XML Implementation Guide:
General Information – Version 2
<table>
<thead>
<tr>
<th>Document Date</th>
<th>Change List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 9 2002</td>
<td>Original Version (Draft)</td>
</tr>
<tr>
<td>Jan 11 2002</td>
<td>Minor corrections to text</td>
</tr>
</tbody>
</table>
| Jan 21 2002   | Minor corrections to text.  
|               | Added schema and LDD samples in Chapter 1.  
|               | Added xsd:Boolean to the “Attribute Types” section of Chapter 2. |
| Feb 14 2002   | Minor corrections to text.  
|               | Added Request and Response Envelope diagrams. |
| Apr 12 2002   | Correct labeling of transaction party in Request Envelope with Submitting Party diagram. |
| Sep 8 2002    | Expand Chapter 3 - MISMO Transaction Envelope to include information about the v2.1.1 “Use Embedded File Indicator” and “MIME Type” attributes that were added to the Preferred Response element. |
| Dec 14 2002   | Add note to Chapter 1 about extensions to MISMO transactions.  
|               | Add note to Chapter 4 regarding exception to substitution of XML reserved characters.  
|               | Add Chapter 5 – HTTP Post Name/Value Pair Recommendation. |
| Dec 26 2002   | Correction to page headers. |
| Aug 28 2003   | Update text in the paragraph in Chapter 1 in the “What About Schemas” section.  
|               | Update Chapter 3 sections, “Preferred Response Element”, and “Submitting Party Element”, with information on Version 2.3 changes.  
|               | Add note to Chapter 4 section “Handling Attributes with no Data”, stating that it doesn’t apply if the attributes are enumerated. |
| Oct 16 2003   | Update Chapter 2 text and example for the IDREF attribute type.  
|               | Update Chapter 4 note about “Handling Attributes with no Data”.  
|               | Change all 40% Gray fonts to 80% Gray to improve readability when document is printed in black and white. |
| Sep 14 2006   | Update Chapter 1 to add Safeguarding Private Borrower Data.  
|               | Update Chapter 1 to change contact information for MISMO Program Manager in the Comments section.  
|               | Insert NEW Chapter 3 – Security Principles.  
|               | Update Chapter 4 (formerly Chapter 3) to add XML Signature section.  
|               | Update Chapter 5 (formerly Chapter 4) section on DOCTYPE usage. |
| Sep 30 2006   | Update Chapter 4 to add Request Envelope Usage Scenarios and Response Envelope Usage Scenarios sections. |

Document Revision Date: September 30, 2006

### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CHAPTER 1: INTRODUCTION</strong></td>
<td>1-1</td>
</tr>
<tr>
<td>Purpose of this Document</td>
<td>1-1</td>
</tr>
<tr>
<td>What Else You Will Need</td>
<td>1-1</td>
</tr>
<tr>
<td>Safeguarding Private Borrower Data</td>
<td>1-1</td>
</tr>
<tr>
<td>Steps Your Organization Can Take</td>
<td>1-2</td>
</tr>
<tr>
<td>State Notification Data Elements (MISMO LDD)</td>
<td>1-4</td>
</tr>
<tr>
<td>What is XML?</td>
<td>1-5</td>
</tr>
<tr>
<td>Elements and Attributes</td>
<td>1-5</td>
</tr>
<tr>
<td>What is a DTD?</td>
<td>1-6</td>
</tr>
<tr>
<td>What about Schemas?</td>
<td>1-6</td>
</tr>
<tr>
<td>What is an LDD?</td>
<td>1-7</td>
</tr>
<tr>
<td>What is MISMO?</td>
<td>1-8</td>
</tr>
<tr>
<td>The MISMO Development Process</td>
<td>1-8</td>
</tr>
<tr>
<td>Mortgage Process Areas</td>
<td>1-9</td>
</tr>
<tr>
<td>Work Groups</td>
<td>1-10</td>
</tr>
<tr>
<td>Version and Release</td>
<td>1-10</td>
</tr>
<tr>
<td>Comments</td>
<td>1-10</td>
</tr>
<tr>
<td><strong>CHAPTER 2: MISMO XML ARCHITECTURE OVERVIEW</strong></td>
<td>2-1</td>
</tr>
<tr>
<td>Transitioning from MISMO Version 1 to MISMO Version 2</td>
<td>2-1</td>
</tr>
<tr>
<td>MISMO Version 1 - The Electronic Loan Package</td>
<td>2-1</td>
</tr>
<tr>
<td>MISMO Version 2 - Transactions</td>
<td>2-1</td>
</tr>
<tr>
<td>MISMO Version 2 - Data Stored in Attributes</td>
<td>2-2</td>
</tr>
<tr>
<td>Elements and Attributes: The DTD Building Blocks</td>
<td>2-4</td>
</tr>
<tr>
<td>Element Types</td>
<td>2-4</td>
</tr>
<tr>
<td>Attribute Types</td>
<td>2-5</td>
</tr>
<tr>
<td>Containment and Pointing Relationships</td>
<td>2-10</td>
</tr>
<tr>
<td>Containment</td>
<td>2-10</td>
</tr>
<tr>
<td>Pointing</td>
<td>2-10</td>
</tr>
<tr>
<td><strong>CHAPTER 3: SECURITY PRINCIPLES</strong></td>
<td>3-1</td>
</tr>
<tr>
<td>General Security Principles</td>
<td>3-1</td>
</tr>
<tr>
<td>Authentication</td>
<td>3-1</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>3-1</td>
</tr>
<tr>
<td>Integrity</td>
<td>3-2</td>
</tr>
<tr>
<td>Non-repudiation</td>
<td>3-2</td>
</tr>
<tr>
<td>Privileged-Based Access Control</td>
<td>3-3</td>
</tr>
<tr>
<td>General Recommendations for Securely Transporting Sensitive Mortgage Information</td>
<td>3-4</td>
</tr>
<tr>
<td>Secure Messaging Requirements for Supporting Electronic Mortgage Processes</td>
<td>3-5</td>
</tr>
<tr>
<td>Recommended Security Solutions</td>
<td>3-6</td>
</tr>
<tr>
<td>Trading Partner-to-Trading Partner Direct Scenario</td>
<td>3-7</td>
</tr>
<tr>
<td>Security Requirements</td>
<td>3-7</td>
</tr>
<tr>
<td>Possible Security Solutions</td>
<td>3-7</td>
</tr>
<tr>
<td>Trading Partners through a Portal Scenario</td>
<td>3-9</td>
</tr>
<tr>
<td>Security Requirements (Variation 1)</td>
<td>3-10</td>
</tr>
<tr>
<td>Security Requirements (Variation 2)</td>
<td>3-10</td>
</tr>
<tr>
<td>Possible Security Solutions</td>
<td>3-11</td>
</tr>
<tr>
<td>Multi-Services Scenario</td>
<td>3-12</td>
</tr>
</tbody>
</table>
Security Requirements...................................................................................................................... 3-12
Possible Security Solutions.................................................................................................................. 3-13
SECURITY DEFINITIONS ...................................................................................................................... 3-15

CHAPTER 4: THE MISMO TRANSACTION ENVELOPE ............................................................. 4-1
The Request Envelope .......................................................................................................................... 4-2
Requesting Party Element .................................................................................................................... 4-2
Preferred Response Element ............................................................................................................... 4-2
Receiving Party Element ..................................................................................................................... 4-4
Submitting Party Element ................................................................................................................... 4-4
Request Element.................................................................................................................................. 4-7
Security Issues with Login Authentication ......................................................................................... 4-8
Request Envelope Usage Scenarios .................................................................................................... 4-8
The Response Envelope ....................................................................................................................... 4-11
Responding Party Element .................................................................................................................. 4-11
Respond To Party Element .................................................................................................................. 4-11
Response Element ............................................................................................................................. 4-12
Response Envelope Usage Scenarios .................................................................................................. 4-13
XML Signature ..................................................................................................................................... 4-15
Signature Element Structure ................................................................................................................ 4-16
Generating the XML Signature ......................................................................................................... 4-16
Validating the XML Signature ............................................................................................................ 4-18

CHAPTER 5: XML IMPLEMENTATION ISSUES............................................................................ 5-1
XML RESERVED CHARACTERS ........................................................................................................... 5-1
Processing Received XML Data.......................................................................................................... 5-1
Generating XML Data .......................................................................................................................... 5-2
Specifying the DTD in the XML Data File .......................................................................................... 5-3
White Space in XML Documents ......................................................................................................... 5-4
Handling Attributes with No Data ...................................................................................................... 5-5
Ordering Elements and Attributes ...................................................................................................... 5-5

CHAPTER 6: HTTP POST NAME/VALUE PAIR RECOMMENDATION........................................... 6-1
Chapter 1: Introduction

Purpose of this Document

This document is designed to assist individuals who are implementing the MISMO XML standards by providing helpful information and sample XML data. Although it is not intended as an XML tutorial, certain aspects of XML that are important for the proper implementation of the standard are highlighted. The guide will also give a brief background of the MISMO effort, followed by an overview of the data architecture and sample data.

What Else You Will Need

This guide covers general aspects of the MISMO standard. To fully implement the standard, MISMO provides other files that can be downloaded from the MISMO web site (http://www.mismo.org).

- **Process Area Implementation Guides** - Separate implementation guides will be provided for each of the mortgage process areas like appraisal, credit reporting, mortgage insurance, mortgage application, title, underwriting and so on. These provide more detailed guidance for implementing the standard.

- **DTD Files** – The Document Type Definition (DTD) files define the structure of each of the process area data sets. These files are utilized to develop, write and read XML data files.

- **LDD Files** – Each mortgage process area also provides a Logical Data Dictionary (LDD), which defines each data element used in the DTD.

- **Your Own Data** – The normal purpose of implementing an XML standard is to either convert your own data to an XML format or to convert your data from an XML format.

Safeguarding Private Borrower Data

Before beginning the discussion of XML data, it is important to note that significant portions of the data in mortgage industry transactions is private personal information that must be safeguarded. The Mortgage Bankers Association and MISMO have provided two documents that discuss this topic.


- “Identifying and Safeguarding Personal Information: Guidelines & Practices”
Steps Your Organization Can Take

Organizations should take steps to protect their electronic data environment. The biggest step your organization can take is acknowledging that data can be a critical asset, and that it needs to be managed and secured like any other critical asset within your organization. Next, every organization needs to understand that a solution for securing sensitive information is not solely a technical solution, but one that involves people and processes as well. The MISMO Information Security Work Group (ISWG) has been promoting a general five-step security method in all of its security activities. This same five-step method, which is consistent with ISO 17799, can be used by any mortgage institution to identify, assess and safeguard information. This method is also useful in performing activities required to comply with industry regulations such as the Federal Trade Commission (FTC) Safeguards Rule as well as the various State legislations addressing notification requirements for security breaches involving disclosure of personal information.

In summary, this method involves these five steps:

Business and Risk Description

Simply stated, the risk is not protecting sensitive information and the ramifications for not protecting that information are legislative and regulatory compliance. Business descriptions are use cases specific to your organization where sensitive information is handled. The ISWG has generally described these use cases as collecting, processing, transferring, storing and disposing sensitive information. Mortgage companies should use these general use cases to identify in more detail the use cases that are specific to their environment, where environment is defined as the physical environment (e.g., buildings, offices), the logical environment (e.g., networks), and the legal environment (e.g., security breach notification laws, consumer protection laws, required security audits). The result of this activity is a detailed understanding of where sensitive information exists and how it should be handled accordingly within your company.

Policy and Architecture

This is the foundation for protecting sensitive information within your organization. The policy defines the high level requirements for securely managing information and in the case where a breach occurs, for providing incident response notification. The architecture is the framework for implementing specific technical and procedural solutions in support of your company’s policy (e.g., segregation of responsibilities and infrastructure, interconnectivity).
Solution Specification

These are the detailed specifications for your organization to comply with your policy and to be implemented in accordance with your architecture. People need to be informed and trained on requirements for handling sensitive information; processes need to be put in place to ensure every individual and computer operates correctly with respect to the handling of sensitive information; and technology needs to be selected and implemented that provides the appropriate level of security (e.g., encryption, access control, auditing, intrusion detection, anti-virus, regulatory compliance).

Support Plan

Information theft and the monitoring and notification of security breaches is an evolving landscape. Your organization should identify individuals who have a responsibility to keep up with this changing landscape (e.g., new laws, new information theft tactics, new security technologies and best practices). By keeping up with the changing landscape, your organization can adapt quickly and implement new solutions (or enhance existing solutions) for protecting your critical information assets. Business Continuity Plan/Disaster Recovery, and maintenance plans including change control are elements of a Support Plan.

Education

Education and awareness may be the single most important program your organization performs regarding the protection of sensitive information. The more your management, employees, contractors, etc. understand the importance for protecting sensitive information and the reputation benefits that can be gained by being an advocate of protected sensitive information, the more successful your organization will be in implementing information security solutions.

If your company needs consultation services, there are many organizations (large and small) that can assist your company through the 5-step method above. Companies with expertise in or offering ISO 17799 compliance services are good candidates. It is highly recommended that you clearly define your initiative as protecting sensitive information and you should ensure that any consultants you hire are able to tailor their services appropriately.
State Notification Data Elements (MISMO LDD)

MISMO has made an effort to designate the individual data elements which have been identified as Personal Information in one or more state breach notification laws by setting a flag on these items in the LDD. Examples of Personal Information identified by state legislation include Social Security Number, financial account numbers, and Driver’s licenses. Additionally, State legislation generally requires a name in combination with personal information, e.g., consumer name and Social Security Number, or name and financial account number. Since the exact requirements vary from state to state the flag in the LDD has been defined to designate a data element which is specified in one or more state laws as being Personal Information either by itself or when combined with certain other flagged field(s).

MISMO is not a regulatory agency and the designation of Personal Information data elements is an exercise in security awareness. The objective of flagging these data elements is to trigger a thorough risk assessment by organizations. Business Analysts, Privacy Officers, Counsel, Application Developers and Information Technologists should work together to ensure risk associated with the use of these (and potentially other) data elements are properly mitigated. Organizations are advised to examine their corporate policies for any additional data elements.
What is XML?

XML is an acronym for Extensible Markup Language. (If you’re wondering why it’s not called EML, you’re not alone.) Data alone does not provide the information a computer needs to properly process and store the data. When we add “markup language” to the data, the purpose of each element of the data becomes clear. It may be obvious to us (but not the computer) that Jonathan is a first name and that Consumer is a last name. The markup language also tells us that Jonathan is a borrower, and that the address information provided is his residence.

Data without Markup Language

JONATHAN CONSUMER
3750 S BRANDYWINE ST # 242
LAS VEGAS NV 89103

Data with Markup Language

<BORROWER _FirstName="JONATHAN" _LastName="CONSUMER">
  <_RESIDENCE _StreetAddress="3750 S BRANDYWINE ST"
              _City="LAS VEGAS" _State="NV" _PostalCode="89103"/>
</BORROWER>

What about the Extensible part of XML? To extend or add to the existing data, all we need to do is add the new data along with its markup language label. (NOTE: Some MISMO XML transactions may contain pre-defined methods for “extending” the standard to add new data. See the appropriate process area’s Implementation Guide for more information.)

Data with Markup Language - Extended

<BORROWER _FirstName="JONATHAN" _LastName="CONSUMER"
          _FRE:NativeLanguage="ENGLISH">
  <_RESIDENCE _StreetAddress="3750 S BRANDYWINE ST"
              _City="LAS VEGAS" _State="NV" _PostalCode="89103"/>
</BORROWER>

Elements and Attributes

In the above example, BORROWER and RESIDENCE are elements of our sample data. Element names begin with a bracket (<) and end, after any attributes, with another bracket (>).

Elements can also have attributes that describe them more completely. Attributes names are followed by an equal sign (=) and the data enclosed in quotes. BORROWER has attributes of First Name, Last Name and Native Language. RESIDENCE has attributes of Street Address, City, State and Postal Code.

There will be more discussion on how elements and attributes are used in the MISMO Version 2 standards, beginning on page 2-4.
What is a DTD?

DTD is an acronym for Document Type Definition. This is a file that defines the “markup language” that will be used to describe the data. The following sample is the DTD that describes the borrower data on the previous page. There are plenty of books and web sites available that explain DTDs. All we are doing here is showing what a simple DTD looks like.

DTD for the Borrower Sample Data

```xml
<!ELEMENT BORROWER { _RESIDENCE* }>
<!ATTLIST BORROWER _FirstName CDATA #IMPLIED
_LastName CDATA #IMPLIED >

<!ELEMENT _RESIDENCE EMPTY >
<!ATTLIST _RESIDENCE _StreetAddress CDATA #IMPLIED
_City CDATA #IMPLIED
_State CDATA #IMPLIED
_PostalCode CDATA #IMPLIED >
```

What about Schemas?

The DTD is only one of several methods available for describing what markup language will be used with a particular set of XML data. A newer method is called the Schema or XSD format, which provides more flexibility and the ability to more precisely describe the data. MISMO is currently working on designing its Version 3 specifications, which will use the Schema/XSD format for defining XML data.

Schema for the Borrower Sample Data

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="BORROWER">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="_RESIDENCE" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="_RESIDENCE">
    <xsd:complexType>
      <xsd:attribute name="_StreetAddress" use="optional" type="xsd:string"/>
      <xsd:attribute name="_City" use="optional" type="xsd:string"/>
      <xsd:attribute name="_State" use="optional" type="xsd:string"/>
      <xsd:attribute name="_PostalCode" use="optional" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```
**What is an LDD?**

LDD is an acronym for the Logical Data Dictionary. The LDD is in a table format that provides definitions for each of the elements used in a MISMO DTD. In MISMO Version 2, there is a consistent linkage between the LDD name and the name used in the XML file. An underscore in front of an attribute or element indicates that its LDD entry would also include the name of its containing element.

```xml
<BORROWER _FirstName="JONATHAN" _LastName="CONSUMER">
  <_RESIDENCE _StreetAddress="3750 S BRANDYWINE ST"
    _City="LAS VEGAS" _State="NV" _PostalCode="89103"
    BorrowerResidencyDurationYears="2"/>
</BORROWER>
```

To derive the LDD entry for _FirstName...

```
BORROWER + _FirstName = “Borrower First Name”
```

To derive the LDD entry for _PostalCode...

```
BORROWER + _RESIDENCE + _PostalCode = “Borrower Residence Postal Code”
```

To derive the LDD entry for BorrowerResidencyDurationYears...

```
BorrowerResidencyDurationYears = “Borrower Residency Duration Years”
```

### Sample Entries from Version 2 Logical Data Dictionary

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Source</th>
<th>Context</th>
<th>Processes</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Borrower First Name</td>
<td>The first name of the borrower. Collected on the URLA in Section III (Borrowers Name).</td>
<td>URLA</td>
<td>Borrow</td>
<td>Service ordering, Credit Reporting, Flood, MI, Secondary, AUS</td>
<td>String</td>
</tr>
<tr>
<td>Borrower Residence Postal Code</td>
<td>The postal code (zip code in the US) of the address of borrowers residence. Zip code may be either 5 or 9 digits. Collected on the URLA in Section III (Present Address).</td>
<td>URLA</td>
<td>Borrow</td>
<td>Credit Reporting, MI, Secondary, AUS</td>
<td>String</td>
</tr>
<tr>
<td>Borrower Residency Duration Years</td>
<td>The number of years the borrower resided at the indicated address. Could be either current or prior address. Collected on the URLA in Section III (Present Address or Former Address).</td>
<td>URLA</td>
<td>Borrow Residence</td>
<td>Credit Reporting, MI, Secondary, AUS</td>
<td>Numeric</td>
</tr>
</tbody>
</table>
What Is MISMO?

The Mortgage Bankers Association (MBA) created the Mortgage Industry Standards Maintenance Organization (MISMO), in October 1999. MISMO gathered individuals from a widely varied group of mortgage industry leaders with extensive business knowledge about the industry. This body of individuals creates the Logical Data Dictionary (LDD) that defines the meaning of each business data element used within the mortgage industry. The creation of the LDD has been the key to the success of the MISMO effort. This data dictionary is the seed for generating the XML structures or any other type of structure that may be used in the future.

The result is a single common data set for the mortgage industry. The borrower, employment, property and other commonly used information have a common data definition, no matter which mortgage process is using the data.

The MISMO Development Process

The first and probably most important product developed by the MISMO work group is the data dictionary or LDD that was mentioned earlier. Each work group first examines the existing “core data” elements that are used in common by more than one process area. They then work to define additional data elements that are needed for their specific area of data coverage. The data dictionary defines all data elements that become the basis for organizing the XML Document Type Definition (DTD) or data schema that will be used in the future.

Process area work groups (i.e. Appraisal, Credit Reporting, Mortgage Insurance, Underwriting, etc.) identify relevant data points and containers. A representative from each work group is responsible for entering the data into a web-enabled tool that warehouses the data dictionary. The work group also then defines the XML DTDs needed to support transactions for their process area. For example, for mortgage services there will normally be a DTD defined to request a service, and a DTD defined for the response from the service provider. Changes made to the data points and containers are monitored by the Core Data Work Group to ensure the integrity of all the DTDs.

An XML Architecture Work Group makes key decisions about the XML standards and their implementation within the mortgage industry. The representatives of the various mortgage process area work groups meet frequently to iron out issues about the data, definitions, and organization of commonly used business data.

There is a document titled, MISMO XML Development Rules and Guidelines, which can be downloaded from the MISMO web site (http://www.mismo.org). This document provides more detailed information on the process of developing MISMO XML standards.
Mortgage Process Areas

MISMO has organized the business process areas across the mortgage industry under four broad categories: Origination, Servicing, Secondary, and Real Estate Services. Processes under Origination include Mortgage Application, Underwriting and Closing. Servicing includes Loan Setup & Transfer, Investor Reporting and Default Reporting. Secondary includes Securitization, Bulk Pool Transfer, Funding, and Pricing & Discovery. Real Estate Services include Appraisal, Credit Reporting, Flood, Mortgage Insurance and Title. Business process areas are still being identified and are subject to modification as the work progresses. At the center of the process areas is “Core Data”, which contains data common to multiple process areas, like Borrower and Property elements.
Work Groups

A list of the current MISMO Work Groups, their purpose and contacts can be found at the http://www.mismo.org web site. The web site also has a page that allows you to sign up for the various Work Group List Serves. The work groups use the List Serves to send email to its members regarding conference calls, release of draft documents, or exchange of other information.

Version and Release

This implementation guide is based on Version 2 of the MISMO standard. The MISMO Logical Data Dictionary and DTDs can be downloaded from the www.mismo.org web site.

Comments

Comments, questions, and suggestions for improvement of this document may be submitted in writing to the Secretariat who will forward them to the appropriate MISMO Work Group.

MISMO Program Manager      (Email: info@mismo.org)
1919 Pennsylvania Avenue, NW
Suite 700
Washington, DC 20006-3404
Chapter 2: MISMO XML Architecture Overview

Transitioning from MISMO Version 1 to MISMO Version 2

Even if you never implemented the MISMO Version 1 XML standards, it may be helpful to have an overview of the basic philosophy behind the Version 1 and Version 2 architectures.

MISMO Version 1 - The Electronic Loan Package

The MISMO Version 1 architecture revolved around the concept of an electronic loan package. The idea was that loan data from the various mortgage process areas could eventually be merged into an individual loan “package” like pieces of a pie. One segment of the “pie” might contain loan application data. Other segments might hold credit report data, mortgage insurance data, title data, appraisal data, loan data, etc. In the middle of the “pie” was “core data” that was common to all mortgage process areas, like borrower and property information.

MISMO Version 1 DTDs all had a common structure. They all used MORTGAGEDATA as the root element. BORROWER data was always under MORTGAGEDATA, whether it was used in the Underwriting DTD, Credit Reporting DTD, or the Mortgage Insurance DTD.

MISMO Version 1 – Borrower Data (ALL DTDs)

MISMO Version 2 - Transactions

In MISMO Version 2, there has been a move towards more “transactional” structures. Rather than focusing on the building of a single loan package, the MISMO Version 2 DTDs are more oriented towards the commerce of exchanging business data as transactions. For example, there is a DTD for requesting an automated underwriting decision. There are others for requesting credit reports, appraisals, flood certifications, mortgage insurance, and title insurance. Still other DTDs are focused on the responses to each of the request transactions, such as the actual credit report data, appraisal data, etc.

Although different DTDs may re-use many of the same structures, they are organized in a manner that is optimal for each transaction. Many DTDs may include the commonly used BORROWER data element, but depending on the transaction it may be located under different element names as shown in the following examples.
While MISMO Version 1 DTDs all used MORTGAGEDATA as a root element name, each MISMO Version 2 DTD may have different root element names, depending on their purpose. The AUS DTD uses LOAN_APPLICATION as the root element name. The Credit Request DTD uses the MISMO transaction envelope’s REQUEST_GROUP element as its root element name. The Credit Response DTD uses the RESPONSE_GROUP element as its root element name. (See “Chapter 4: The MISMO Transaction Envelope” for more information on the MISMO envelope structures).

MISMO 2 – Borrower Data (AUS DTD)

```xml
<LOAN_APPLICATION>
  ... other data ...
  <BORROWER>... other borrower data ...</BORROWER>
</LOAN_APPLICATION>
```

MISMO 2 – Borrower Data (Credit Request DTD)

```xml
<REQUEST_GROUP>
  ... other "envelope" data ...
  <CREDIT_REQUEST>
    ... other data ...
    <LOAN_APPLICATION>
      ... other data ...
      <BORROWER>... other borrower data ...</BORROWER>
    </LOAN_APPLICATION>
  </CREDIT_REQUEST>
</REQUEST_GROUP>
```

MISMO 2 – Borrower Data (Credit Response DTD)

```xml
<RESPONSE_GROUP>
  ... other "envelope" data ...
  <CREDIT_RESPONSE>
    ... other data ...
    <BORROWER>... other borrower data ...</BORROWER>
  </CREDIT_RESPONSE>
</RESPONSE_GROUP>
```

MISMO Version 2 - Data Stored in Attributes

The MISMO Version 1 DTDs, both XML attributes and elements were used for storing data. Elements require a “start” tag and an “end” tag around the data element and these were used for storing most types of data. The example below shows a simple set of borrower data represented as XML elements.

MISMO Version 1 – Borrower Data (~150 bytes)

```xml
<BORROWER BORROWERID="ssn530889999">
  <FirstName>MARY</FirstName>
  <MiddleName>ANNE</MiddleName>
  <LastName>SMITH</LastName>
  <SSN>530889999</SSN>
</BORROWER>
```
In MISMO Version 2, most data is stored using the XML attributes. Attributes use a tag name followed by an “equals sign” and then the data enclosed in quotes. Notice that the label identifying the data only appears once, before the data (unlike elements where the label appears before and after the data). This makes the file sizes smaller, which can be important in an environment processing high volumes of transactions.

MISMO Version 2 – Borrower Data (~100 bytes)

```xml
<BORROWER BorrowerID="ssn530889999"
_FirstName="MARY"
_MiddleName="ANNE"
_LastName="SMITH"
_SSN="530889999"/>
```
**Elements and Attributes: The DTD Building Blocks**

XML provides two types of structures to define data --- elements and attributes. There are no set XML rules for how these two structures are to be used, but the MISMO XML Architecture Work Group has defined guidelines for their use, so that a uniform set of XML data structures can be defined for the mortgage industry.

In MISMO Version 2 DTDs, **Elements** are primarily used to “contain” other elements and attributes. **Attributes** are primarily used to hold the actual data values.

**Element Types**

There are three types of XML elements commonly used in the MISMO DTDs:

**Container Elements** – These elements are primarily used to contain other container elements, empty elements and may also have attributes. Container element names will always be in UPPERCASE letters, and will always have a start tag, and an end tag.

```xml
<BORROWER _FirstName="MARY" _MiddleName="ANNE" _LastName="SMITH" _SSN="530889999" _UnparsedName="MARY ANNE SMITH" MaritalStatusType="Married">
  <RESIDENCE _StreetAddress="4112 N GRAND AV # 242" _City="ATLANTA" _State="GA" _PostalCode="30050" BorrowerResidencyDurationYears="2"/>
</BORROWER>
```

**Empty Elements** – These elements will only contain attributes. Empty element names will always be in UPPERCASE letters, and will always have a start tag, and end with a forward slash and the closing bracket.

```xml
<PURCHASE_CREDIT _Amount="14000" _SourceType="Employer" _Type="RelocationFunds"/>
```

**Repeating Elements** – In XML, elements can be defined to appear more than one time, attributes cannot. While in the MISMO Version 2 DTDs most data is stored in attributes; some DTDs may define repeating elements for holding data such as repeating lines of remarks or comments. Since repeating elements contain actual data, their element names do not use all uppercase letters in their names, like container elements and empty elements.

```xml
<CREDIT_COMMENT _SourceType="RepositoryBureau">
  <Text>Revolving Charge Account</Text>
  <Text>Card Lost</Text>
  <Text>Account Closed</Text>
</CREDIT_COMMENT>
```
Attribute Types

In the MISMO Version 2 architecture, attributes are used as the primary method for storing data. MISMO has defined several categories of attributes for this purpose.

Enumerated Attributes – This type of attribute has an enumerated list of allowable values defined in the MISMO DTD or Schema. When an enumerated attribute is used in an XML data file, only a value defined in the enumerated list may be used.

The following section from the AUS DTD shows an enumerated attribute list for the _CalculationType attribute of the RATE_ADJUSTMENT element. This attribute has three allowed values: “Add Percent To Current Rate”, “Add Percent To Original Rate” and “Index Plus Margin”.

```xml
<!ATTLIST RATE_ADJUSTMENT
   _CalculationType (AddPercentToCurrentRate | AddPercentToOriginalRate | IndexPlusMargin ) #IMPLIED>
```

The following is a sample of XML data that shows how the data defined in the DTD sample above would appear. The selected Enumerated Attribute value is enclosed in quotes.

```xml
<RATE_ADJUSTMENT
   FirstRateAdjustmentMonths="25"
   _CalculationType="AddPercentToOriginalRate"
   _DurationMonths="12"
   _Percent="0.5"/>
```

String Attributes – This type of attribute contains a text string, which could be a code, word or phrase. Unlike the enumerated attribute there is no list of valid values provided in the DTD or Schema file.

All of the attributes in the borrower data below – First Name, Last Name, Street Address, City, State and Postal Code - are string attributes.

```xml
<BORROWER _FirstName ="JONATHAN" _LastName="CONSUMER">
   <_RESIDENCE _StreetAddress="3750 S BRANDYWINE ST"
               _City="LAS VEGAS" _State="NV" _PostalCode="89103"/>
</BORROWER>
```
**Boolean Attributes** – This is a special type of enumerated attribute that always has the values “Y” or “N” representing a “Yes” or “No” value. The *DECLARATIONS* section of the AUS DTD has several Boolean attributes as shown in the partial DTD sample below.

```xml
<!ATTLIST DECLARATION
  AlimonyChildSupportObligationIndicator (Y|N) #IMPLIED
  BankruptcyIndicator (Y|N) #IMPLIED
  BorrowedDownPaymentIndicator (Y|N) #IMPLIED
  CoMakerEndorserOfNoteIndicator (Y|N) #IMPLIED
  LoanForeclosureOrJudgementIndicator (Y|N) #IMPLIED
  OutstandingJudgementsIndicator (Y|N) #IMPLIED
  PresentlyDelinquentIndicator (Y|N) #IMPLIED />
```

The following sample shows how the data defined in the DTD sample above would appear in an XML data file.

```xml
<DECLARATION
  AlimonyChildSupportObligationIndicator="N"
  BankruptcyIndicator="N"
  BorrowedDownPaymentIndicator="Y"
  CoMakerEndorserOfNoteIndicator="Y"
  LoanForeclosureOrJudgementIndicator="N"
  OutstandingJudgementsIndicator="N"
  PresentlyDelinquentIndicator="N" />
```

**xsd:Boolean Attributes** – MISMO has also adopted a second Boolean data type that corresponds to the *w3c* (World Wide Web Consortium) schema primitive data type of Boolean. It is represented in a MISMO DTD as an enumerated attribute that may have valid values of "0", "1", "false", or "true".

The following shows how the *xsd:Boolean* data type is declared in the *MERS* element of the Servicing Transfer DTD.

```xml
<!ELEMENT MERS EMPTY>
<!ATTLIST MERS
  MERS_ID ID #IMPLIED
  MERSMortgageIdentifier CDATA #IMPLIED
  MERSRegistrationIndicator (true | false | 0 | 1) #IMPLIED
  MERSMortgageeOfRecordIndicator (true | false | 0 | 1) #IMPLIED />
```

The following samples show two equivalent representations of sample XML data for the *MERS* element.

```xml
<MERS MERS_ID="MERS001"
  MERSMortgageIdentifier="1234567890"
  MERSRegistrationIndicator="true"
  MERSMortgageeOfRecordIndicator="false" />

<MERS MERS_ID="MERS001"
  MERSMortgageIdentifier="1234567890"
  MERSRegistrationIndicator="1"
  MERSMortgageeOfRecordIndicator="0" />
```
**Date/Time Attributes** – MISMO has adopted the ISO 8601 international standard for representing dates and times. The Date/Time element can hold a date only, or a combined date and time. A full date is formatted in a CCYY-MM-DD format as in the date shown below.

```
InterviewerApplicationSignedDate="2000-03-10"
```

If there is no “day” value for a date it will be stored in a CCYY-MM format as in the following sample credit liability account opened date.

```
_AccountOpenedDate="2000-03"
```

When a time is included in the element, the date and time are separated by the letter “T” as shown in the Request Date Time element shown below. The time portion is set in a HH:MM:SS format. If the “seconds” value is not available the time is set in the HH:MM format. Time values use the 24-hour format (i.e. 11:00 = 11am, 13:00 = 1pm, 14:00 = 2pm, etc.).

```
ResponseDateTime="2000-04-30T10:13:59"
```

**Money Attributes** – This MISMO-defined attribute holds money values. Money attributes have the same character limitations as numeric attributes. Fractional dollar amounts are expressed to two decimal places. Whole dollar amounts do not have to include the “.00” decimal value and should not contain dollar signs or commas. The money attribute values are always assumed to be in U.S. dollars.

**Valid Values**

```
BaseLoanAmount="225000.00"
```

```
PropertyAppraisedValueAmount="240000"
```

```
_MonthlyPaymentAmount="1934.85"
```

**Invalid Values**

*(Invalid - contains a comma)*

```
BaseLoanAmount="225,000.00"
```

*(Invalid – contains a dollar sign)*

```
BaseLoanAmount="$225000"
```
Numeric Attributes – Numeric attributes are used for “non-money” data, like social security numbers, rates, percents, counts or totals, etc. Even though DTDs cannot enforce data types, numeric attributes should only contain the numbers “0” through “9”, plus or minus signs and the decimal point.

_MonthsReviewedCount="5"

Two specific types of numeric attributes are described here in more detail - Rate attributes and Percent attributes. A Rate is a numeric comparison between two values, a fraction that is expressed as a decimal. A Percentage is a number representing a part of a whole that is represented as a quotient multiplied by 100.

Rate attributes represent a ratio that is multiplied directly against a value to produce a result. For example, the Tax Amount on a $500 item with a Tax Rate of .05 is $25. To express the Tax Rate of .05 as a Tax Percent multiply the Rate times 100 (.05 x 100 = 5, or 5%). The following attribute samples both express the same value, the first one as a rate, and the second one as a percent.

StateSalesTaxRate=".055"
StateSalesTaxPercent="5.5"

ID Attributes – ID attributes can be used to identify a specific instance of a repeating element. They provide a mechanism for quickly locating specific data elements. Each XML ID attribute value used in an XML data file must be unique throughout the entire file. It must also be a valid XML name – that is, it begins with a letter and is composed of alphanumeric characters and the underscore without white space (spaces, tabs, carriage returns, line feeds).

One method of making sure the ID attributes are unique while maintaining a numeric count is to combine an alpha identifier with a zero padded number. The example below shows two BORROWER container elements - one with a BorrowerID attribute of “BorRec0001” and the other with a BorrowerID attribute of “BorRec0002”. The BorrowerID could just as easily have values like “Borrower-1”, “JonathanConsumer”, or “ssn548603388” (his Social Security Number).

<BORROWER BorrowerID="BorRec0001"
JointAssetBorrowerID="BorRec0002"
_FirstName="JONATHAN" _LastName="CONSUMER"
_SSN="548603388"
_UnparsedName="JONATHAN CONSUMER"/>

<BORROWER BorrowerID="BorRec0002"
JointAssetBorrowerID="BorRec0001"
_FirstName="JANE" _LastName="CONSUMER"
_SSN="599019999"
_UnparsedName="JANE CONSUMER"/>
IDREF Attributes – IDREF attributes are used to “refer to” a specific ID attribute in another element. The ID attribute for the BORROWER record in the previous example, has a value of “BorRec0001”. In the AUS Loan Application DTD, the JointAssetBorrowerID IDREF attribute “refers to” or identifies the other borrower element with whom the borrower jointly owns the assets listed in the loan application. In this example, the data is saying that the Jonathan Consumer (whose BorrowerID = “BorRec0001”), has joint assets with Jane Consumer (JointAssetBorrowerID = “BorRec0002”, which is Jane’s BorrowerID value).

```xml
<BORROWER BorrowerID="BorRec0001"
    JointAssetBorrowerID="BorRec0002"
    _FirstName="JONATHAN" _LastName="CONSUMER"
    _SSN="548603388"
    _UnparsedName="JONATHAN CONSUMER"/>
</BORROWER>

<BORROWER BorrowerID="BorRec0002"
    JointAssetBorrowerID="BorRec0001"
    _FirstName="JANE" _LastName="CONSUMER"
    _SSN="599019999"
    _UnparsedName="JANE CONSUMER"/>
</BORROWER>
```

NOTE: A more detailed discussion of ID, IDREF and IDREFS attributes begins on the next page.
**Containment and Pointing Relationships**

**Containment**

When organizing data, it makes sense to place data like name, birth date, SSN, and employment directly within the confines of a Borrower container, since that data belongs directly to each borrower. Within MISMO, this method of organization is called **containment**. The majority of the data within the MISMO DTDs, especially in Version 2 DTDs, use the containment method to organize the data.

**Example Illustrating “Containment” Relationships**

```xml
<LOAN_APPLICATION>
  <BORROWER BorrowerID="ssn111111111" _UnparsedName="JOHN DOE">
    <EMPLOYER _Name="Ace Casino Supplies"
      EmploymentPositionDescription="VP Sales"
      IncomeEmploymentMonthlyAmount="7500"/>
  </BORROWER>

  <BORROWER BorrowerID="ssn222222222" _UnparsedName="JANE DOE">
    <EMPLOYER _Name="Mandalay Bay Hotel"
      EmploymentPositionDescription="Reservations Mgr"
      IncomeEmploymentMonthlyAmount="5100"/>
  </BORROWER>
</LOAN_APPLICATION>
```

**Pointing**

A potential problem with containment can occur when a borrower shares one or more assets or liabilities with another borrower. If we list the jointly owned asset or liability under each borrower, then when we calculate the total assets we could inadvertently count an asset or liability more than once. In this instance, it is better method to place all of the assets into a single list, and place liabilities into a single list. Then you can add a data attribute to identify which borrower(s) the asset or liability belongs to. This method is used within the MISMO Version 2 DTDs for organizing asset and liability data and is called **pointing**. It is called a pointing relationship, because each asset and liability “points to” the borrower or borrowers it is related to.

XML provides a simple mechanism that allows a container element to “point to” another container element. This mechanism is comprised of the ID attributes and IDREFS attributes discussed earlier in this guide. In simplest terms:

An IDREF attribute “refers to” or “points to” an element with an ID attribute having the same value.

An IDREFS attribute “points to” one, or more than one element, each having an ID attribute equal to one of the IDREFS attribute values.
To demonstrate how pointing relationships work, the following example shows some sample data provided on a fictitious loan application. The borrowers are a married couple – John and Jane Doe, plus John’s father – Henry Doe. John and Jane have assets and liabilities, some joint and some individual. Henry Doe also has assets and liabilities. The borrower’s SSN is used as an identifier in the example, but a number (Bor001) or name (John_Doe) could be used just as easily.

Example Illustrating “Pointing” Relationships

```xml
<LOAN_APPLICATION>
  <BORROWER BorrowerID="ssn111111111" _UnparsedName="JOHN DOE"/>
  <BORROWER BorrowerID="ssn222222222" _UnparsedName="JANE DOE"/>
  <BORROWER BorrowerID="ssn333333333" _UnparsedName="HENRY DOE"/>
  <ASSET BorrowerID="ssn111111111 ssn222222222" _Type="RealEstateOwned" _CashOrMarketValueAmount="152000"/>
  <ASSET BorrowerID="ssn111111111" _Type="Automobile" _CashOrMarketValueAmount="25000"/>
  <ASSET BorrowerID="ssn333333333" _Type="RealEstateOwned" _CashOrMarketValueAmount="295000"/>
  <LIABILITY BorrowerID="ssn111111111" _HolderName="American Express" _UnpaidBalanceAmount="830"/>
  <LIABILITY BorrowerID="ssn111111111 ssn222222222" _HolderName="Macys West" _MonthlyPaymentAmount="120" _UnpaidBalanceAmount="4725"/>
  <LIABILITY BorrowerID="ssn111111111 ssn222222222" _HolderName="Discover Financial Services" _MonthlyPaymentAmount="90" _UnpaidBalanceAmount="3195"/>
  <LIABILITY BorrowerID="ssn333333333" _HolderName="ANB AMRO VISA" _MonthlyPaymentAmount="150" _UnpaidBalanceAmount="400"/>
</LOAN_APPLICATION>
```

The organization of the XML data is very similar to how the assets and liabilities are organized on the “URLA 1003/65” loan application form. To total the assets listed for the loan, just add up all of the “Cash Or Market Value Amounts”. To total the assets of only John and Jane Doe or only Henry Doe, just total up the amounts of assets having the desired BorrowerID attribute values. The same methods would apply to liabilities.

**NOTE:** In the actual MISMO AUS 2.1 LOAN_APPLICATION structure, the BORROWER elements appear after the ASSET and LIABILITY elements.
Chapter 3: Security Principles

The intent of this Chapter of the MISMO XML Implementation Guide – General Information – Version x is to provide some education and some general guidance with respect to security principles.

Armed with knowledge, trading partners can then select their own security and risk mitigation tools and solutions to meet both their specific business needs and the needs of their trading partners.

General Security Principles

As the name implies, there are some generally accepted principles that are commonly used when addressing security concerns regarding the movement of data between various parties to a business transaction.

Authentication

Authentication is the process of establishing confidence in user identities.\(^1\)

Trading partners must perform authentication to establish a degree of confidence in the identity with who they are in communication. This can be driven by legislative, regulatory, intellectual propriety, financial and general privacy protection requirements. Regardless, trading partners must deploy a reliable mechanism to assert their identity as well as validate their partner’s identity.

Within this chapter, the representation of identity is referred to as a “token”. Tokens can take many forms: login string, passwords, account numbers or digital certificates. Some forms for tokens are more secure than others. Account numbers or passwords are usually clear-text strings and contain no embedded protection to provide privacy (see additional comments throughout this Guide related to clear text logins and passwords); hence, confidentiality must be applied when they are used. These recommendations attempt to address protection requirements for non-secure tokens.

Confidentiality

Confidentiality is reserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. [44 U.S.C., SEC. 3542]\(^2\)

Confidentiality and privacy are often used synonymously. There are several methods to achieve privacy for information from strong access controls to encryption. Robust authentication can be used to implement identity based

---

\(^1\) NIST SP 800-63

\(^2\) NIST SP 800-199
access controls. However, for MISMO transactions (data-in-motion) between trading partners, encryption is the preferred solution. As a general rule, private or sensitive data-in-motion should be protected, regardless of internal (enterprise) or external transports. Restated, any sensitive data that is transferred over public networks (e.g., internal eMail/IM) or stored in portable storage media (e.g., laptops, flash/USB drives) should be encrypted to protect it from unauthorized access or disclosure.

**Integrity**

Integrity involves guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity. [44 U.S.C., SEC. 3542]³

Integrity comprises timely, accurate, complete, and consistent data. The information must not be manipulated in any way, either through electronic errors or human intention. The use of hashing functions and digital signatures are very common in many system applications to provide data integrity services. A strong hashing function ensures that data modification does not go undetected. And by then digitally signing the hash value, one can ensure that the hash can be trusted.

**Non-repudiation**

Non-repudiation can provide various levels of assurance that the sender of information is provided with proof of delivery and the recipient is provided with proof of the sender’s identity, so neither can later deny having processed the information.⁴

Historically, repudiate is a legal term for the ability to deny or reject validity or authority. In the world of eCommerce, the goal of non-repudiation is to prevent repudiation of valid transactions, which is critical to the success of eCommerce. The ability to prevent an entity from denying a particular act must be supported to ensure intent. There are several examples from a simple transaction between parties to an electronic signature (eSignature).

Appropriate policies and procedures, along with the security principles of authentication and integrity are combined into a single principle. This helps to ensure the identity of the entity and integrity of the associated transaction, which provides evidence against modification. A commonly used method for non-repudiation is XML digital signature (DigSig).

---

³ NIST SP 800-199
⁴ NIST SP 800-53, Revision 1
Privileged-Based Access Control

Privileged-based access control is the enforcement of specified authorization rules based on positive identification of users and the systems or data they are permitted to access.\(^5\)

Of the five basic security principles, access control is outside the scope of this document. Access controls or authorization are based on privileges granted between trading partners and is not within the current scope of this standards body.

The accuracy, confidentiality and integrity of data are critical to the mortgage process. Without appropriate security solutions, issues such as fraud and privacy rights violations can severely hamper an organization's ability to migrate to the use of electronic data exchanges and other core eCommerce processes.

The application of these principles to MISMO’s mission is the foundation for the recommendations included in this chapter. Authentication, confidentiality, and integrity are fundamentals applied to secure transmission of information. Each of these three principles is practiced to resolve specific requirements related to access, disclosure, or modification of sensitive information. Non-repudiation is often achieved by implementing a digital signature to a document or data element.

\(^5\) [www.utmb.edu/is/security/glossary.htm](http://www.utmb.edu/is/security/glossary.htm)
General Recommendations for Securely Transporting Sensitive Mortgage Information

As with all security programs, there are trade-offs to be made between achieving regulatory compliance, and assessing legal ramifications, privacy concerns, resources, and expenses. Businesses are required by various laws to maintain their security programs and risk mitigation controls. Authentication, confidentiality and integrity solutions can be found in many different tools and solutions.

In general, all communications between trading partners should be performed through a secure transport or tunnel. A secure tunnel can be described as an encrypted connection between two end points. For example, when a consumer uses the Internet for eCommerce (e.g., on-line banking), a secure tunnel is establish between the user’s browser and the remote web server. Over time, consumers have come to recognize that the “lock” icon appears as notification to the user that a secure connection has been established. All data transmitted between the two end points (browser and server) is now encrypted. Business-to-business (B2B) transactions should also be performed through a secure tunnel using protocols such as SSL/TLS, SFTP/SSH or IPsec. If you are not sure whether your current electronic business operations utilize secure tunnels, then it is recommended that your organization research and verify this environment. If your organization has not implemented the use of secure tunnels, then it is highly recommended that your organization implement them to ensure appropriate protection of sensitive information being transported between business entities.

Other areas of concern when transporting sensitive information are electronic mail (eMail) and Instant Messaging (IM). Both technologies offer convenience and productivity to users. In addition, both technologies offer users the ability to transmit sensitive data outside the controls of an application environment. For example, many organizations deploy a server outside the internal network in a DMZ (demilitarized zone) with a secure tunnel to the remote client/system and some form of authentication. These automated processes are intended to ensure access controls, confidentiality, and protection to the internal network. Tools such as eMail and IM circumvent that infrastructure. There is no guaranty of privacy for the text or attachment(s), and authentication is based on the sender’s eMail or IM address. There are however, solutions that can be implemented such as S/MIME (Secure/Multipurpose Internet Mail Extensions), PGP, or other commercially available data encryption technologies to provide for a more secure message exchange. Businesses should establish policies around the use of eMail and IM, and if required deploy secure messaging technology and policies that incorporate the general security principles discussed.
Secure Messaging Requirements for Supporting Electronic Mortgage Processes

The MISMO Information Security Work Group (ISWG) has defined a set of requirements for securely transporting sensitive electronic mortgage information. The requirements are independent of the technology solutions. Current MISMO standards may not be able to provide support for all of the requirements, but the security requirements are expected to be made mandatory in future MISMO messaging standards.

MISMO ISWG Recommendations for Secure Messaging Requirements:

Multiple security tokens for authentication – Messaging standards must have support for multiple authentication tokens. Authentication must support User ID, account number, password, and digital certificate.

Multiple encryption and digital signature technologies – Messaging standards must have support for multiple encryption and digital signature algorithms that secure all or a part of the content being transported. Support should include commercially recommended hash/digest, symmetric, and asymmetric algorithms.

Integrity – Messaging standards must have embedded integrity support for all or a part of the content being transported.

Multiple Trusted Service Providers – Messaging standards must support separate security domains (e.g., Service Providers) providing independent authentication, confidentiality and integrity for each domain thus allowing data to flow between multiple parties within a single business transaction.

End-to-end message-level security – Messaging standards must provide support that ensures all or a part of the content being transported is adequately secured (i.e., encrypted and/or digitally signed) between two parties, including if the content is being transported through an intermediary entity(ies) (e.g., portal server).
Recommended Security Solutions

Recommended security solutions are demonstrated as use case scenarios.

There are three basic use case scenarios:
1. Trading Partner to Trading Partner Direct
2. Trading Partners through a Portal
3. Multi-services (single service requesting entity to multiple providers of services)

Each use case scenario will contain a description, business use case, security requirements, and possible security recommendations. The possible security recommendations will address each security requirement with one or more potential solutions. The list is not an inclusive list of all potential solutions; rather it is a list that is most applicable to the electronic mortgage environment.

Assumptions:

All transactions are performed using a secure tunnel.

Trading partners have pre-negotiated legal terms and conditions, and authentication tokens or privilege rights have been exchanged.

Examples selected will embed the MISMO Envelope as a header for the process area transaction. For example, both the MISMO Credit Request and Response DTDs utilize the MISMO Envelope Request/Response Group DTDs as the header for their business process area.

The simple diagrams are not intended to represent a complex network environment. The diagram associated with the “Client” appears to represent a Desktop PC. In reality it could be a server or some other device. Both “Client” and “Service Provider” environments could contain several domains with the application service in one domain and a communication server in a DMZ.
Trading Partner-to-Trading Partner Direct Scenario

This is the simplest of all use cases. There are only two parties involved in the transaction: a requesting party (client) and the responding party or Service Provider (SP). The transaction is a single document or MISMO DTD. For example, lender ABC Mortgage issues a Credit Request DTD to XYZ Credit Bureau. XYZ Credit Bureau will respond with the corresponding MISMO Credit Response DTD.

Security Requirements

1. Client authenticates Service Provider (SP). SP presents token to Client, and Client validates token. Client authentication of the Service Provider (server) will reduce pharming scams and is useful for synchronous transactions.
2. SP authenticates Client. Client presents token to SP, and SP validates token.
3. Protect any sensitive data (e.g., PI) transmitted between Client and SP. Many MISMO transactions (DTDs) contain personal information (PI). These sensitive elements should be identified to ensure appropriate protection is being applied. At a minimum, a secure tunnel is to be used to pass both the request and response transactions.
4. Ensure integrity of data being transmitted between Client and SP. A secure and mutually authenticated tunnel will provide some implicit level of integrity.

Possible Security Solutions

Authentication Solutions

- SSL/TLS – Client and SP mutual authentication using digital certificates (i.e., SSL server certificate for SP and client digital certificate for client).
- SSL/TLS – Client and SP mutual authentication using a combination of digital certificates and username/passwords. Client authenticates SP by validating SP’s SSL server certificate; SP authenticates client by validating client’s username/password (obtained from HTTP header).
- **Secure FTP** – Client and SP mutual authentication using a password or certificate based secured FTP connection.

- **VPN** over frame reply/dedicated lines using *PKI digital certificates* or Share Secrets for mutual authentication.

- Digitally signed email using S/MIME or PGP to provide mutual authentication of sender and recipient.

### Confidentiality Solutions

- SSL/TLS using minimum 1024 bit *RSA public keys* and 128 bit *3DES* or *AES*. SSL/TLS provides for an encrypted tunnel to transport plaintext data.

- VPN using SSL/TLS or IPSec. VPN provides for an encrypted tunnel to transport plaintext data.

- Encrypted email using S/MIME or PGP and minimum 1024 bit RSA public keys and 128 bit 3DES or AES.

- **XML encryption** to provide additional confidentiality on specific data elements within an XML document being transported through a secure tunnel.

### Integrity Solutions

- SSL/TLS – Implicit integrity is achieved through establishment of a mutually authenticated SSL/TLS connection.

- VPN using SSL or IPSec – Implicit integrity is achieved through establishment of a mutually authenticated VPN connection.

- S/MIME email – A digitally signed email also provides data integrity services.

- XML signature – Additional data integrity can be provided on specific XML data elements within an XML document being transported through a secure tunnel.
Trading Partners through a Portal Scenario

In this scenario, the client interacts with a SP via an intermediary entity such as a Portal. There are two variations on this scenario as described below. In the first variation, the Portal acts as a pass through for the requests and responses that are exchanged between the client and the SP. In the second variation, the Portal redirects the client’s request to the appropriate SP.

Trading Partners through Portal - Scenario 1

Trading Partners through Portal - Scenario 1
Security Requirements (Variation 1)

4. SP authenticates Portal. Portal presents token to SP, and SP validates token.
5. Protect any sensitive data (e.g., PI) transmitted between Client, Portal and SP. Many MISMO transactions (DTDs) contain personal information (PI). These sensitive elements should be identified to ensure appropriate protection is being applied. At a minimum, a secure tunnel is to be used to pass both the request and response transactions.
6. Ensure non-disclosure of sensitive data to unauthorized entities. For example, the Portal may not be authorized to view certain information in the request/response sequence.
7. Ensure integrity of data being transmitted between Client, Portal and SP. A secure and mutually authenticated tunnel will provide some implicit level of integrity.

Security Requirements (Variation 2):

3. Portal determines an authorized SP for the Client, and redirects Client to that SP.
4. SP authenticates Client. Client presents token to SP, and SP validates token. (Note: The Client’s token may be forwarded by the Portal to the SP.)
5. Protect any sensitive data (e.g., PI) transmitted between Client, Portal and SP.
6. Ensure non-disclosure of sensitive data to unauthorized entities. For example, the Portal may not be authorized to view certain information in the request/response sequence.
7. Ensure integrity of data being transmitted between Client, Portal and SP. A secure and mutually authenticated tunnel will provide some implicit level of integrity.
Possible Security Solutions

Authentication Solutions

- SSL/TLS – Client and Portal mutual authentication using digital certificates (i.e., SSL server certificate for Portal and client digital certificate for client).
- SSL/TLS – Client and Portal mutual authentication using a combination of digital certificates and username/passwords. Client authenticates Portal by validating Portal’s SSL server certificate; Portal authenticates client by validating client’s username/password (obtained from HTTP header).
- SSL/TLS – SP and Portal mutual authentication using digital certificates (i.e., SSL server certificates for Portal and SP).
- SSL/TLS – SP and Portal mutual authentication using a combination of digital certificates and username/passwords. SP authenticates Portal by validating Portal’s SSL server certificate; Portal authenticates SP by validating SP’s username/password (obtained from HTTP header).

Confidentiality Solutions

- SSL/TLS using minimum 1024 bit RSA public keys and 128 bit 3DES or AES. SSL/TLS provides for an encrypted tunnel to transport plaintext data.
- XML encryption to provide additional confidentiality on specific data elements within an XML document being transported through a secure tunnel.

Integrity Solutions

- SSL/TLS – Implicit integrity is achieved through establishment of a mutually authenticated SSL/TLS connection.
- XML signature – Additional data integrity can be provided on specific XML data elements within an XML document being transported through a secure tunnel.
Multi-Services Scenario

In this scenario, the client interacts with a multi-services provider (MSP) to receive the requested information. The client submits a single request to the MSP. The MSP then submits various requests to different SPs. In many cases the MSP uses the same document in each of its requests to the SPs to minimize the number of different requests that have to be created by the MSP. The SPs individually respond back to the MSP, which then creates and sends a single response back to the client.

Multi-Services - Scenario 1

Security Requirements

1. Client authenticates MSP. MSP presents token to Client, and Client validates token.
2. MSP authenticates Client. Client presents token to MSP, and MSP validates token.
3. MSP authenticates SP1. SP1 presents token to MSP, and MSP validates token.
4. SP1 authenticates MSP. MSP presents token to SP1, and SP1 validates token.
5. MSP authenticates SPn. SPn presents token to MSP, and MSP validates token.
6. SPn authenticates MSP. MSP presents token to SPn, and SPn validates token.
7. Protect any sensitive data (e.g., PI) transmitted between Client, MSP and SPs. Many MISMO transactions (DTDs) contain personal information (PI). These sensitive elements should be identified to ensure appropriate protection is being applied. At a minimum, a secure tunnel is to be used to pass both the request and response transactions.
8. Ensure non-disclosure of sensitive data to unauthorized entities. For example, a particular SP may not be authorized to view certain information in the request/response sequence, especially if the MSP is using a single XML document for all SP request/response sequences.
9. Ensure integrity of data being transmitted between Client, MSP and SPs. A secure and mutually authenticated tunnel will provide some implicit level of integrity.

Possible Security Solutions

Authentication Solutions

- SSL/TLS – Client and MSP mutual authentication using digital certificates (i.e., SSL server certificate for MSP and client digital certificate for client).
- SSL/TLS – Client and MSP mutual authentication using a combination of digital certificates and username/passwords. Client authenticates MSP by validating MSP’s SSL server certificate; MSP authenticates client by validating client’s username/password (obtained from HTTP header).
- SSL/TLS – MSP and SP mutual authentication using digital certificates (i.e., SSL server certificates for MSP and SP).
- SSL/TLS – MSP and SP mutual authentication using a combination of digital certificates and username/passwords. SP authenticates MSP by validating MSP’s SSL server certificate; MSP authenticates SP by validating SP’s username/password (obtained from HTTP header).
- Secure FTP – MSP and SP mutual authentication using a password or certificate based secured FTP connection.
- VPN over frame reply/dedicated lines using PKI digital certificates or Share Secrets for mutual authentication between MSP and SPs.
- Digitally signed email using S/MIME or PGP to provide mutual authentication between MSP and SPs.

Confidentiality Solutions

- SSL/TLS using minimum 1024 bit RSA public keys and 128 bit 3DES or AES. SSL/TLS provides for an encrypted tunnel to transport plaintext data.
- VPN using SSL/TLS or IPSec. VPN provides for an encrypted tunnel to transport plaintext data.
- Encrypted email using S/MIME or PGP and minimum 1024 bit RSA public keys and 128 bit 3DES or AES.
• XML encryption to provide additional confidentiality on specific data elements within an XML document being transported through a secure tunnel.

Integrity Solutions

• SSL/TLS – Implicit integrity is achieved through establishment of a mutually authenticated SSL/TLS connection.

• VPN using SSL or IPSec – Implicit integrity is achieved through establishment of a mutually authenticated VPN connection.

• S/MIME email – A digitally signed email also provides data integrity services.

• XML signature – Additional data integrity can be provided on specific XML data elements within an XML document being transported through a secure tunnel.
## Security Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DES</td>
<td>Also referred to as <em>triple DES</em>, a mode of the DES encryption algorithm that encrypts data three times. Three 64-bit keys are used, instead of one, for an overall key length of 192 bits (the first encryption is encrypted with the second key, and the resulting cipher text is again encrypted with a third key).</td>
</tr>
<tr>
<td>AES</td>
<td>Short for <em>Advanced Encryption Standard</em>, a symmetric 128-bit block data encryption algorithm.</td>
</tr>
<tr>
<td>Asymmetric encryption</td>
<td>In deploying asymmetric data encryption, the use of public and private key pairs to encrypt and subsequently decrypt the data is required. Use of these pairs of keys can provide for both confidentiality and authentication.</td>
</tr>
<tr>
<td>Digital signature</td>
<td>A digital code that can be attached to an electronically transmitted message that uniquely identifies the sender. Like a written signature, the purpose of a digital signature is to guarantee that the individual sending the message really is who he or she claims to be. Digital signatures are especially important for electronic commerce and are a key component of most authentication schemes.</td>
</tr>
<tr>
<td>DMZ</td>
<td>Short for <em>demilitarized zone</em>, a computer or small subnetwork that sits between a trusted internal network, such as a corporate private LAN, and an untrusted external network, such as the public Internet. Typically, the DMZ contains devices accessible to Internet traffic, such as Web (HTTP) servers, FTP servers, SMTP (e-mail) servers and DNS servers. The term comes from military use, meaning a buffer area between two enemies.</td>
</tr>
</tbody>
</table>
DTD Short for *document type definition*. A DTD states what tags and attributes are used to describe content in an SGML, XML or HTML document, where each tag is allowed, and which tags can appear within other tags. For example, in a DTD one could say that LIST tags can contain ITEM tags, but ITEM tags cannot contain LIST tags. In some editors, when authors are inputting information, they can place tags only where the DTD allows. This ensures that all the documentation is formatted the same way. Applications will use a document's DTD to properly read and display a document's contents. Changes in the format of the document can be easily made by modifying the DTD.

Encryption The translation of data into a secret code. Encryption is the most effective way to achieve data security. To read an encrypted file, you must have access to a secret key or password that enables you to decrypt it. Unencrypted data is called *plain text*; encrypted data is referred to as *cipher text*.

There are two main types of encryption: asymmetric encryption (also called public-key encryption) and symmetric encryption.

Frame Relay An efficient data transmission technique used to send digital information quickly and cheaply in a relay of frames to one or many destinations from one or many end-points. Network providers commonly implement frame relay for voice and data as an encapsulation technique, used between local area networks (LANs) over a wide area network (WAN). Each end-user gets a private line (or leased line) to a frame-relay node. The frame-relay network handles the transmission over a frequently-changing path transparent to all end-users.
Hash/digest

Producing *hash values* for accessing data or for security. A hash value (or simply *hash*), also called a *message digest*, is a number generated from a string of text. The hash is generated by a formula in such a way that it is extremely unlikely that some other text will produce the same hash value.

Hashes play a role in security systems where they're used to ensure that transmitted messages have not been tampered with. The sender generates a hash of the message, encrypts it, and sends it with the message itself. The recipient then decrypts both the message and the hash, produces another hash from the received message, and compares the two hashes. If they're the same, there is a very high probability that the message was transmitted intact.

HTTP

Short for *HyperText Transfer Protocol*, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.

IPsec

Short for *IP Security*, a set of protocols developed by the IETF to support secure exchange of packets at the IP layer. IPsec has been deployed widely to implement Virtual Private Networks (VPNs).

IPsec supports two encryption modes: Transport and Tunnel. Transport mode encrypts only the data portion (*payload*) of each packet, but leaves the header untouched. The more secure Tunnel mode encrypts both the header and the payload. On the receiving side, an IPsec-compliant device decrypts each packet.
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PGP</strong></td>
<td>Pretty Good Privacy, abbreviated as <em>PGP</em>, a technique developed by Philip Zimmerman for encrypting messages. PGP is one of the most common ways to protect messages on the Internet because it is effective, easy to use, and free. PGP is based on the public-key method, which uses two keys -- one is a public key that you disseminate to anyone from whom you want to receive a message. The other is a private key that you use to decrypt messages that you receive.</td>
</tr>
<tr>
<td><strong>Pharming</strong></td>
<td>Pharming seeks to obtain personal or private (usually financial related) information through domain spoofing. Pharming 'poisons' a DNS server by infusing false information into it, resulting in a user's request being redirected elsewhere. Your browser, however will show you are at the correct Web site, which makes pharming a bit more serious and more difficult to detect.</td>
</tr>
<tr>
<td><strong>PKI digital certificate</strong></td>
<td>Short for <em>public key infrastructure</em>, a system of digital certificates, Certificate Authorities, and other registration authorities that verify and authenticate the validity of each party involved in an Internet transaction. PKIs are currently evolving and there is no single PKI nor even a single agreed-upon standard for setting up a PKI.</td>
</tr>
<tr>
<td><strong>RSA public keys</strong></td>
<td>An public-key encryption technology developed by RSA Data Security, Inc. The acronym stands for Rivest, Shamir, and Adelman, the inventors of the technique. The RSA algorithm is based on the fact that there is no efficient way to factor very large numbers. Deducing an RSA key, therefore, requires an extraordinary amount of computer processing power and time.</td>
</tr>
</tbody>
</table>
S/MIME

Short for *Multipurpose Internet Mail Extensions*, a specification for formatting non-ASCII messages so that they can be sent over the Internet. Many e-mail clients now support MIME, which enables them to send and receive graphics, audio, and video files via the Internet mail system. In addition, MIME supports messages in character sets other than ASCII.

There are many predefined MIME types, such as GIF graphics files and PostScript files. It is also possible to define your own MIME types.

In addition to e-mail applications, Web browsers also support various MIME types. This enables the browser to display or output files that are not in HTML format.

MIME was defined in 1992 by the Internet Engineering Task Force (IETF). A new version, called S/MIME, supports encrypted messages.

Secure FTP

See SFTP/SSH

Security domain

The subset of data required by a party to fulfill their respective portion of the request.

SFTP/SSH

In computing, *Secure Shell* or SSH is a set of standards and an associated network protocol that allows establishing a secure channel between a local and a remote computer. It uses public-key cryptography to authenticate the remote computer and (optionally) to allow the remote computer to authenticate the user. SSH provides confidentiality and integrity of data exchanged between the two computers using encryption and message authentication codes.

FTP over SSH is sometimes referred to as *secure FTP or SFTP.*
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSL</strong></td>
<td>SSL, Short for <em>Secure Sockets Layer</em>, a protocol developed by Netscape for transmitting private documents via the Internet. SSL uses a cryptographic system that uses public and private key pairs to encrypt data. Both Netscape Navigator and Internet Explorer support SSL, and many Web sites use the protocol to obtain confidential user information, such as credit card numbers. By convention, URLs that require an SSL connection start with <code>https:</code> instead of <code>http:</code>.</td>
</tr>
<tr>
<td><strong>Symmetric encryption</strong></td>
<td>In deploying data encryption, the use of a key to de-crypt the data is required. When utilizing a symmetric algorithm, both the sender and receiver of the data share a common key.</td>
</tr>
</tbody>
</table>
| **TLS** | Short for *Transport Layer Security*, a protocol that guarantees privacy and data integrity between client/server applications communicating over the Internet. The TLS protocol is made up of two layers:  
  - The *TLS Record Protocol* -- layered on top of a reliable transport protocol, such as TCP, it ensures that the connection is private by using symmetric data encryption and it ensures that the connection is reliable. The TLS Record Protocol also is used for encapsulation of higher-level protocols, such as the TLS Handshake Protocol.  
  - The *TLS Handshake Protocol* -- allows authentication between the server and client and the negotiation of an encryption algorithm and cryptographic keys before the application protocol transmits or receives any data. |
| **VPN** | Short for *virtual private network*, a network that is constructed by using public wires to connect nodes. For example, there are a number of systems that enable you to create networks using the Internet as the medium for transporting data. These systems use encryption and other security mechanisms to ensure that only authorized users can access the network and that the data cannot be intercepted. |
XML encryption  

An encryption method for XML data that offers the ability to selectively encrypt certain data elements, while leaving the remainder in clear text.
Chapter 4: The MISMO Transaction Envelope

Since the MISMO Version 2 standards are oriented towards transactions, MISMO has developed a set of Transaction Envelope DTDs that can be used to wrap the transaction data. The DTDs contain basic information common to most transactions – elements that identify the requesting party, receiving party, responding party and other reference data that is commonly exchanged between business partners.

The use of the MISMO Transaction Envelope is optional. Some business partners may prefer to use other methodologies such as SOAP (Simple Object Access Protocol) to wrap their transactions.

This chapter of the implementation guide covers both the Transaction Request Envelope used in requesting services or products, and the Transaction Response envelope used by vendors delivering services or products. The drawing below shows a simple request transaction between a lender and a service vendor, followed by the response transaction from the service vendor to the lender.

Request and Response Transactions

[Diagram showing a simple request transaction between a lender and a service vendor, followed by the response transaction from the service vendor to the lender.]
The Request Envelope

The MISMO Architecture provides a variety of request elements that allow for a flexible request structure that can be adapted to a variety of business-to-business scenarios.

Sample REQUEST with Requesting and Receiving Parties

```xml
<REQUEST_GROUP MISMOVersionID="2.1">
  <REQUESTING_PARTY _Name="MFC Mortgage" _StreetAddress="21650 Oxnard Street" _City="Woodland Hills" _State="CA" _PostalCode="91364"/>
  <RECEIVING_PARTY _Name="ABC Services" _StreetAddress="7200 Peachtree Street" _City="Atlanta" _State="GA" _PostalCode="30010"/>
  <REQUEST RequestDatetime="2002-01-08T17:19:01" InternalAccountIdentifier="ABC-0732">
    <KEY _Name="MFC Transaction ID" _Value="702430023"/>
    <KEY _Name="MFC Portfolio ID" _Value="MFC2002-0030"/>
    <REQUEST_DATA>
      ... BODY OF REQUEST GOES HERE ...
    </REQUEST_DATA>
    </REQUEST>
  </REQUEST_GROUP>
```

Requesting Party Element

The Requesting Party is the ultimate customer of the service needed for the mortgage transaction. It is usually the entity that is ultimately billed for the service. MFC Mortgage is the requesting party used in the request samples in this section of the guide.

Preferred Response Element

Normally, the response format for B2B (Business To Business) transactions is defined between the trading partners beforehand and is the same for all transactions. However, the Preferred Response element of the Requesting Party can be used to override the pre-defined response format settings on a transaction-by-transaction basis. This element’s attributes allow the Requesting Party to specify the format and destination of the response transactions. They can also specify whether a response format is to be separate from the XML data or embedded within it.
The **Preferred Response Format** attribute has the options: PCL, PDF, Text, or Other. The **Preferred Response Method** can be Fax, File, FTP, HTTP, HTTPS, Message Queue, SMTP, VAN, etc. A **Preferred Response Destination** attribute specifies a value appropriate for the method chosen. For example, its value could be a Fax phone number, file name, email address, a URL, etc.

The **Version 2.1.1** Request envelope has an optional **Use Embedded File Indicator** attribute that allows you to specify whether or not a response format should be stored inside the **Embedded File** element. There is also an optional **MIME Type** attribute for more precise definition of certain response formats.

The **Version 2.3** has added a **Version Identifier** attribute to the **Preferred Response** element to allow the requester to specify the version of the response file they would like to receive. For example, attributes set as _Format_="XML" and _VersionIdentifier_="MISMO 2.1" would indicate that the requester expects to receive a MISMO Version 2.1 response file.

**NOTE:** Check with the service provider to verify whether the Preferred Response element is supported and which Formats, Methods, and MIME Types are recognized.

**Sample Requesting Party with Preferred Response**

```xml
<REQUEST_GROUP MISMOVersionID="2.1.1">
  <REQUESTING_PARTY
    _Name="MFC Mortgage"
    _StreetAddress="21650 Oxnard Street"
    _City="Woodland Hills" _State="CA" _PostalCode="91364">
    <PREFERRED_RESPONSE
      _Format="PDF"
      _Method="File"
      _Destination="70240023.pdf"
      _UseEmbeddedFileIndicator="Y"
      MIMEType="application/pdf"/>
  </REQUESTING_PARTY>
  <PREFERRED_RESPONSE
    _Format="Text"
    _Method="SMTP"
    _Destination="DSmith@MFC.com"
    _UseEmbeddedFileIndicator="N"
    MIMEType="multipart/encrypted"/>
</REQUEST_GROUP>
```

NEW: Ver 2.1.1
Receiving Party Element

The Receiving Party simply identifies the party that is on the receiving end of the request transaction. Normally it is a provider of a service or product. **ABC Services** is the service provider in these samples.

Submitting Party Element

The Submitting Party is an individual or organization that makes a request for a service on behalf of the Requesting Party. For example, when a mortgage service broker makes a request for a service on behalf of the end user of the service, they would identify themselves in the request transaction as the Submitting Party, and identify the end user as the Requesting Party.

In the sample file below, the service broker – **AAA Mortgage Trader** (Submitting Party) is generating a request for a mortgage service on behalf of **MFC Mortgage** (Requesting Party). AAA Mortgage Trader sends the request to **ABC Services** (Receiving Party), who processes the request. ABC Services will bill MFC Mortgage for the service.

Sample REQUEST with Requesting, Receiving and Submitting Parties

```xml
<REQUEST_GROUP MISMOVersionID="2.1">

<REQUESTING_PARTY _Name="MFC Mortgage" _StreetAddress="21650 Oxnard Street" _City="Woodland Hills" _State="CA" _PostalCode="91364"/>

<RECEIVING_PARTY _Name="ABC Services" _StreetAddress="7200 Peachtree Street" _City="Atlanta" _State="GA" _PostalCode="30010"/>

<SUBMITTING_PARTY _Name="AAA Mortgage Trader" _StreetAddress="324 Redstone Ave" _City="Denver" _State="CO" _PostalCode="81351"/>

<REQUEST RequestDateTime="2002-01-08T17:19:01" InternalAccountIdentifier="ABC-0732">

  <KEY _Name="AAA Transaction #" _Value="23029497729"/>
  <KEY _Name="MFC Transaction ID" _Value="702430023"/>
  <KEY _Name="MFC Portfolio ID" _Value="MFC2002-0030"/>

  <REQUEST_DATA>
  ... BODY OF REQUEST GOES HERE ...
  </REQUEST_DATA>

</REQUEST>

</REQUEST_GROUP>
```
The diagram on the next page shows an original request transaction from a lender to an Internet portal (Service Broker). In the first request transaction, the lender is the **Requesting Party** and the Internet portal is the **Receiving Party**.

In the second request transaction from the service broker to the service provider, the lender is still the **Requesting Party**, but the service broker is the **Submitting Party** and the service provider is the **Receiving Party** (as shown in the sample above).

In the response transaction, the service provider is the **Responding Party** and the lender is the **Respond To Party**.

For Version 2.3, several enhancements were made to the **Submitting Party** data structure. First, this element can now appear more than once within a **Request Group**. In addition a **Sequence Identifier** attribute has been added. These two changes now allow more than one submitting party to be listed in a request transaction, which may be needed in situations where a request is being handled and passed through by more than one web portal. A **Login Account Identifier** attribute and a **Login Account Password** attribute have also been added for situations where authentication information needs to be passed through to another party in the transaction.

**Security issues regarding login authentication are discussed later in this chapter.**
Request and Response Transactions with Submitting Party

Lender

MFC MORTGAGE

FROM: <Requesting Party> MFC Mortgage
TO: <Receiving Party> AAA Mortgage Trader

Request Envelope

Response Envelope

AAA Mortgage Trader

FROM: <Responding Party> ABC Services
TO: <Respond To Party> MFC Mortgage

Service Broker

ABC SERVICES

FROM: <Requesting Party> MFC Mortgage
<Submitting Party> AAA Mortgage Trader
TO: <Receiving Party> ABC Services

Service Vendor
Request Element

Each MISMO 2.x Request Envelope can contain one or more Request Elements. The Request Element can contain one or more Request Data Elements that holds the actual details of the particular type of request (i.e. Appraisal, Credit, Flood, Mortgage Insurance, Title, etc.).

The Request Element contains several optional attributes:

- **RequestDatetime** – when the Requesting Party made the request.
- **InternalAccountIdentifier** – the account number assigned to the Requesting Party by the Receiving Party for billing purposes.
- **LoginAccountIdentifier** – required by some Receiving Party vendors to gain access to their system. See next page for more information regarding security issues related to login authentication data.
- **LoginAccountPassword** – may also be required by some vendors.

One additional feature of the Request Element is the Key Element. This element allows the passing of one or more reference or identification “keys” in the request transaction to the service provider. The service provider will then return the same “keys” in the response transaction.

The use of the KEY element in this manner allows the requester to send reference data in the request, which will make it easy for the requester to match the response transaction to the original request. Examples of data that could be included in the KEY element could include Loan IDs, Request IDs, Mailbox Numbers, or any other data that would allow the requester to match a response to a request. Each KEY element has a **Name** and **Value** attribute. The Key Name describes the Key Value.

Sample REQUEST with Reference KEY Elements

```xml
<REQUEST_GROUP MISMOVersionID="2.1">
  <REQUESTING_PARTY
    _Name="MFC Mortgage"
    _StreetAddress="21650 Oxnard Street"
    _City="Woodland Hills" _State="CA" _PostalCode="91364"/>
  <RECEIVING_PARTY
    _Name="ABC Services"
    _StreetAddress="7200 Peachtree Street"
    _City="Atlanta" _State="GA" _PostalCode="30010"/>
  <REQUEST RequestDatetime="2002-01-08T17:19:01"
            InternalAccountIdentifier="ABC-0732">
    <KEY _Name="MFC Transaction ID" _Value="702430023"/>
    <KEY _Name="MFC Portfolio ID" _Value="MFC2002-0030"/>
    <REQUEST_DATA>
      ... BODY OF REQUEST GOES HERE ...
    </REQUEST_DATA>
  </REQUEST>
</REQUEST_GROUP>
```
Security Issues with Login Authentication

Even if Login Account Identifier and Login Account Password data is encrypted while it is being transmitted, when it reaches its destination it will generally be unencrypted at that point for processing and/or storage. At this point the authentication data is potentially vulnerable and could be misused. Therefore, the practice of passing login authentication data in this manner should be discouraged and should only be used in limited situations where the security of the login authentication data can be guaranteed.

A preferred alternative to passing of login data is to use digital certificates which can be stored and transmitted as encrypted data and passed to the receiving party where it can be used to authenticate the sender. More information about digital certificates is available in the Document Library of the Security Identity Services Accreditation Corporation (SISAC). This organization was setup by the Mortgage Bankers Association to develop and maintain industry standards for security identity credentials to be used in electronic mortgage transactions. More information about SISAC and their Document Library can be found on their web site at www.sisac.org.

Request Envelope Usage Scenarios

The MISMO Request envelope structure was designed to be used for many types of mortgage services. The REQUEST_GROUP structure can be used for requesting Credit, Flood, Mortgage Insurance, Title, and other services as well as Automated Underwriting. The envelope structure supports multiple loans (one per REQUEST element), and multiple service requests per loan (one per REQUEST_DATA element).

BUNDLED SERVICES REQUEST FOR A SINGLE LOAN

In this example, there is a request for Flood and Title for the same loan from a bundled services provider. Note that the all of the requests for the loan are contained in a single REQUEST element, and each service request is contained in its own REQUEST_DATA element.

```
<REQUEST_GROUP>
  <REQUESTING_PARTY _Name="ABC Bank"/>
  <RECEIVING_PARTY _Name="XYZ Bundled Services"/>
  <REQUEST>
    <KEY _Name="Loan File ID" _Value="2005-0001"/>
    <REQUEST_DATA>
      <FLOOD_REQUEST FloodRequestID="FR0001"/>
    </REQUEST_DATA>
    <REQUEST_DATA>
      <TITLE_REQUEST TitleRequestID="TR0001"/>
    </REQUEST_DATA>
  </REQUEST>
</REQUEST_GROUP>
```
BUNDLED SERVICES REQUEST FOR MULTIPLE LOANS

This example is similar to the previous one except there are two loans, each with a request for Flood and Title. Note again that all of the requests for each loan are contained in a single REQUEST element, and each service request is contained in its own REQUEST_DATA element.

```xml
<REQUEST_GROUP>
  <REQUESTING_PARTY _Name="ABC Bank"/>
  <RECEIVING_PARTY _Name="XYZ Bundled Services"/>
  <REQUEST>
    <KEY _Name="Loan File ID" _Value="2005-0002"/>
    <REQUEST_DATA>
      <FLOOD_REQUEST FloodRequestID="FR0002"/>
    </REQUEST_DATA>
    <REQUEST_DATA>
      <TITLE_REQUEST TitleRequestID="TR0002"/>
    </REQUEST_DATA>
  </REQUEST>
  <REQUEST>
    <KEY _Name="Loan File ID" _Value="2005-0003"/>
    <REQUEST_DATA>
      <FLOOD_REQUEST FloodRequestID="FR0003"/>
    </REQUEST_DATA>
    <REQUEST_DATA>
      <TITLE_REQUEST TitleRequestID="TR0003"/>
    </REQUEST_DATA>
  </REQUEST>
</REQUEST_GROUP>
```
AUTOMATED UNDERWRITING REQUESTS FOR MULTIPLE LOANS

The Automated Underwriting LOAN APPLICATION structure can be embedded within the MISMO envelope request structure, so that multiple AUS requests can be included in a single transmission. In this example, there is one loan for each REQUEST element.

```xml
<REQUEST_GROUP>
  <REQUESTING_PARTY _Name="ABC Bank"/>
  <RECEIVING_PARTY _Name="GSE Loan Mine"/>
  <REQUEST>
    <KEY _Name="Loan File ID" _Value="2005-0011"/>
    <REQUEST_DATA>
      <LOAN_APPLICATION>
        <BORROWER BorrowerID="id20395092360927709276" _FirstName="John" _LastName="Doe"/>
        <BORROWER BorrowerID="id20973829859872938577" _FirstName="Jane" _LastName="Doe"/>
      </LOAN_APPLICATION>
    </REQUEST_DATA>
  </REQUEST>
  <REQUEST>
    <KEY _Name="Loan File ID" _Value="2005-0012"/>
    <REQUEST_DATA>
      <LOAN_APPLICATION>
        <BORROWER BorrowerID="id02970919750977198846" _FirstName="Jon" _LastName="Consumer"/>
      </LOAN_APPLICATION>
    </REQUEST_DATA>
  </REQUEST>
  <REQUEST>
    <KEY _Name="Loan File ID" _Value="2005-0013"/>
    <REQUEST_DATA>
      <LOAN_APPLICATION>
        <BORROWER BorrowerID="id029836098670772272279" _FirstName="Sam" _LastName="Smith"/>
        <BORROWER BorrowerID="id9957698379625662666" _FirstName="Betty" _LastName="Smith"/>
        <BORROWER BorrowerID="id8295802937566620077" _FirstName="Gary" _LastName="Jones"/>
      </LOAN_APPLICATION>
    </REQUEST_DATA>
  </REQUEST>
</REQUEST_GROUP>
```
The Response Envelope

This section describes the various elements used in the RESPONSE element of a response transaction. In the response, the party roles names are changed. The request’s Requesting Party becomes the Respond To Party. The request’s Receiving Party becomes the Responding Party.

Responding Party Element

The Responding Party identifies the entity that prepared the response transaction – usually the service vendor. In the request transaction, they were the Receiving Party.

Respond To Party Element

In a response transaction, the Respond To Party is the entity receiving the response transaction. In the request transaction, they were the Requesting Party.

Sample RESPONSE with Responding and Respond To Parties

```xml
<RESPONSE_GROUP MISMOVersionID="2.1">
  <RESPONDING_PARTY _Name="ABC Services" _StreetAddress="7200 Peachtree Street" _City="Atlanta" _State="GA" _PostalCode="30010"/>
  <RESPOND_TO_PARTY _Name="MFC Mortgage" _StreetAddress="21650 Oxnard Street" _City="Woodland Hills" _State="CA" _PostalCode="91364"/>
  <RESPONSE RequestDateTime="2002-01-08T17:19:12" InternalAccountIdentifier="ABC-0732">
    <KEY _Name="MFC Transaction ID" _Value="702430023"/>
    <KEY _Name="MFC Portfolio ID" _Value="MFC2002-0030"/>
    <RESPONSE_DATA>
      ... BODY OF RESPONSE GOES HERE ...
    </RESPONSE_DATA>
  </RESPONSE>
</RESPONSE_GROUP>
```
Response Element

Each MISMO 2.1 Response Envelope can contain one or more Response elements. The Response Element can contain one or more Response Data elements that hold the actual details of the particular type of response (i.e. Appraisal, Credit, Flood, Mortgage Insurance, Title, etc.). The Response Element also can contain a Status element, which can be used for reporting error or status messages about the service or product requested.

The Response Element contains several optional attributes:

- **ResponseDateTime** – when the Responding Party created the response transaction.
- **InternalAccountIdentifier** – the account number from the request transaction that was assigned to the Requesting Party by the Receiving Party for billing purposes.
- **LoginAccountIdentifier** – required by some Receiving Party vendors to gain access to their system. See the “Security Issues with Login Authentication” section earlier in this chapter for more information.
- **LoginAccountPassword** – may also be required along with the Login ID above.

The service provider in the Response Transaction will return the same Key Elements that were included in the Request Transaction, in their entirety. The sample RESPONSE element below contains the same KEY elements that were initially sent by the customer in the request file.

**Sample RESPONSE with Response Element and Key Elements**

```xml
<RSPONSE_GROUP MISMOVersionID="2.1">
  <RESPONDING_PARTY>
    _Name="ABC Services"
    _StreetAddress="7200 Peachtree Street"
    _City="Atlanta" _State="GA" _PostalCode="30010"/>
  <RESPOND_TO_PARTY>
    _Name="MFC Mortgage"
    _StreetAddress="21650 Oxnard Street"
    _City="Woodland Hills" _State="CA" _PostalCode="91364"/>
  <RESPONSE ResponseDateTime="2002-01-08T17:19:12"
    InternalAccountIdentifier="ABC-0732">
    <KEY _Name="MFC Transaction ID" _Value="702430023"/>
    <KEY _Name="MFC Portfolio ID" _Value="MFC2002-0030"/>
    <RESPONSE_DATA>
      ... BODY OF RESPONSE GOES HERE ...
    </RESPONSE_DATA>
  </RESPONSE>
</RESPONSE_GROUP>
```
Response Envelope Usage Scenarios

The MISMO Response envelope structure was designed to be used for many types of mortgage services. In addition to the Credit Response, the RESPONSE_GROUP structure can be used for Flood Responses, Title Responses, Mortgage Insurance Responses, Appraisals Responses, etc. The Response envelope structure is flexible because its use can vary depending on how a business needs to use it.

Here are some possible scenarios for using the RESPONSE and RESPONSE_DATA elements:

MULTIPLE CREDIT RESPONSES PER LOAN

Some customers send multiple credit requests per loan - sometimes two, three, or more credit requests depending on the number of borrowers on the loan. When the credit responses are returned, they can be returned in the same RESPONSE element since they are all for the same loan, with each credit response in its own RESPONSE_DATA element.

```xml
<RESPONSE_GROUP>
  <RESPONDING_PARTY_Name="XYZ Credit Services"/>
  <RESPOND_TO_PARTY_Name="ABC Bank"/>
  <RESPONSE>
    <KEY_Name="Loan File ID" Value="2004-0001"/>
    <RESPONSE_DATA>
      <CREDIT_RESPONSE CreditResponseID="CreditRpt0001"/>
    </RESPONSE_DATA>
    <RESPONSE_DATA>
      <CREDIT_RESPONSE CreditResponseID="CreditRpt0002"/>
    </RESPONSE_DATA>
  </RESPONSE>
</RESPONSE_GROUP>
```
MULTIPLE LOANS PER RESPONSE_GROUP

Other customers may send large batches of credit requests for multiple loans and want them returned in a batch. Each loan might have one or more than one credit response associated with it.

```xml
<RESPONSE_GROUP>
    <RESPONDING_PARTY _Name="XYZ Credit Services"/>
    <RESPOND_TO_PARTY _Name="ABC Bank"/>
    <RESPONSE>
        <KEY _Name="Loan File ID" _Value="2004-0032"/>
        <RESPONSE_DATA>
            <CREDIT_RESPONSE CreditResponseID="CreditRpt0003"/>
        </RESPONSE_DATA>
    </RESPONSE>
    <RESPONSE>
        <KEY _Name="Loan File ID" _Value="2004-0033"/>
        <RESPONSE_DATA>
            <CREDIT_RESPONSE CreditResponseID="CreditRpt0004"/>
        </RESPONSE_DATA>
    </RESPONSE>
</RESPONSE_GROUP>
```

MULTIPLE SERVICES IN SAME RESPONSE ENVELOPE

The MISMO response can also be used by companies that provide "bundled" services to provide multiple credit responses in the same response envelope. This sample includes one RESPONSE for each loan, and one RESPONSE_DATA for each service.

```xml
<RESPONSE_GROUP>
    <RESPONDING_PARTY _Name="XYZ Bundled Services"/>
    <RESPOND_TO_PARTY _Name="ABC Bank"/>
    <RESPONSE>
        <KEY _Name="Loan File ID" _Value="2004-0051"/>
        <RESPONSE_DATA>
            <FLOOD_RESPONSE FloodResponseID="FloodRpt0001"/>
        </RESPONSE_DATA>
        <RESPONSE_DATA>
            <APPRAISAL_RESPONSE AppraisalResponseID="AppraisalRpt0001"/>
        </RESPONSE_DATA>
        <RESPONSE_DATA>
            <TITLE_RESPONSE TitleResponseID="TitleRpt0001"/>
        </RESPONSE_DATA>
    </RESPONSE>
    <RESPONSE>
        <KEY _Name="Loan File ID" _Value="2004-0052"/>
        <RESPONSE_DATA>
            <TITLE_RESPONSE TitleResponseID="TitleRpt0002"/>
        </RESPONSE_DATA>
    </RESPONSE>
</RESPONSE_GROUP>
```
XML Signature

Beginning with Version 2.3.1, both the Request Group and Response Group structures were enhanced to support the XML Signature structure, based on the W3C XML Signature Standards (http://www.w3.org/TR/xmldsig-core/). The XML Signature is used in the MISMO Transaction Envelope structures to maintain the data integrity and authenticity of the original data. It can be applied to either all or just a portion of the transaction contents.

For example, a Credit Reporting Vendor might want to apply an XML Signature to the contents of the Credit Response transaction to create a “tamper evident seal” around the credit report data that was delivered to the lender. If there was ever a need to verify that the data contents were unmodified from the original report, the XML Signature can be validated to make sure that it still matches the data contents.

The XML Signature provides two features that are important for the security of the data:

- **Authentication** – The XML Signature, when properly implemented, includes a digital certificate from the sender of the transaction document. This certificate can be validated by the receiver to verify that the document was created by the sender and not some other source. MISMO recommends the use of digital certificates that have been issued by a SISAC-approved certificate provider.

  SISAC, the Secure Identity Services Accreditation Corporation, has been established by the Mortgage Bankers Association to verify that approved certificate providers meet its strict requirements (see www.SISAC.org). A properly issued and managed digital certificate provides a high degree of certainty regarding the identity of the lender, service provider or other entity that is generating the XML Signature.

- **Tamper Evidence** – When the XML Signature is applied to a transaction document by the sender, the receiver will be able to verify whether or not the data in the document has been changed since the XML Signature was first created. The XML Signature *does not prevent* tampering of the data, it only provides a way of detecting if the data has been changed.

Even though portions of the XML Signature data are encrypted, it is important to note that the **XML Signature by itself does not encrypt the transaction data**. This is a common misconception about the XML Signature. The encryption of the transactional data can be handled in a number of ways. There is an XML Encryption standard, but at the writing of this document, MISMO has not made any recommendations about the encryption of transactional data.
Signature Element Structure

The sample below shows a simplified Response Group envelope that contains the Signature element. The Reference element’s URI attribute contains the CreditResponseID value of the CREDIT RESPONSE element, designating that this is the portion of the credit response transaction to be “signed”.

Sample RESPONSE Envelope with <Signature> Element

```
<RESPONSE_GROUP MISMOVersionID="2.3.1">
  <RESPONDING_PARTY _Name="ABC Credit Services"/>
  <RESPOND_TO_PARTY _Name="MFC Mortgage"/>
  <RESPONSE ResponseDateTime="2005-04-18T13:01:58">
    <RESPONSE_DATA>
      <CREDIT_RESPONSE MISMOVersionID="2.3.1" CreditResponseID="CResp02472">
      ... CREDIT RESPONSE DATA GOES HERE ...
      </CREDIT_RESPONSE>
    </RESPONSE_DATA>
  </RESPONSE>
  <Signature>
    <SignedInfo>
      <CanonicalizationMethod Algorithm="xml-exc-c14n#"/>
      <SignatureMethod Algorithm="xmldsig#rsa-sha1"/>
      <Reference URI="#CResp02472">
        <DigestMethod Algorithm="xmldsig#sha1"/>
        <DigestValue>SMn9GmLXyfuzGbmrLr4xU3</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>jesGhs92msJ9S8snwXhwK7</SignatureValue>
    <KeyInfo>
      <X509Data>
        ... X.509 Digital Certificate data goes here ...
      </X509Data>
    </KeyInfo>
  </Signature>
</RESPONSE_GROUP>
```

Generating the XML Signature

Although many companies may use pre-built software or a web service to generate the XML Signature block, it is helpful to have a basic understanding of the steps involved in creating the Signature element. The steps below are a simplified for demonstration purposes.

1 – Identify the area(s) to be signed

In the previous sample, the CREDIT RESPONSE element was identified, using its CreditResponseID attribute value. This value is entered into the Reference element’s URI attribute that is part of the Signature element. More than one Reference element may be included in a Signature element.
2 – Transform areas to be signed to Canonical XML

This step puts the XML data into a common normalized format, defined by the **CanonicalizationMethod**. This allows the Digest Values, which are created in the next step and in the Signature Validation process, to match each other.

3 – Calculate the Digest Value of Area(s) to be Signed

The data in the area to be signed is reduced to a unique **DigestValue**, using a method specified in **DigestMethod**.

4 – Calculate the Signature Value

An encrypted **SignatureValue** is now calculated on all of the data in the **SignedInfo** element.

5 – Store the Key Certificate

The **KeyInfo** element contains the certificate data that will be used to validate the signature.

**Sample Signature Element Structure**

```xml
<Signature>
  <SignedInfo>
    <CanonicalizationMethod Algorithm="xml-exc-c14n#"/>
    <SignatureMethod Algorithm="xmldsig#rsa-sha1"/>
    <Reference URI="#CResp02472">
      <DigestMethod Algorithm="xmldsig#sha1"/>
      <DigestValue>SMn9Gm1XgfuzGbmrLr4xU3</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>jESGhs92msJS7S8snwXhwK7</SignatureValue>
  <KeyInfo>
    <X509Data>
      ... X.509 Digital Certificate data goes here ...
    </X509Data>
  </KeyInfo>
</Signature>
```
Validating the XML Signature

When a party receives a document signed with an XML Signature, they can validate the signed portion of the document to determine whether or not any of the XML data in the signed area has been changed. The party that receives the document re-generates the **DigestValue** data values using the same steps used to generate original data values. If the original Digest Values match the new calculated values then the data has not changed.

Because validating the XML Signature involves the use of properly issued digital certificate, this process also provides reliable identification of the sender of the transaction.
Chapter 5: XML Implementation Issues

XML Reserved Characters

The topic of XML “reserved” characters is normally covered in XML tutorials, but is worth emphasizing here because ignoring these reserved characters is a frequent cause of errors in mortgage data. There are five characters that are reserved and cannot be used directly in XML element or attribute data, they must be replaced with what are called – XML Entity References. Some XML software systems will automatically do the conversion of the XML reserved characters; otherwise logic must be added to perform the conversion of the characters. The following table shows the reserved character in the first column, followed by the XML Entity Reference that it is replaced with in the second column.

<table>
<thead>
<tr>
<th>Reserved Character</th>
<th>Substitute With</th>
<th>Character Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>&amp;</td>
<td>Ampersand</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>Less Than</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>Greater Than</td>
</tr>
<tr>
<td>’</td>
<td>'</td>
<td>Apostrophe</td>
</tr>
<tr>
<td>“</td>
<td>&quot;</td>
<td>Quote</td>
</tr>
</tbody>
</table>

Processing Received XML Data

The following examples show the XML data received in a mortgage transaction being converted into “normal” data, as it would be stored in an internal database.

Example 1:

XML Data:  
>CreditorName="Lord &amp; Taylor"

Converts To:  
Lord & Taylor

Example 2:

XML Data:  
>CreditorName="Macy’s"

Converts To:  
Macy’s

NOTE: Some XML parsers may not require that the apostrophe be converted to the &apos; entity, when it appears inside an attribute that is enclosed in quotes as shown in Example 2 above.
Generating XML Data

The following examples show the conversion of “normal” data into XML data using the table from the previous page.

Example 1:
Original Data:
AT&T

XML Data:
CreditorName="AT&T"

Example 2:
Original Data:
William "Billy" Goat

XML Data:
_AttorneyName="William "Billy" Goat"
Specifying the DTD in the XML Data File

Near the beginning of XML data files, there is a DOCTYPE section that specifies the location of the DTD that is used to validate the data in the XML file. Although the DTD itself can be included within the DOCTYPE section, it is usually maintained as a separate file. The DOCTYPE declaration specifies the name and/or location of the DTD file. The following line is a sample DOCTYPE declaration line from an XML data file.

```xml
<!DOCTYPE LOAN_APPLICATION SYSTEM "AUS2_1.DTD"> 
```

The first argument of the DOCTYPE declaration specifies the name of the “root” or main element of the data file – in this case, LOAN_APPLICATION. The remainder of the line specifies the type of DTD and its name and/or location. In this case, the DTD name is shown as “AUS2_1.DTD”. Since a directory path or URL (Uniform Resource Locator) is not specified for the DTD, the software processing the XML data file will look for the DTD in the same directory as the XML data file.

The next two DOCTYPE samples specify a DTD in a specific directory path – the first on a local hard drive and the following one located at the MISMO web site.

```xml
<!DOCTYPE LOAN_APPLICATION SYSTEM "C:/DTD/AUS2_1.DTD”>
<!DOCTYPE RESPONSE_GROUP PUBLIC "-//MISMO//DTD AUS//EN//2.1" "http://www.mismo.org/mismo/dtd/CreditResponse_v2_1.DTD”>
```

The important point of this section is that when sending XML data files to your trading partners, they may have differing requirements for where the DTD is to be found by their processing software. Some may not require a specific location for the DTD and will use the default DOCTYPE that specifies the DTD name without a path or URL. Other trading partners may ask you to specify a specific directory path or URL for locating the DTD.
All of the XML sample data shown in this document includes formatting characters to make the data more readable. Each element or attribute is on its own line, and data within element containers is indented. These space, tab, carriage return and line feed characters are known as “white space” and create a nicely formatted output as shown below.

Sample XML Data with “White Space”:

```xml
<BORROWER BorrowerID="BorRec0001" _BirthDate="1970-11-25" _FirstName="JERRY" _MiddleName="VICTOR" _LastName="SMITH" _SSN="222449999" _UnparsedName="JERRY VICTOR SMITH" MaritalStatusType="NotProvided">
  <_RESIDENCE _StreetAddress="3750 S MIAMI AV" _City="MIAMI" _State="FL" _PostalCode="33014"/>
  <EMPLOYER _Name="Knight-Ridder Newspapers" _StreetAddress="811 Biscayne Blvd" _City="Miami" _State="FL" _PostalCode="33127" EmploymentCurrentIndicator="Y" EmploymentPositionDescription="VP" IncomeEmploymentMonthlyAmount="5100"/>
</BORROWER>
```

The XML files that are exchanged between trading partners will normally be sent without any “white space” characters as shown below.

Sample XML Data without “White Space”:

```xml
<BORROWER BorrowerID="BorRec0001" _BirthDate="1970-11-25" _FirstName="JERRY" _MiddleName="VICTOR" _LastName="SMITH" _SSN="222449999" _UnparsedName="JERRY VICTOR SMITH" MaritalStatusType="NotProvided">
  <_RESIDENCE _StreetAddress="3750 S MIAMI AV" _City="MIAMI" _State="FL" _PostalCode="33014"/>
  <EMPLOYER _Name="Knight-Ridder Newspapers" _StreetAddress="811 Biscayne Blvd" _City="Miami" _State="FL" _PostalCode="33127" EmploymentCurrentIndicator="Y" EmploymentPositionDescription="VP" IncomeEmploymentMonthlyAmount="5100"/>
</BORROWER>
```

“White space” characters are generally eliminated from production XML files for two reasons:

1. Eliminating the “white space” characters makes the XML files smaller (approximately 7% to 9%).
2. Most Internet browser software and XML editors automatically format the XML data with “white space” to make it more readable.
Handling Attributes with No Data

When processing XML data from your business partners, you should be aware that there are two valid ways of showing (or not showing) that an attribute has no data value.

The preferred method is just to not include the attribute name in the XML file. The following data sample shows a container record for a borrower with no middle name – and no _MiddleName attribute.

Sample Borrower - No Middle Name Attribute:

```xml
<BORROWER_FirstName="NATALIE" _LastName="FERGUSON" />
```

Some XML generation software will create an “empty” attribute, where the attribute name, equal sign and quotes are shown, but there is not data within the quotes.

Sample Borrower – “Empty” MiddleName Attribute:

```xml
<BORROWER_FirstName="NATALIE" _MiddleName="" _LastName="FERGUSON" />
```

NOTE: Enumerated attributes, ID attributes, and Boolean attributes cannot be displayed as “empty”. This would cause the XML files to fail validation.

Ordering Elements and Attributes

Elements

Data elements contained within any of the major components must appear in the order specified by the DTD or schema. Below is a DTD sample for the REQUEST_GROUP container element. Its elements (REQUESTING_PARTY, RECEIVING_PARTY, SUBMITTING_PARTY AND REQUEST) must appear in the same order in the XML data file, as they appear in the DTD.

```xml
<!ELEMENT REQUEST_GROUP
  (REQUESTING_PARTY*, RECEIVING_PARTY?, SUBMITTING_PARTY?, REQUEST*)>
```

Attributes

When an element contains multiple attributes, the attributes may appear in any order in the XML data file.

```xml
<!ELEMENT KEY EMPTY>
<!ATTLIST KEY
  _Name CDATA #IMPLIED
  _Value CDATA #IMPLIED >
```
Chapter 6: HTTP Post Name/Value Pair Recommendation

(Last Modified: October 3, 2002)

NOTE: This recommendation was approved by the MISMO Architecture Work Group on December 12, 2002.

Many service providers are using the HTTP/HTTPS Post operation for web based transactions, but each one currently has different sets of name/value pairs used in the post. This recommendation provides a standard set of name value pairs for use in MISMO transactions.

The main purpose of the HTTP/HTTPS Post is to allow a lender to login to a service provider's system so that the request payload can be sent and processed. The service provider needs enough information to identify and validate the requester so that the request payload can be routed and processed properly. In this recommendation, the HTTP Post Names match existing elements already defined in the MISMO Logical Data Dictionary (LDD).

**Login Account Identifier** – The login ID of an individual user.

**Login Account Password** – The login password of an individual user.

**Internal Account Identifier** – This is the billing account ID assigned by the service provider to the lender. Some service providers may derive this data based on the **Login Account Identifier**.

**Lender Branch Identifier** – For lenders with multiple branches this is the ID that identifies a specific branch. Once again some service providers may derive this based on the **Login Account Identifier**.

**Request Data** – This is the final HTTP Post Name that contains the actual request “payload”.

Sample HTTP Post Transaction with Recommended Name/Value Pairs:

```
LoginAccountIdentifier=J Doe&LoginAccountPassword=e255Xb&InternalAccountIdentifier=ABC+Bank&LenderBranchIdentifier=Westlake&RequestData={...Request transaction data goes here...}
```