WHAT IS LEGACY DATA?
Legacy data is simply data that has been, and is being, generated by various business processes. The term was originally applied to the data generated by custom-built mainframe applications. Such data is often stored as record files (data sets) and accessed in sequential (VSAM), indexed (ISAM), and other file organizations. It is also commonly stored in relational (e.g. DB2) and hierarchical (e.g., IMS-DB) databases. We take a more expanded view of the definition of legacy data to include custom applications, vendor purchased mainframe applications, and client-server applications.

The overwhelming fact of legacy data is that there is a lot of it and is critical to the business operations of the company that owns them.

UNLOCKING LEGACY DATA
Despite being a critical revenue generation asset in many corporations, largely due to historical and technological reasons, legacy data is often not easily accessible.

By unlocking legacy data, we mean not only providing means of access, but also an idiomatic understanding of the data for both human and computer use. XMLSolutions is a pioneer in recognizing this need and developing a series of XML-based solutions.

Corporate synergy is achieved when legacy data is accessed, understood and utilized across the corporation in a timely manner. This is truer today, and will be more so in the future given the fast paced information economy in which they operate.

Web enabling of legacy applications and data is a goal of many companies, who look forward to leveraging their legacy data assets by migrating to a Web-based communication paradigm. However, web enablement provides mere access, but not necessarily knowledge of what the data represents. Knowing what the data represents leads to methods to transform and use it effectively. The following scenarios explain our vision of what can be accomplished when legacy data is paired with XML.

E-commerce built on legacy data
Unlocking legacy data to e-commerce is of tremendous value for business growth because it provides timely and up to date information on products, pricing, policy, and availability to its customers. Our approach goes a step beyond in laying the groundwork towards automating some of these processes. In effect, we wish to allow legacy applications to interchange data with other legacy applications, and to do so in an anonymous manner.

Figure 1 provides a capsule view of the enabling architecture. Two legacy applications are interfaced to an XML-based e-commerce implementation through special units, which we shall refer to as metadata engines. The metadata engines are shown in gray in the figure. The e-commerce implementation uses XML to
convey data and metadata between the two XML servers anchoring the e-commerce link at both ends. The roles played by the metadata engines are critical in the proper functioning of the scheme because their role is to associate metadata with legacy data in a manner suitable for processing by the legacy applications. We will use three simple examples, Year 2000 compliant dates, Euro Currency, and Part Number to illustrate the functionality of the metadata engine.

![Figure 1](image1.png)

**Figure 1. XML-based legacy data integration into e-commerce.**

**Example 1. Exchanging dates between Year 2000 compliant legacy systems.**

During the recent Year 2000 renovations, two major techniques - date expansion and windowing were used to bring legacy systems into Year 2000 compliance. Consider the following example where two applications having different internal date representations need to communicate with each other. One application is using a Flat file to store a Policy record with the Policy Issue date in MMDDYY format. The following illustration shows the stages of transformation of legacy data from one of the applications prior to its transmittal over the XML-based e-commerce network.

**Figure 1 Legacy COBOL application input**

<table>
<thead>
<tr>
<th>Legacy Application (Cobol record)</th>
<th>Legacy Data (Dataset)</th>
<th>Metadata (Semantic transformation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 POLICY-REC.</td>
<td>...052998...</td>
<td>&lt;POLICY-ISSUED&gt; =&gt; &lt;ISSUE DATE&gt;</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>&lt;FORMAT ISO-8601&gt; &lt;= CC &amp;&amp; YY &amp;&amp;MM&amp;&amp;DD</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>05 POLICY-ISSUED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 MM PIC 9(2).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 DD PIC 9(2).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 YY PIC 9(2).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2 Metadata engine output**

```xml
<POLICY>
  <ISSUE_DATE FORMAT="ISO-8601">1998-05-29</ISSUE_DATE>
</POLICY>
```

The metadata engine (not illustrated) on the other side convert the XML data into the appropriate legacy data suitable for processing by the legacy application.

**Example 2. Handling mixed currencies**

With the advent of Euro, a fourth reserve currency has been added to the current three widely used reserve currencies, the US dollar, the Pound sterling and the Yen. Many legacy applications can now be enabled to handle e-commerce in Euro. In this example, e-commerce interaction is depicted between a US invoicing application and a European (say, French) account payment application. The meta-data engine on the European side integrates the two applications by presenting dollar invoices in Euro augmented by the proper VAT amount.

The fact that the ultimate goal of the interchange is communication between the two legacy applications, US Invoicing and European Accounts Payable, is represented below by the two application names bracketing the XML source. The semantic transformations are listed at the bottom of the example.

**Figure 3 US invoicing application legacy data XML output**

```xml
<POLICY>
  <ISSUE_DATE FORMAT="ISO-8601">1998-05-29</ISSUE_DATE>
</POLICY>
```
In this example, the metadata engine on the European side performs the following semantic transformations before passing the data to the legacy application:

- Converts US Dollars to Euro
- Computes VAT tax payable
- Presents to the Accounts Payable application the correct amount to be paid.

**Example 3: Parts List extraction**

In this example, we illustrate yet another instance of the use of XML tags as metadata. Here, the e-commerce scenario involves a large customer putting forth a bid list of a large number of parts and a vendor who needs to extract only those items that it can supply.

**Figure 5  Customer's consolidated bid list in XML**

```xml
<UNIT>MD-STLOUIS</UNIT>
<DEPARTMENT>EN</DEPARTMENT>
<CONTACT>J.DOE</CONTACT>
<TEL>8001111111</TEL>
<EMAIL>JDOE@MD-STLOUIS.COM</EMAIL>

... ...

<ITEMNO>24</ITEMNO>
  <PARTNUM dimension="124">12T20-2</PARTNUM>
  <QUANTITY>2000</QUANTITY>

... ...

<ITEMNO>52</ITEMNO>
  <PARTNUM dimension="28">12T20-2</PARTNUM>
  <QUANTITY>3000</QUANTITY>

<ITEMNO>53</ITEMNO>
  <PARTNUM dimension="52">12T20-2</PARTNUM>
  <QUANTITY>3000</QUANTITY>

... ...

<BID_OOB format="ISO-8601">1999-08-21</BID_OOB>
```

**Figure 6  Vendor Metadata engine output**

```xml
<CUST>MD-STLOUIS</CUST>
<DEPARTMENT>EN</DEPARTMENT>
<ATTN>J.DOE</ATTN>
<TEL>8001111111</TEL>
<EMAIL>JDOE@MD-STLOUIS.COM</EMAIL>

... ...

<ITEMNO>52</ITEMNO>
  <PARTNUM dimension="28">12T20-2</PARTNUM>
  <QUANTITY>3000</QUANTITY>

<ITEMNO>53</ITEMNO>
  <PARTNUM dimension="52">12T20-2</PARTNUM>
  <QUANTITY>3000</QUANTITY>
```

In this example, the metadata engine on the European side performs the following semantic transformations before passing the data to the legacy application:
In this example, the vendor can supply a certain part number but only in certain sizes. The Metadata engine looks at the part number and its attribute to extract the specific items for bid preparation. In addition, the vendor uses certain tags in his bids to all his customers, which allows him to track his bids effectively. The Metadata engine therefore translates certain tags (UNIT, CONTACT, and BID_OOB) in the customer bid list into tags that the vendor customarily uses in all its bids.

In all three examples discussed above, legacy data is changed in an anonymous manner. This means that the legacy applications process data in the formats that they are familiar with.

AUTOMATING THE CORPORATE LEGACY DATA FLOW: LEVERAGING YOUR Y2K INVESTMENT A SECOND TIME AROUND

Corporate departments are both producers and consumers of vast quantities of legacy data. Yet it is often the case that sharing of legacy data within the corporation is limited. When data is shared, it is often in a specific context, such as a report, memo, or other document. Document data are summaries or extracts from larger data sources and they tend to be formatted to suit the reporting context. Extracting data from a document is often difficult or impossible. When it is possible to extract data from a document, the data may not be current since the document would have been through an editorial process before it was distributed.

Legacy data tends to be complex structurally and semantically. The only sure source of knowledge of these data structures is the legacy application. To utilize legacy data, one needs to know its structure and semantics, which would entail analyzing the legacy application. Corporations have spent vast amounts of resources in fixing the Y2K problem. To fix Y2K problems in legacy data they have found it necessary to analyze the legacy data structures in detail, yielding detailed knowledge of its structure and semantics. This valuable knowledge can be made to pay a second dividend in the context of legacy data flow.

What is legacy data flow?

Legacy data flow is an emerging concept that embodies the idea of making legacy data speedily available where ever it is needed in the corporation, and in a form that is suitable for use.

The key ideas embodied in this concept are:

- Data flow occurs from corporate data generation points to data use points through the corporate intranet.
- Data use points determine the usage of the data. The term “usage” includes aggregation, transformation, formatting and presentation. Thus the same data may be used in different ways at different use points.
- Data semantics, i.e., the meaning assigned to data, determines its usage and therefore a data use point may ascribe its own semantics to the data it receives.
- Data flow is autonomous

There are multiple advantages implicit in this concept. First and most importantly, it provides the conceptual framework under which corporate data is truly integrated and thereby becomes a competitive advantage. Second it relaxes the notion that a corporate mono-culture (terminology, acronyms, etc.), is essential to its success. Data is made available to all groups, and each group interprets it in ways that is useful to its process or task. It thus establishes a kind of data ubiquitous-ness that may prove to be the key ingredient of success. Third, data use points may use data in multiple ways: to generate documents, as input to their applications, for external use such as e-commerce databases or for statistical purposes such as data mining. Fourth, the concept embraces a full range of data atomicity, structure, and type. For example, data atomicity may range from a single product id to a departmental organizational chart, data structures may range from lists, tables, and tree structures to entire documents, and data types may feature both conventional numeric data and non-numeric data such as company or product logos (bit maps).

In such a scenario, the traditional role of documents as collators and interpreters of data is not diminished; rather it is enhanced. Legacy data flow will simply make documents and corporate
communication more precise, and corporate processes run faster and less prone to errors or delays. The usage scenarios described below illustrate these points.

**Usage scenario 1:** The corporate product catalogs generation process get speeded up due to quicker data updates and lesser need for manual editing and error checking. The busy-wait process of data exchange between corporate departments is eliminated.

**Usage scenario 2:** The corporate E-commerce server database can better reflect product availability and pricing changes. This is especially true when complex product configurations or multiple pricing models are involved. Human data entry errors, catalog maintenance delays and other factors that bedevil product databases are reduced.

**Usage scenario 3:** A corporate merger can be consummated more efficiently if both corporations’ legacy data are organized on legacy data flow principles. Disruption in either partner’s operations during the initial days of the merger can be reduced by initially providing bridges between the two data flows. Later, the best practices of each corporation can be incorporated in the merged corporation by retaining the corresponding data flows.

**Usage scenario 4:** A trip wire reporting facility based on the occurrence or non-occurrence of data events may be implemented. Examples include triggers based on critical parts availability, financial flows, personnel issues, etc. More sophisticated versions may provide a data filtering mechanism based on the semantic content of data such as e-mail.

---

**XML-based Legacy data flow**

XML provides the ideal technology framework for implementing the legacy data flow concept. XML, through its tagged data and metadata structure, provides the means to “describe” the data to each use point. Two items of essential metadata are needed for legacy data to work. XML can be used to describe both. The metadata are:

- Context metadata
- Semantic attribute metadata

Context is key because the same data may be interpreted differently in different contexts. A use point may thus use or ignore data depending on context. Context information provides such information as the means by which the data was generated (legacy application name, for example), the data generation point, generation date, time, etc. It would also contain the name of a larger data structure of which it is a part.

Semantic attribute information on the other hand is metadata that is intrinsic to the data itself; and may therefore remain unchanged under different contexts. Simply put, it is the interpretation of the data according to the data generating point.

![XML-based legacy data flow for employee name](image)

*Figure 2. XML-based legacy data flow for employee name*
Figure 2 depicts a simple example of how employee data flow may occur between four departments in a corporation: the Human Resources (HR) department, the Technology Services department, the Payroll department, and the Project Group. Each arc represents data flow between its end points. On each arc, the context and attribute metadata is shown as callouts; the actual data is not. No attempt is made to trace all possible data flow travels between the departments. The information channel can be implemented as a centralized Information channel. The actual flows that occur may occur at different times: possibly Bi-weekly between HR and Payroll, and at unscheduled times between Technical Services and Project Group. Generally, we can expect today’s processes to implement efficient data flow when it is regularly scheduled, as between HR and Payroll. However, it is a different story when it comes to unscheduled data flows.

**XML COMMUNICATION SERVER**

One way to implement the concept of legacy data flow is to start out with a preliminary list of data elements around which individual flows are then constructed. These flows will be multi point data flows, unlike the E-commerce examples that we discussed in the first section. Moreover, each data element will have its own individual data flow tree or graph. In addition, we need to decide when a specific flow occurs- time triggered or event triggered.

To manage these flows simultaneously, we define a XML Communication Server. The XML Communication Server processes XML metadata just like the Metadata engine. Therefore it is not surprising that many of the XML Communication server’s functions overlap with that of the Metadata engine. Among these are homonymic and synonymic transformations of metadata, data formatting, aggregation, and extraction. However, managing the individual data flows still remains the most important task handled by the XML Communications server. In doing so, it must construct one or more contexts for a given data element at the generation point. Similarly, at the use point, the XML Communication server must first evaluate the context, decide the course of action (accept or ignore), extract data and other related information and then perform the meta language transformations necessary to insert the data in the use center’s database.

**Role of data repositories**

The legacy application data repository is a source of information regarding data relationships. Such repositories have been constructed during the Year2000 application analysis phase. In particular, the data dictionaries are the starting points of building the legacy data flows. Each application legacy data contains its own data dictionaries, which are relatively isolated from other data dictionaries. In the legacy data flow implementation, care must be taken to ensure that these repositories are not modified in a manner incompatible with the legacy application that it serves. The bridge between these legacy data repositories is the XML Communications Server Metadata repository. This repository consists of one or more metadata dictionaries spanning multiple legacy applications. Data element descriptions, attribute descriptions, context specifications, and metadata transformation grammars are some of the objects in the metadata repository.

**LEGACY SYSTEM CHALLENGES**

Despite its name, legacy applications are not frozen in time, but are continuously evolving. The challenge is therefore in being able to maintain compatibility between legacy data changes and the metadata descriptions of legacy data. Sometimes these changes are very subtle. For example, the windowing technique used in many Year2000 fixes involves no change in the structure of the application data, but in its interpretation! A metadata repository will have to be able to interpret this change perhaps at the level of individual date elements. Another example, is where apparently unstructured data (e.g. Cobol record FILLERS) is interpreted as structured data within the application, due to new application enhancements. Such examples are not rare scare stories, but rather common occurrences in the world of legacy applications. They are the outcome of a requirement to maintain compatibility with “old” legacy data. High throughput on-line transaction oriented legacy systems may pose another challenge. It may be difficult to design XML boxes that will provide metadata transformation synchronized to the transaction rate.
CONCLUSION
We have described the role that XML can play in “unlocking” legacy data. XML provides the structure to describe metadata, data attributes, and contextual data to allow legacy data to be interpreted according to the needs of the user, which might be another legacy application. Two development paths – bringing legacy data to e-commerce and implementing corporate legacy data flows are described.