ", StDatetext, False)
Call DebugAppend("EDatetext: ", EDatetext, False)
Call DebugAppend("DurationDay: ", DurationDay, False)
Call DebugAppend("alamat: ", alamat, False)
Call DebugAppend("notel: ", notel, False)
Call DebugAppend("datetext: ", datetext, False)
Call DebugAppend("approver: ", approver, False)
Call DebugAppend("maildisplay2: ", maildisplay2, False)
Call DebugAppend("approval: ", approval, False)
Call DebugAppend(Now, False)

' Get Leave Application folder
Set objFields = objInfoStore.Fields
strEntryID = objFields.Item(&H66310102).Value
Set objRootFolder = objSession.GetFolder( strEntryID, objInfoStore.ID)
Set objFolders = objRootFolder.Folders
Set objFolder = objFolders.GetFirst()
Do While objFolder.Name <> "Leave Application (SAX)"
    Set objFolder = objFolders.GetNext()
Loop
Set objMsgs = objFolder.Messages
intCounter=0
For Each objMsg In objMsgs
    If objMsg.Fields("nokak").Value(0)= nokak Then
        If objMsg.Fields("napemohon").Value(0)=napemohon Then
            If objMsg.Fields("Jawatan").Value(0)= Jawatan Then
                If objMsg.Fields("jab").Value(0)= jab Then
                    If objMsg.Fields("bakicuti").Value(0) = bakicuti Then
                        If objMsg.Fields("StDatetext").Value(0)= StDatetext Then
                            If objMsg.Fields("EDatetext").Value(0)= EDatetext Then
                                If objMsg.Fields("DurationDay").Value(0)= DurationDay Then
                                    If objMsg.Fields("alamat").Value(0)= alamat Then
                                        If objMsg.Fields("notel").Value(0)= notel Then
                                            If objMsg.Fields("datetext").Value(0)= datetext Then
                                                Call DebugAppend("masuk", false)
                                                intCounter=intCounter + 1
                                                objMsg.Unread = False
                                                objMsg.Fields.Add "approver", 8, approver
                                                objMsg.Fields.Add "maildisplay2", 8, maildisplay2
                                                objMsg.Fields.Add "approval", 8, approval
                                                If approval="Yes" Then
                                                    objMsg.Subject="Your leave application (" & objMsg.Fields("day").Value(0) & objMsg.Fields("month").Value(0) & objMsg.Fields("year").Value(0) & ") has been APPROVED"
                                                End If
                                                If approval="No" Then
                                                    Assign that variable to field in 'form approval' form in Leave Application * public folder
objMsg.Subject="Your leave application ( ) has been REJECTED"
objMsg.Fields.Add "reason", 8, reason
End If
objMsg.Update

' Error detected ?
If Not Err.Number = 0 Then
' Error changing message class
    Call DebugAppend("Error - Could not change message class", True)
Else

' Message class successfully changed
    Call DebugAppend("Message class successfully changed", True)
End If
Exit For
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
Next

Call DebugAppend(Now, False)
If intCounter=0 Then

    Call DebugAppend(intCounter, False)

    ' Create new message
    Call DebugAppend(Now, False)
    Set objMsgNew = objMsgs.Add
    objMsgNew.Unread = False
    objMsgNew.Type = "IPM.Post.form approval"
    objMsgNew.Subject="Leave Application from " & napemohon
    objMsgNew.Update
    objMsgNew.Fields.Add "nokak", 8,nokak
    objMsgNew.Fields.Add "napemohon", 8, napemohon
    objMsgNew.Fields.Add "Jawatan", 8, Jawatan
    objMsgNew.Fields.Add "jab", 8, jab
    objMsgNew.Fields.Add "bakicuti", 8, bakicuti
    objMsgNew.Fields.Add "stdatelabel", 8, StDatetext
    objMsgNew.Fields.Add "edatelabel", 8, EDatetext
    objMsgNew.Fields.Add "StDate", 7, StDatetext
    objMsgNew.Fields.Add "StDatetext", 8, StDatetext
    objMsgNew.Fields.Add "EDate", 7, EDate
    objMsgNew.Fields.Add "EDatetext", 8, EDatetext
objMsgNew.Fields.Add "harilabel", 8, DurationDay
objMsgNew.Fields.Add "DurationDay", 8, DurationDay
objMsgNew.Fields.Add "addresslabel", 8, alamat
objMsgNew.Fields.Add "phonelabel", 8, notel
objMsgNew.Fields.Add "date", 7, datetext
objMsgNew.Fields.Add "datetext", 8, datetext
objMsgNew.Fields.Add "approver", 8, approver
objMsgNew.Fields.Add "mailldisplay2", 8, mailldisplay2
objMsgNew.Fields.Add "form", 8, "form approval"
objMsgNew.Fields.Add "reason", 8, ".-"
objMsgNew.Update
Call DebugAppend(Now, False)
  ' Error detected ?
  If Not Err.Number = 0 Then
  ' Error changing message class
  Call DebugAppend("Error - Could not change message class", True)
  Else
  ' Message class successfully changed
  Call DebugAppend("Message class successfully changed", True)
  End If
End If
f.Close

objCurrentMsg.Update
objCurrentMsg.delete
Next
Else
  ' Check for any possible sys errors
  Call DebugAppend("Undefinied Error detected", True)
End If
' Clear objects
Set objSession = Nothing
Set objInfoStores = Nothing
Set objInfoStore = Nothing
Set objFields = Nothing
Set objRootFolder = Nothing
Set objMsg = Nothing
Set objCurrentMsg = Nothing
Set objMsgNew = Nothing
Set objFolders = Nothing
Set objFolder = Nothing
Set objFolderCur = Nothing
Set objAttachment = Nothing
Set objAttachments = Nothing
Set objxmlDoc = Nothing
Set strEntryID = Nothing
Set objMsgprofile = Nothing
Set objMsgs = Nothing
Set objMsgsprofile = Nothing
Set objFolderprofile = Nothing
' Write results to the Scripting Agent log
Script.Response = g_bstrDebug

End Sub

' DESCRIPTION: This event is fired when the timer on the folder expires
Public Sub Folder_OnTimer
    'Not used
End Sub

' DESCRIPTION: This event is fired when a message in the folder is changed
Public Sub Message_OnChange
    'Not used
End Sub

' DESCRIPTION: This event is fired when a message is deleted from the folder
Public Sub Folder_OnMessageDeleted

End Sub

'------------------------------------------------------------------------'
' Name: DebugAppend
' Area: Debug
' Desc: Simple Debugging Function
' Parm: String Text, Bool ErrorFlag
'------------------------------------------------------------------------

Private Sub DebugAppend(bstrParm,boolErrChkFlag)
    If boolErrChkFlag = True Then
        If Err.Number <> 0 Then
            g_bstrDebug = g_bstrDebug & bstrParm & vbCrLf & Err.Description & vbCrLf & vbCrLf
        Else
            g_bstrDebug = g_bstrDebug & vbCrLf & vbCrLf
        End If
    Else
        g_bstrDebug = g_bstrDebug & vbCrLf & vbCrLf
    End If
End Sub
MessageAttachment (SAX) agent script: parsing xml file
This agent run when a new item is posted in MessageAttachment (SAX) folder.

<SCRIPT RunAt=Server Language=VBScript>
Option Explicit
'--------------------------------------------------------------------------------------------------------------------------
' Global Variables
'--------------------------------------------------------------------------------------------------------------------------
Dim g_bstrDebug
' Debug String
'--------------------------------------------------------------------------------------------------------------------------
' CONSTANTS
'--------------------------------------------------------------------------------------------------------------------------
Dim g_Const_MBX
Dim g_Const_Directory

' Enter the display name of the mailbox which you want have send the report to
' g_Const_MBX = "Administrator"

' Enter the directory where you want to save all attachments
' g_Const_Directory = "c:\"
'--------------------------------------------------------------------------------------------------------------------------
' EVENT HANDLERS
'--------------------------------------------------------------------------------------------------------------------------
' DESCRIPTION: This event is fired when a new message is added to the folder
Public Sub Folder_OnMessageCreated
  Dim objSession
  ' Session
  Dim objInfoStores
  Dim objInfoStore
  Dim objFields
  Dim objRootFolder
  Dim objMsg
  ' new message
  Dim objMsgprofile
  ' new message
  Dim objMsgNew
  ' new message
  Dim objMsgs
  ' collections of new message
  Dim objMsgsprofile
  ' collections of new message
  Dim objCurrentMsg
  ' Current message
  Dim objCurrentMsgs
  ' collection of message
  Dim objFolders
  ' Current folder
  Dim objFolder
  ' Current folder
  Dim objFolderCur
  ' Current folder
  Dim objFolderprofile
  ' Current folder
  Dim objAttachment
  ' Attachment object
  Dim objAttachments
  ' Attachment collection
  Dim objxmlDoc
  Dim strEntryID
  Dim intCounter
  Dim countmsg
  Dim subj
</SCRIPT>
' Initialize objects
Set objSession = Nothing
Set objInfoStores = Nothing
Set objInfoStore = Nothing
Set objFields = Nothing
Set objRootFolder = Nothing
Set objMsg = Nothing
Set objMsgNew = Nothing
Set objMsgProfile = Nothing
Set objMsgs = Nothing
Set objMsgsProfile = Nothing
Set objFolders = Nothing
Set objFolder = Nothing
Set objFolderCur = Nothing
Set objFolderProfile = Nothing
Set objAttachment = Nothing
Set objAttachments = Nothing
Set xmlDoc = Nothing
Set strEntryID = Nothing
Set subj = Nothing

' Clear error buffer
Err.Clear

' Get session informationen
Set objSession = EventDetails.Session

' No errors detected?
If Err.Number = 0 Then

' Write some logging
Call DebugAppend(objSession.CurrentUser & " SendAtt - Proccessing startet", False)

' Get current folder
Set objFolderCur = objSession.GetFolder(EventDetails.FolderID, Null)

Set objCurrentMsgs = objFolderCur.Messages

int countmsg = 0
For Each objCurrentMsg In objCurrentMsgs
    countmsg = countmsg + 1
    Call DebugAppend("Msg " & countmsg, False)
    ' Set current message to read
    objCurrentMsg.Unread = False
    objCurrentMsg.Update
    ' Remember subject of arrived message
    Call DebugAppend("New message with subject: <" & objCurrentMsg.Subject & "> arrived", False)
' Get attachment of the message
On Error Resume Next
Set objAttachments = objCurrentMsg.Attachments

' No errors detected?
' If Not objAttachments Is Nothing Then

' Attachments found, write logging
Call DebugAppend("Attachments found", False)

' Extract all attachments
For Each objAttachment In objAttachments

' Check if attachment is a file or link
' Note that CDO 1.2x does not support to save OLE objects
' and embedded messages to the file system
If (objAttachment.Type = 1) Or (objAttachment.Type = 2) Then

' Save attachment to the filesystem, write logging
Call DebugAppend("Save attachments to filesystem", False)

' Write attachment to the filesystem
objAttachment.WriteToFile(g_Const_Directory & objAttachment.Name)

End If
Exit For
Next

' Delete attachments
' On Error Resume Next
' objAttachments.Delete

' End If

Dim wShell
Set wShell = CreateObject("Shell.Application")
Call DebugAppend(Now, False)
wShell.Open "C:\parse.exe"
Call DebugAppend(Now, False)
objCurrentMsg.Update
objCurrentMsg.delete

Call DebugAppend(Now, False)
' Write some logging, without the folder name

Save Cuti.xml to C folder
Run parse.exe to extract Cuti.xml to parsefile.txt and put the txt file in the C folder
Call DebugAppend("SaveAtt - Processing finished", False)

Set objInfoStores = objSession.InfoStores
For Each objInfoStore in objInfoStores
    If objInfoStore.Name = "Public Folders" Then
        Exit For
    End If
Next

Call DebugAppend("lepas", False)
' Write some logging
Call DebugAppend(objSession.CurrentUser & " SendAtt - Processing started", False)

' Get Agent folder
Set objFields = objInfoStore.Fields
strEntryID = objFields.Item(&H66310102).Value
Set objRootFolder = objSession.GetFolder(strEntryID, objInfoStore.ID)
Set objFolders = objRootFolder.Folders
Set objFolder = objFolders.GetFirst()
Do While objFolder.Name <> "Agent"
    Set objFolder = objFolders.GetNext()
Loop

Set objMsgs = objFolder.Messages
Set objMsgNew = objMsgs.Add
objMsgNew.unread = False
objMsgNew.Update

Next
Else
    ' Check for any possible sys errors
    Call DebugAppend("Undefined Error detected", True)
End If
Call DebugAppend(Now, False)
' Clear objects
Set objSession = Nothing
Set objInfoStores = Nothing
Set objInfoStore = Nothing
Set objFields = Nothing
Set objRootFolder = Nothing
Set objMsg = Nothing
Set objCurrentMsg = Nothing
Set objMsgNew = Nothing
Set objFolders = Nothing
Set objFolder = Nothing
Set objFolderCur = Nothing
Set objAttachment = Nothing
Set objAttachments = Nothing
Set objxDoc = Nothing
Set strEntryID = Nothing

Create new message in public folder named Agent
Set objMsgsProfile = Nothing
Set objMsgs = Nothing
Set objMsgsProfile = Nothing
Set objFolderProfile = Nothing
Set wShell = Nothing

' Write results to the Scripting Agent log
Script.Response = g_bstrDebug

End Sub

' DESCRIPTION: This event is fired when the timer on the folder expires
Public Sub Folder_OnTimer
    'Not used
End Sub

' DESCRIPTION: This event is fired when a message in the folder is changed
Public Sub Message_OnChange
    'Not used
End Sub

' DESCRIPTION: This event is fired when a message is deleted from the folder
Public Sub Folder_OnMessageDeleted
End Sub

' **********************************************************
' PRIVATE FUNCTIONS/SUBS
' **********************************************************

'------------------------------------------------------------------------------
'   Name: DebugAppend
'   Area: Debug
'   Desc: Simple Debugging Function
'   Parm: String Text, Bool ErrorFlag
'------------------------------------------------------------------------------

Private Sub DebugAppend(bstrParm, boolErrChkFlag)
    If boolErrChkFlag = True Then
        If Err.Number <> 0 Then
            g_bstrDebug = g_bstrDebug & bstrParm & vbCrLf & Err.Description & vbCrLf
            Err.Clear
        End If
    Else
        g_bstrDebug = g_bstrDebug & bstrParm & vbCrLf
    End If

End Sub