MISMO Version 3 Reference Model
General Information Guide
<table>
<thead>
<tr>
<th>Document Date</th>
<th>Revision List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 18 2013</td>
<td>Submitted for Public Review.</td>
</tr>
<tr>
<td>Nov 1 2013</td>
<td>Public Review Period Ended.</td>
</tr>
<tr>
<td>Jan 2 2014</td>
<td>Updated “Rules For Xlink Relationships” to incorporate new rules in MEG 0036 1.1 that were approved by MISMO Architecture Work Group.</td>
</tr>
<tr>
<td>Feb 4 2014</td>
<td>Chapter 2 - Updated “Data Types Tab” to add a reference to MEG 0007 Class Words document. Updated “V3.x.x.x Tab” to use final Version 3.3.0 build references. Chapter 3 – Added ‘Cardinality’ section Chapter 4 – Updated “Major Elements of the Schema” to make reference to the four V3 root elements which now include PARTY. Swap the LOAN and PARTY sections so that PARTY is grouped with the other three root elements. Updated the DOCUMENT – REFERENCE and PARTY REFERENCE sections to add XML samples.</td>
</tr>
<tr>
<td>Mar 30, 2015</td>
<td>Chapter 4 – Updates to clarify of use of MESSAGE/DEAL_SETS and MESSAGE/DOCUMENT_SETS and the use of PARTIES</td>
</tr>
<tr>
<td>Apr 16, 2015</td>
<td>Chapter 4 – Updates from review by AWG for clarification of use of MESSAGE/DEAL_SETS and MESSAGE/DOCUMENT_SETS and the use of PARTIES</td>
</tr>
<tr>
<td>Apr 23, 2015</td>
<td>Chapter 3 – Updated to clarify selecting the proper RELATIONSHIPS level. Chapter 7 – Updated to add link for V3 SMART Docs FAQ. Final Review – Approved for publication by Architecture Work Group.</td>
</tr>
<tr>
<td>Feb 11, 2016</td>
<td>Chapter 3 – Added “Attributes of Data Elements” sections to describe the most commonly used attributes. Chapter 4 – Updated show new V3.4 elements added to DEAL (COMMUNICATION EVENTS, DEAL DETAIL, and SUPPORTING RECORD SETS), SERVICES (CLOSING and REASONS), and PARTY / ROLE (GOVERNMENT MONITORING). Final Review – Approved for publication by Architecture Work Group.</td>
</tr>
<tr>
<td>Jul 28, 2016</td>
<td>Chapter 4 – Updated the LEGAL ENTITY DETAIL section to add a reference to the About Legal Entity Identifier section of Chapter 8. Chapter 8 – New chapter on Universal Loan Identifier and Legal Entity Identifier. Final Review – Approved for publication by Architecture Work Group.</td>
</tr>
</tbody>
</table>
MISMO Version 3 Reference Model

General Information Guide

<table>
<thead>
<tr>
<th>Document Date</th>
<th>Jul 28, 2016</th>
</tr>
</thead>
</table>
| Editors         | Mike Bixby, Bixby Consulting/eHereNow – mbixby@eherenow.com
|                 | Karen Field, CoreLogic – kfield@corelogic.com |
| Contributors    | Steve Acker, Closergeist LLC – steve@closergeist.com
|                 | Ted Adams, Freddie Mac – Ted_Adams@freddiemac.com
|                 | Greg Alvord, RealEC – Gregory.Alvord@RealEC.com
|                 | Nora Beyerle, Harland Financial Solutions – Nora.Beyerle@harlandfs.com
|                 | Leo Bijnagte, Wells Fargo – Leendert.Bijnagte@wellsfargo.com
|                 | Lynn Calahan, Wells Fargo – Lynn.M.Calahan@wellsfargo.com
|                 | Jim Cooper, Freddie Mac, James_Cooper@freddiemac.com
|                 | Mike Fleck, RealEC – Michael.Fleck@RealEC.com
|                 | Harry Gardner, SigniaDocs – Harry.Gardner@elliemae.com
|                 | Elizabeth Green, rel-e-vant – liz@rel-e-vant.com
|                 | Matthew Hailstone, Simplifile – matthew.hailstone@simplifile.com
|                 | Dave Krause, Radian Guaranty Inc. – david.krause@radian.biz
|                 | Abdias Lira, Wolters Kluwer – Abdias.Lira@WoltersKluwer.com
|                 | Jim Metzger, D+H – Jim.Metzger@dh.com
|                 | Rachael Sokolowski, Magnolia Technologies, rsokolowski@magnoliatech.com
|                 | Chris Ulbright, DocMagic – chris@docmagic.com
|                 | Gloria Zimmer, MERSCORP Holdings – gloriaz@mersinc.org

COPYRIGHT 2016 MORTGAGE INDUSTRY STANDARDS MAINTENANCE ORGANIZATION (MISMO)
ALL RIGHTS RESERVED.

THIS MISMO STANDARD INCLUDES THE END USER LICENSE AGREEMENT ATTACHED HERETO AT WWW.MISMO.ORG/ABOUTMISMO/POLICIESANDPROCEDURES.HTM AND IS GOVERNED BY AND SUBJECT TO THE END USER LICENSE AGREEMENT. NO USER OF THIS STANDARD MAY REMOVE THIS REFERENCE TO AND STATEMENT REGARDING THE END USER LICENSE. ANY HARD COPY PUBLICATION OF THIS STANDARD MUST INCLUDE AND ATTACH A HARD COPY PRINT OUT OF THE END USER LICENSE. ANY FURTHER ELECTRONIC DISTRIBUTION OF THIS STANDARD MUST INCLUDE A SPECIFIC REFERENCED LINK TO THE END USER LICENSE AGREEMENT OR OTHER MEANS OF ATTACHMENT OF THE END USER LICENSE AGREEMENT.

DISCLAIMER: THIS MISMO STANDARD IS PROVIDED “AS IS.” MISMO, THE MORTGAGE BANKERS ASSOCIATION OF AMERICA (“MBA”), THE COPYRIGHT HOLDER, THE AUTHORS OF THIS MISMO STANDARD AND ANY STANDARD-SETTING BODY PARTICIPANTS TO THIS MISMO STANDARD MAKE NO REPRESENTATIONS OR WARRANTIES (I) EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT; (II) THAT THE CONTENTS OF SUCH MISMO STANDARD ARE FREE FROM ERROR OR SUITABLE FOR ANY PURPOSE; NOR THAT IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD-PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. IN NO EVENT WILL MISMO, MBA, THE COPYRIGHT HOLDER OR THE STANDARD-SETTING BODY PARTICIPANTS TO THIS MISMO STANDARD BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES FOR ANY USE OF THIS MISMO STANDARD, INCLUDING, WITHOUT LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, LOSS OF PROGRAMS OR OTHER DATA ON YOUR INFORMATION HANDLING SYSTEM OR OTHERWISE, EVEN IF MISMO, MBA, THE COPYRIGHT HOLDER AND/OR ANY AUTHORS AND/OR ANY STANDARD-SETTING BODY PARTICIPANTS TO THIS MISMO STANDARD ARE EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
# Table of Contents

Chapter 1: Getting Started........................................................................................................................................ 1
   What Else You Need........................................................................................................................................ 1
   About MISMO .................................................................................................................................................. 1
   MISMO’s Objective ........................................................................................................................................ 2
   Who Participates in MISMO? .................................................................................................................... 2
   Mortgage Industry Areas ............................................................................................................................ 2
   The MISMO Reference Model...................................................................................................................... 3
   The MISMO Development Process............................................................................................................. 4
   Types of Workgroups .................................................................................................................................. 4
   The Development Process ............................................................................................................................ 5
   Who Uses the MISMO Reference Model? .................................................................................................. 5
   Data Architecture........................................................................................................................................ 5
   Data Quality / Data Stewardship / Master Data Management .................................................................... 5
   Business Requirements Definition ............................................................................................................ 6
   Database Design / Data Modeling .................................................................................................................. 6
   Business Users / Regulators / Compliance Officers .................................................................................. 6

Chapter 2: Reference Model – Logical Data Dictionary......................................................................................... 7
   V3.x.x.x Tab .................................................................................................................................................. 7
   Data Points Tab .......................................................................................................................................... 8
   Containers Tab .......................................................................................................................................... 8
   Attributes Tab .......................................................................................................................................... 9
   Arc Roles Tab ........................................................................................................................................... 10
   Enumerations Tab .................................................................................................................................... 12
   Data Types Tab ....................................................................................................................................... 12
   Acronyms Tab .......................................................................................................................................... 13
   Usages Tab ............................................................................................................................................... 14
   Data Points (Dep Ver) Tab ....................................................................................................................... 14
   Containers (Dep Ver) Tab ......................................................................................................................... 14
# Table of Contents

Data Points (Dep Cum) Tab ................................................................. 15  
Containers (Dep Cum) Tab ................................................................. 15  

Chapter 3: Reference Model – XML Schema ...................................... 16  

Data Elements..................................................................................... 16  
  Attributes of Data Elements ............................................................... 16  
    Identifier Owner URI Attribute ....................................................... 16  
    Ignore Time Segment Indicator Attribute ...................................... 16  
    Sensitive Indicator Attribute .......................................................... 17  
    Data Not Supplied Attributes ........................................................ 17  
    Display Time Zone Text Attribute ................................................. 18  

Container Elements ......................................................................... 18  
Cardinality ......................................................................................... 20  

Data Relationships............................................................................. 20  
  One-to-One Relationship .................................................................. 20  
  One-to-Many Relationship .............................................................. 21  
  Many-to-Many Relationship ............................................................ 21  
  Reflexive Relationship .................................................................... 22  
  Irreflexive Relationship ................................................................... 22  

Rules for XLink Relationships ............................................................ 22  
  Overview .......................................................................................... 22  

RELATIONSHIP Element Types “From” and “To” End Point Conditions ........................................ 24  
RELATIONSHIP Element Attributes: from, to and arcrole .......................................................... 25  
RELATIONSHIP Element “Arcrole” Naming Convention .............................................................. 25  
RELATIONSHIP Element “Arcrole” FromName/ToName Value Order ........................................ 26  

Chapter 4: Major Elements of the Schema ......................................... 27  
MESSAGE Element ............................................................................ 27  
MESSAGE Attributes .......................................................................... 28  
ABOUT_VERSIONS / ABOUT_VERSION Elements ........................................ 29  
DEAL_SETS Element .......................................................................... 29  
DOCUMENT_SETS Element .............................................................. 31  
RELATIONSHIPS / RELATIONSHIP Elements ....................................... 32
<table>
<thead>
<tr>
<th>Document Element</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM_SIGNATURES / SYSTEM_SIGNATURE Elements</td>
<td>32</td>
</tr>
<tr>
<td>DEAL Element</td>
<td>32</td>
</tr>
<tr>
<td>DEAL Attributes</td>
<td>32</td>
</tr>
<tr>
<td>REFERENCE Element</td>
<td>34</td>
</tr>
<tr>
<td>ABOUT_VERSIONS / ABOUT_VERSION Elements</td>
<td>34</td>
</tr>
<tr>
<td>ASSETS Element</td>
<td>35</td>
</tr>
<tr>
<td>COLLATERALS Element</td>
<td>36</td>
</tr>
<tr>
<td>COMMUNICATION EVENTS / COMMUNICATION_EVENT Elements</td>
<td>39</td>
</tr>
<tr>
<td>DEAL DETAIL Elements</td>
<td>39</td>
</tr>
<tr>
<td>EXPENSES / EXPENSE Elements</td>
<td>39</td>
</tr>
<tr>
<td>LIABILITIES / LIABILITY Elements</td>
<td>40</td>
</tr>
<tr>
<td>LOANS Element</td>
<td>40</td>
</tr>
<tr>
<td>PARTIES / PARTY Elements</td>
<td>41</td>
</tr>
<tr>
<td>RELATIONSHIPS / RELATIONSHIP Elements</td>
<td>41</td>
</tr>
<tr>
<td>SERVICES / SERVICE Elements</td>
<td>42</td>
</tr>
<tr>
<td>SUPPORTING_RECORD_SETS / SUPPORTING_RECORD_SET Elements</td>
<td>43</td>
</tr>
<tr>
<td>DOCUMENT Element</td>
<td>43</td>
</tr>
<tr>
<td>Document Concepts</td>
<td>43</td>
</tr>
<tr>
<td>DOCUMENT Attributes</td>
<td>44</td>
</tr>
<tr>
<td>REFERENCE Element</td>
<td>44</td>
</tr>
<tr>
<td>UNKNOWN_VERSION3DOCUMENT Element</td>
<td>45</td>
</tr>
<tr>
<td>AUDIT_TRAIL Element</td>
<td>45</td>
</tr>
<tr>
<td>DEAL_SETS Element</td>
<td>45</td>
</tr>
<tr>
<td>MAP Element</td>
<td>46</td>
</tr>
<tr>
<td>RELATIONSHIPS / RELATIONSHIP Elements</td>
<td>46</td>
</tr>
<tr>
<td>SIGNATORIES / SIGNATORY Elements</td>
<td>46</td>
</tr>
<tr>
<td>SYSTEM_SIGNATURES / SYSTEM_SIGNATURE Elements</td>
<td>46</td>
</tr>
<tr>
<td>VIEWS / VIEW Elements</td>
<td>46</td>
</tr>
<tr>
<td>ABOUT_VERSIONS / ABOUT_VERSION Elements</td>
<td>47</td>
</tr>
<tr>
<td>DOCUMENT_CLASSIFICATION Element</td>
<td>47</td>
</tr>
<tr>
<td>Document Profiles</td>
<td>47</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>PARTY Element</td>
<td>49</td>
</tr>
<tr>
<td>REFERENCE Element</td>
<td>50</td>
</tr>
<tr>
<td>INDIVIDUAL Element</td>
<td>51</td>
</tr>
<tr>
<td>LEGAL_ENTITY Element</td>
<td>52</td>
</tr>
<tr>
<td>ROLES / ROLE Elements</td>
<td>53</td>
</tr>
<tr>
<td>GOVERNMENT MONITORING Element (V3.3.1 &amp; Earlier)</td>
<td>55</td>
</tr>
<tr>
<td>GOVERNMENT MONITORING Element (V3.4 &amp; Later)</td>
<td>56</td>
</tr>
<tr>
<td>LOAN Element</td>
<td>58</td>
</tr>
<tr>
<td>LOAN ROLE Attribute</td>
<td>58</td>
</tr>
<tr>
<td>LOAN_STATE Element</td>
<td>59</td>
</tr>
<tr>
<td>LOAN ROLE – LOAN STATE Use Cases</td>
<td>60</td>
</tr>
<tr>
<td>Non-Modified Loans Delivered to Investor</td>
<td>60</td>
</tr>
<tr>
<td>Modified Loans Delivered to Investor</td>
<td>61</td>
</tr>
<tr>
<td>Converted Loans</td>
<td>62</td>
</tr>
<tr>
<td>Reset Balloon Loans</td>
<td>63</td>
</tr>
<tr>
<td>Second Lien Subject Loan with Related First Lien Loan</td>
<td>64</td>
</tr>
<tr>
<td>Loans with Concurrent Secondary Financing</td>
<td>65</td>
</tr>
<tr>
<td>ADJUSTMENT Element</td>
<td>66</td>
</tr>
<tr>
<td>INDEX RULE</td>
<td>66</td>
</tr>
<tr>
<td>LIFETIME ADJUSTMENT RULE Elements</td>
<td>67</td>
</tr>
<tr>
<td>CONVERSION OPTION PERIOD ADJUSTMENT RULES</td>
<td>68</td>
</tr>
<tr>
<td>PER CHANGE ADJUSTMENT RULES</td>
<td>68</td>
</tr>
<tr>
<td>PERIODIC ADJUSTMENT RULES</td>
<td>69</td>
</tr>
<tr>
<td>RATE OR PAYMENT CHANCE OCCURRENCES</td>
<td>72</td>
</tr>
<tr>
<td>Chapter 5: Extending the MISMO Standard</td>
<td>73</td>
</tr>
<tr>
<td>Goals of the MISMO Extension Model</td>
<td>73</td>
</tr>
<tr>
<td>Guidelines for Using the EXTENSION Element</td>
<td>73</td>
</tr>
<tr>
<td>EXTENSION Element – Version 3.2 and Later</td>
<td>74</td>
</tr>
<tr>
<td>EXTENSION Element – Versions 3.0 and 3.1</td>
<td>75</td>
</tr>
<tr>
<td>Adding Data Point Elements</td>
<td>75</td>
</tr>
<tr>
<td>Adding Container Elements</td>
<td>76</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Adding Schema Validation for EXTENSION Elements</td>
<td>77</td>
</tr>
<tr>
<td>Approach 1: Modified Schema Files Using xs:redefine</td>
<td>77</td>
</tr>
<tr>
<td>Approach 2: Modifying MISMO Extension Schema Files</td>
<td>77</td>
</tr>
<tr>
<td>Chapter 6: Implementing MISMO: Programming Guidance</td>
<td>79</td>
</tr>
<tr>
<td>Reading and Processing MISMO XML</td>
<td>79</td>
</tr>
<tr>
<td>Using an XML Object Model</td>
<td>79</td>
</tr>
<tr>
<td>Validating Against a Schema</td>
<td>80</td>
</tr>
<tr>
<td>Performance Considerations</td>
<td>80</td>
</tr>
<tr>
<td>Creating MISMO Data XML</td>
<td>80</td>
</tr>
<tr>
<td>Use an XML Object Model or Other Built-in XML Support</td>
<td>80</td>
</tr>
<tr>
<td>Use a Template File</td>
<td>81</td>
</tr>
<tr>
<td>Handling Large Sets of Data</td>
<td>81</td>
</tr>
<tr>
<td>Transforming Other XML Data Formats to MISMO</td>
<td>81</td>
</tr>
<tr>
<td>Sending and Receiving MISMO XML</td>
<td>82</td>
</tr>
<tr>
<td>Conclusion</td>
<td>82</td>
</tr>
<tr>
<td>Chapter 7: Version 3 SMART Doc®</td>
<td>83</td>
</tr>
<tr>
<td>Overview</td>
<td>83</td>
</tr>
<tr>
<td>The V3 SMART Doc® I-Guides</td>
<td>83</td>
</tr>
<tr>
<td>SMART Doc® PDF I-Guide</td>
<td>83</td>
</tr>
<tr>
<td>XML Digital Signature I-Guide</td>
<td>84</td>
</tr>
<tr>
<td>Zip Archive Implementation Guide</td>
<td>84</td>
</tr>
<tr>
<td>Earlier SMART Doc® I-Guides</td>
<td>85</td>
</tr>
<tr>
<td>eMortgage Guide</td>
<td>85</td>
</tr>
<tr>
<td>eMortgage Closing Guide</td>
<td>85</td>
</tr>
<tr>
<td>eRecording – Electronic Document Formatted Recordable Instruments</td>
<td>85</td>
</tr>
<tr>
<td>eSigned PDF Guidelines</td>
<td>86</td>
</tr>
<tr>
<td>eMortgage Vaulting Guide</td>
<td>87</td>
</tr>
<tr>
<td>Chapter 8: Universal Loan Identifier</td>
<td>88</td>
</tr>
<tr>
<td>New Identification Requirements</td>
<td>88</td>
</tr>
<tr>
<td>CFPB Rule for Universal Loan Identifier: Format Requirements</td>
<td>89</td>
</tr>
<tr>
<td>About the Legal Entity Identifier</td>
<td>89</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Multi-Services Scenario</td>
<td>112</td>
</tr>
<tr>
<td>Security Requirements</td>
<td>112</td>
</tr>
<tr>
<td>Possible Security Solutions</td>
<td>113</td>
</tr>
<tr>
<td>Authentication Solutions</td>
<td>113</td>
</tr>
<tr>
<td>Integrity Solutions</td>
<td>114</td>
</tr>
<tr>
<td>Security Definitions</td>
<td>114</td>
</tr>
</tbody>
</table>
CHAPTER 1: GETTING STARTED

This chapter provides an overview of this Guide as well as an overview of MISMO and its processes. It includes the following sections:

- What else you will need
- About the MISMO organization
- The MISMO development process
- Who uses the MISMO Reference Model?

What Else You Need

In addition to this guide, to implement MISMO Version 3, you must have the following items, which you can find on the MISMO Web site at www.mismo.org:

- **Industry Workgroup Implementation Guides**: Each industry workgroup produces an implementation guide to document its area of the Standard. These provide more detailed guidance for implementing the Reference Model.
- **Logical Data Dictionary (LDD) files**: The LDD files define each mortgage data element used in the Reference Model.
- **XML Schema**: The schema defines the structure of the mortgage data sets. Use these files to develop, write, and read XML data files.
- **Your own data**: In implementing the MISMO Reference Model, you must convert your own data either to or from an XML format.

About MISMO

The Mortgage Bankers Association (MBA) created the Mortgage Industry Standard Maintenance Organization, commonly known MISMO, in October, 1999. This section provides the following information about MISMO:

- MISMO’s objective
- Who Participates in MISMO?
- Mortgage Industry Areas
- The MISMO Reference Model

For detailed information about MISMO’s structure and mission, see the www.mismo.org/AboutMISMO web page.
MISMO’s Objective

MISMO’s purpose is to develop and maintain the XML data formats businesses use to exchange data electronically in mortgage transactions. This XML protocol is called the “MISMO Reference Model”. To document the MISMO Reference Model, the MISMO workgroups work together to produce the following:

- A data dictionary (the LDD) to provide business definitions and corresponding architecture data element tag names.
- An XML architecture that encompasses all data contributed by the Industry Workgroups.

Who Participates in MISMO?

MISMO is made up of hundreds of mortgage industry professionals who meet regularly to discuss the requirements of exchanging mortgage data electronically. Anyone working in the mortgage industry can participate in MISMO.

Mortgage Industry Areas

MISMO has grouped the business areas of the mortgage industry into four major categories: Origination, Servicing, Secondary, and Real Estate Services. Each category covers specific business processes within that industry area.

- Origination includes the industry areas of Mortgage Application, Underwriting, and Closing.
- Servicing includes Loan Setup & Transfer, Investor Reporting and Default Reporting and Non-Performing Loan data.
- Secondary includes Delivery of Loans to Investors, Securitization, Bulk Pool Transfer, Funding, and Pricing & Discovery.
- Real Estate Services include Property Valuation, Credit Reporting, Flood, Mortgage Insurance, Title, and other mortgage related services.

The Core Data Structures Workgroup coordinates data common to multiple industry areas, such as borrower and property elements.
This image diagrams the relationships among categories and industry areas.

The MISMO Reference Model
A reference model is a framework used to structure information. The MISMO Reference Model structures the concepts and data points used for the mortgage industry.

The MISMO Reference Model is documented in two formats:

- **Logical Data Dictionary (LDD):** To document the Reference Model, MISMO creates a Logical Data Dictionary (LDD) that defines each mortgage data element. The result is a single, central data set for the mortgage industry. The borrower, employment, property and other commonly used information have a common data definition, no matter which mortgage industry sector or process is using the data.
• **XML Schema:** The MISMO V3 XML Schema specifies how the data elements are organized into a logical, well-defined structure that allows for both the exchange of data and documents as well as its use within a system or enterprise.

**The MISMO Development Process**

To create the Reference Model and the resulting LDD and XML Schema, MISMO has organized itself into *workgroups*. A workgroup is a group of mortgage professionals who meet regularly to define the XML Reference Model for a mortgage industry area or an overarching function of the MISMO Reference Model.

A workgroup’s charter is to produce an LDD and an implementation guide for its industry area.

**Types of Workgroups**

Three of the MISMO workgroups guide the overall structure of the XML Reference Model or the MISMO organization itself. These workgroups include Architecture, Core Data Structures, and the MISMO Council of Chairs.

- The Architecture Workgroup provides overall technical direction and support for development of the MISMO Reference Model. Representatives from the industry workgroups make up the Architecture workgroup.
- The Core Data Structures (CDS) workgroup makes sure that all workgroups use the same data definitions and business concepts across the Reference Model. CDS uses data modeling and relational concepts to do so. Business data experts and technical experts from the industry workgroups make up the CDS workgroup.
- The MISMO Council of Chairs includes all industry work groups’ chairpersons. It provides a forum for workgroups to share information with each other.

*Industry workgroups* focus on the interests of a particular area and include subject matter experts for that industry area. Examples of industry workgroups include Credit, Verification Services, eMortgage, Origination, Appraisal, and others.

*Development workgroups* are formed to produce a specific item. Rather than being an ongoing workgroup, a development workgroup meets to produce its deliverable and then disbands once the project is complete. An example of a development workgroup is the MISMO General Implementation Guide (I-Guide) Workgroup, which has met over several months to produce this Implementation Guide. Once the MISMO I-Guide workgroup has published its implementation guide, it will disband, but it might form again to develop another I-guide when the MISMO Reference Model is updated.
The Development Process

MISMO has defined a process for the development and maintenance of the Reference Model and other Documents. The details of this process may be found in the MISMO Development Process Document, available on the MISMO web site (www.mismo.org).

Who Uses the MISMO Reference Model?

Within the mortgage industry there are a great variety of participants, some big, some small. There are the lenders, government sponsored enterprises, mortgage insurers, and the various providers of mortgage-related services such as property appraisals, credit reports, flood certifications, hazard insurers, and title services and so on. There are also the investors who purchase pools of loans as mortgage backed securities, as well as the government agencies who monitor and regulate the industry.

Within an organization there are many ways to use the components of the MISMO Reference model. Here are a few examples of how MISMO is used by various departments of an organization.

Data Architecture

In addition to the MISMO Reference Model, the MISMO Engineering Guidelines (MEGs) are another resource available to CIOs and Data Architects. The MEGs cover a variety of useful topics including defining data classes, namespaces, data elements and attributes, use of acronyms, data extensions, and version release processes. The MEGs are developed primarily for usage by MISMO in the development of the Reference Model; however, they may also be useful to some companies for their own internal policies. Established companies may already have architecture policies that are much more comprehensive that those defined within the MEGs. Others may find them useful for building or augmenting their own data architecture policies.

Data Quality / Data Stewardship / Master Data Management

One of the big challenges for data quality and data stewardship teams is making sure that the meaning and validation parameters of each data point under their supervision are clearly documented and understood. This becomes especially critical when mapping data to and from external sources where the data may not always have good accompanying data dictionaries. The MISMO standards have been implemented in mortgage service transactions and automated underwriting systems for well over ten years. The extensive use of MISMO data point names and definitions over this period of time have made the MISMO LDD and Schema natural resources for an internal data dictionary or as the core of a full Master Data Management system.
**Business Requirements Definition**

Almost every business project begins by defining the business requirements. The MISMO LDD can be a valuable resource for business systems analysts tasked with identifying data points to be included in the business requirements and assigning them a standard industry name and definition.

For data points that have a limited set of possible values, the LDD documents the enumerated attribute values that the MISMO industry work groups have compiled. From the MISMO list of valid data points the analyst can either restrict the list of values or extend it depending on their business use of each data point.

**Database Design / Data Modeling**

The MISMO LDD and the XML Schema are valuable resources for designers and modelers of databases being used for a mortgage data warehouse or an operational data store. When designing a logical data model for their application, many times the data modelers can use the data point names directly from the MISMO LDD or modified names that meet the requirements of the target database or their internal database architecture rules and naming conventions.

When defining the data table structures and their organization, the MISMO XML Schema can provide guidance in both the naming of the tables and their relationship with each other. The MISMO XML schema can also help the data modeler understand the one-to-one, one-to-many and many-to-many relationships within the MISMO data set.

**Business Users / Regulators / Compliance Officers**

The purpose of information technology is to support the business entity and allow it to be more productive, accurate and operate more efficiently. The use of industry standards such as the MISMO Reference Model helps towards that goal. The use of a single, clearly defined data standard for all data exchanges reduces the cost of implementing connections between users and validating the quality of the data.

The use of MISMO standards within the mortgage industry also helps the regulators who oversee the industry and verify compliance with applicable laws and regulations. MISMO can become not just the source for a library of mortgage industry terms and definitions, but also the target for implementing new changes and regulations where they affect the mortgage data itself. Using the MISMO Reference Model data throughout the mortgage process for individual loans has led to greater visibility of the data and better data quality.
CHAPTER 2: REFERENCE MODEL – LOGICAL DATA DICTIONARY

The first component of the MISMO Reference Model is the Logical Data Dictionary (LDD). The LDD is a listing of individual data elements, containers for those data elements, and other reference information about the data elements that are useful for business analysts and software developers. The LDD is formatted as an Excel workbook format for most users but is also available as an XML document format for automated system use.

The LDD workbook is made up of worksheets containing alphabetical lists of each data point, container, attribute, enumerated values, data types, acronyms and arc role (container relationships) used in the MISMO V3 Reference Model. There may be minor differences in the formatting of the LDD worksheets across versions but the basic concepts remain the same. This section describes some of the foundational concepts presented in each the MISMO LDD worksheet tabs.

V3.x.x.x Tab

This worksheet tab identifies the version identifier of the LDD and Reference Model, legal declarations, and statistical information about the MISMO data set – data point counts, container counts, etc.

- **Build** = “B298”: internal MISMO identifier for each Reference Model / LDD Build. The Build identifier is generated by the system that generates the MISMO Reference Model. There could be several Builds created before the release of a specific LDD Version or Model Version.
- **Date** = “2014-02-04”: date that the version of the “build” was created.
- **Model Version** = “3.3.0”: identifier for the Reference Model, the XML schema that defines the MISMO data structure.
- **Version** = “3.3.0.0”: identifier for this release of the Logical Data Dictionary.
Data Points Tab

This worksheet tab of the LDD contains a list of each of the data points (Simple Type XML elements) used in the MISMO V3 Reference Model. Here are some of the features of the **Data Points Tab** of the worksheet:

- Lists data points alphabetically by their term name.
- Provides a definition for each data element.
- For data elements that only allow specific values, provides that list of values with a value definition, if it is needed. This tab will list up to the first 30 enumerations. If the number of enumerations exceeds that number, you must refer to the Enumerations Tab for a complete list.
- Lists the names of the data containers in which the data point is used plus a total usage count.
- Lists all locations within the model in which the data point appears, also known as *XPaths*. This tab will list up to the first 30 usages. If the number of usages exceeds that number, you must refer to the Usages Tab for a complete list.
- Lists the data type of each data point (name, date, identifier, text, percent, amount, etc.)
- List the XML attributes associated with the data point.

### Containers Tab

This worksheet tab of the LDD contains a list of each “container” element used in the MISMO V3 Reference Model. Here are some of the features of the **Containers Tab** of the worksheet:
• Lists containers alphabetically by their term name.
• Provides a definition for each container.
• List the data points held in that container and a total of the number of those data points in that container. Lists the name/s of the parent container/s in which the container is used plus a total usage count.
• Lists the XPath locations within the model in which the container appears. This tab will list up to the first 30 usages. If the number of usages exceeds that number, you must refer to the Usages Tab for a complete list.
• List the XML attributes associated with the container.

Attributes Tab

This worksheet tab contains an alphabetical list of the attributes defined in the V3.x Reference Model. Attributes define additional qualities about either containers or data points. Here are some of the features of the Containers Tab of the worksheet:

• Lists attributes alphabetically by their term name.
• Provides a definition for each attribute.
• For attributes that only allow specific values, provides that list of values with a value definition, if it is needed.
• Lists all data points or containers that have the attribute, plus a total usage count. This tab will list up to the first 30 usages. If the number of usages exceeds that number, you must refer to the Usages Tab for a complete list.
• Lists the data type associated with the attribute.
The MISMO Reference Model uses an XML specification called XLink to define data relationships in a MISMO message that are not naturally expressed by the MISMO container hierarchy. One of the XLink attributes, arcrole, describes the type of relationship between source data and target data. The XLink from attribute identifies the source data point or container of the relationship. The XLink to attribute identifies the target data point or container of the relationship. The XLink label attribute is an identifier that is attached to data point or container elements that will be used to express an arcrole relationship. The MISMO XML RELATIONSHIP elements hold the XLink arcrole, from and to attributes that describe the data relationships.

This worksheet tab contains an alphabetical list of the Arcroles defined in the V3.x Reference Model. Here are some of the features of the Arcrole Tab of the worksheet:

- Lists Arcroles alphabetically by their name.
- Provides a brief description of the business usage.
- Lists the XLink label.
- Lists the source (from) and target (to) of the relationship.
Below is a simplified XML sample that shows how the XLink attributes are used to establish a relationship between a source and target data point or container element. In this mortgage deal sample below, there are two assets, three liabilities and two borrowers. Note that the third liability is a joint account associated with both borrowers.

XLink label attributes are added to each ASSET element, each LIABILITY element, and each PARTY element’s ROLE element. The MISMO RELATIONSHIP element identifies which liability is associated with each borrower using the XLink arcrole, from and to attributes.

```
DEAL
ASSETS
  ASSET xlink:label = "Asset1" (Auto - $21,200)
  ASSET xlink:label = "Asset2" (Stocks - $132,000)
LIABILITIES
  LIABILITY xlink:label = "Liability1" (GMAC - $23,700)
  LIABILITY xlink:label = "Liability2" (Sears - $400)
  LIABILITY xlink:label = "Liability3" (Visa - $6,700)
PARTIES
  PARTY / ROLE xlink:label="Borrower1" (John Doe)
  PARTY / ROLE xlink:label="Borrower2" (Jane Doe)
RELATIONSHIPS
  RELATIONSHIP xlink:from="Asset1" xlink:to="Borrower2"
    xlink:arcrole="ASSET_IsAssociatedWith_ROLE"
  RELATIONSHIP xlink:from="Asset2" xlink:to="Borrower2"
    xlink:arcrole="ASSET_IsAssociatedWith_ROLE"
  RELATIONSHIP xlink:from="Liability1" xlink:to="Borrower1"
    xlink:arcrole="LIABILITY_IsAssociatedWith_ROLE"
  RELATIONSHIP xlink:from="Liability2" xlink:to="Borrower1"
    xlink:arcrole="LIABILITY_IsAssociatedWith_ROLE"
  RELATIONSHIP xlink:from="Liability3" xlink:to="Borrower1"
    xlink:arcrole="LIABILITY_IsAssociatedWith_ROLE"
  RELATIONSHIP xlink:from="Liability3" xlink:to="Borrower2"
    xlink:arcrole="LIABILITY_IsAssociatedWith_ROLE"
```

There is a more detail discussion in the “Rules for XLink Relationships” section of the “Reference Model – XML Schema” chapter of this guide.
Enumerations Tab

This worksheet tab shows the term name, definitions and valid values for all enumerated data and attributes. An *enumeration* is a specific data value for data point. Here are some of the features of the Enumerations Tab of the worksheet:

- Lists alphabetically the enumerated term.
- For ease of use, repeats the definition and usage count for the enumerated term which may also be found on the Data Point / Attribute Tabs.
- Lists alphabetically all enumerations and their definitions. This is the complete list. When the number of enumerations exceeds 30, you will need to use this tab to see the entire list.
- Lists the Base term name for the data point that is used in the XML schema.

Data Types Tab

This worksheet tab of the LDD contains a list of each of the basic MISMO Data Types for the data points (Simple Type XML elements) used in the MISMO V3 Reference Model.

**NOTE:** The MISMO Engineering Guidelines (MEG 0007) for Class Words discusses this topic in more detail. It is available at the link below:


Here are some of the features of the **Data Types Tab** of the worksheet:

- Lists MISMO data types by their name.
- Provides a definition for each data type.
- Lists the total number of data terms that use the data type and displays the first 30 of those. This tab will list up to the first 30 usages. If the number of usages exceeds that number, you must refer to the Usages Tab for a complete list.
- Lists XML Schema Type used to derive the MISMO Data Type, plus any additional constraints applied in the MISMO Data Type definition.
Acronyms Tab

This worksheet tab of the LDD contains the MISMO approved list of acronyms that are incorporated into some Data Point names and Attribute names. Acronyms are approved by MISMO for use based on guidelines defined by the MISMO Architecture Work Group.

NOTE: The MISMO Engineering Guidelines (MEG 0008 and 0008A) for Approved Acronyms discusses this topic in more detail. It is available at the link below:


Here are some of the features of the Acronyms Tab of the worksheet:

- Lists acronyms alphabetically.
- Provides a definition - the full name of the acronym.
- Lists the number of times it is used in the Reference Model and the data point names, acronym names and enumeration values where each acronym is used.
Usages Tab

This worksheet tab shows the XPaths for every use of data points, containers and attributes within the MISMO Reference Model. Here are some of the features of the Usages Tab of the worksheet:

- Lists alphabetically every data point, container and attribute.
- For ease of use, repeats the definition and usages for the data point / container / attribute which may also be found on the Data Point / Container / Attribute Tabs.
- Lists all XPath usages. This is the complete list. When the number of enumerations exceeds 30, you will need to use this tab to see the entire list.

Data Points (Dep Ver) Tab

This worksheet tab shows the “Deprecated Version” data points. This lists any data points that have been removed from the current major version since the previous major version.

For example, for the V3.3 major version the LDD would list any data points that have been deprecated since the last V3.2 version.

Containers (Dep Ver) Tab

This worksheet tab shows the “Deprecated Version” containers. This lists any containers that have been removed from the current major version since the previous major version.

For example, the V3.3 major version of the LDD would list any containers that have been deprecated since the last V3.2 version.
Data Points (Dep Cum) Tab

This worksheet tab shows the “Deprecated Cumulative” data points. This lists any data points that have been removed from the current major version since the first release of Version 3.0.0.

For example, for the V3.3 major version the LDD would list any data points that have been deprecated since V3.0.

Containers (Dep Cum) Tab

This worksheet tab shows the “Deprecated Cumulative” containers. This lists any containers that have been removed from the current major version since the first release of Version 3.0.0.

For example, for the V3.3 major version the LDD would list any containers that have been deprecated since V3.0.
CHAPTER 3: REFERENCE MODEL – XML SCHEMA

A reference model is a framework used to structure information. MISMO uses the industry standard XML Schema to provide an organized structure for the data points defined in the LDD.

The schema, which acts as an XML template, captures the data in mortgage transactions in XML elements. There are two types of elements: data elements and container elements. The elements can reuse each other’s data by linking to each other through data relationships. This section explains data elements, container elements, and the relationships that can tie them together.

Data Elements

Data elements capture concrete pieces of information in the XML transactions. For example, the LDD data point “Loan Maturity Date” is defined as the date when the loan is scheduled to be paid in full as reflected in the Note. In an XML document, this would be expressed as an XML data element LoanMaturityDate as shown below:

```xml
<LoanMaturityDate>2043-09-30</LoanMaturityDate>
```

Attributes of Data Elements

Some data elements may also have additional XML attribute that can be used to more precisely define the content. The attributes that are available for use with each data element are listed within the MISMO Schema and in the MISMO LDD. Below are XML samples showing the use of some of the more commonly used data element attributes. See the DataPoints tab, “Attributes” column of the LDD worksheet for a complete list.

Identifier Owner URI Attribute

Here is an XML example of the use of the IdentifierOwnerURI attribute to more precisely identify whose LoanIdentifier is being used in this XML data element:

```xml
<LoanIdentifier IdentifierOwnerURI="http://www.quickenloans.com">13-10595887</LoanIdentifier>
```

Ignore Time Segment Indicator Attribute

MISMO Datetime element values require both date and time values. When a time value is not available, the IgnoreTimeSegmentIndicator attribute is added. To validate against the schema, a zero time value must also be provided as shown in the following example:

```xml
<LoanIdentifier IdentifierOwnerURI="http://www.quickenloans.com">13-10595887</LoanIdentifier>
```
<CreatedDatetime IgnoreTimeSegmentIndicator="true">
  2015-12-02T00:00:00+00:00
</CreatedDatetime>

### Sensitive Indicator Attribute

The `SensitiveIndicator` attribute identifies an element data value that contains sensitive information so it can receive special treatment or processing.

```
<TaxpayerIdentifierValue SensitiveIndicator="true">
  333445555
</TaxpayerIdentifierValue>
```

### Data Not Supplied Attributes

Beginning with Version 3.4.0, three new attributes were added to support Evidence of Compliance. When data for an element is not available these attributes describe why the data was not available. Here are the three attributes that are now available for use with all data elements:

- **DataNotSuppliedReasonType** (`NotCollected | NotReceived | NotRelevant | Omitted | Other`)
- **DataNotSuppliedReasonTypeOtherDescription**
- **DataNotSuppliedReasonTypeAdditionalDescription**

Below is an XML snippet for a borrower `NAME` element with a missing `LastName` data value.

```
<NAME>
  <FirstName>Prince</FirstName>
  <LastName DataNotSuppliedReasonType="NotReceived"
           DataNotSuppliedReasonTypeAdditionalDescription="Borrower states he only uses one name"/>
</NAME>
```

MISMO data element names ending in the class words: Amount, Date, DateTime, Percent, Rate, and Type will not validate against the MISMO Reference Model schema if they have blank values. In Version 3.4 and later an additional attribute, `xsi:nil="true"`, must be added to allow blank data values to be validated by the schema. The XML sample below shows how this attribute is implemented for a Type element.

```
<HMDAEthnicityType DataNotSuppliedReasonType="NotCollected" xsi:nil="true"/>
```

**NOTE:** The `xsi:nil` attribute is defined in the [http://www.w3.org/2001/XMLSchema-instance](http://www.w3.org/2001/XMLSchema-instance) namespace, so the XML Instance file must contain the following declaration in the root element or some other element:

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```
Display Time Zone Text Attribute

Beginning with Version 3.4.0, the DisplayTimeZoneText attribute may be added to MISMO Datetime element to identify the name of the time zone region that applies to the time value. The time value includes the offset in hours and minutes from universal time (UTC), but in some cases it is helpful to also identify the time zone text name.

For example, during the summer the displayed time and UTC offset in Phoenix (Mountain time zone) is the same as west coast cities (Pacific time zone), because Phoenix does not to switch to Daylight Savings Time. The DisplayTimeZoneText attribute in the example below identifies the time zone name associated with the time value.

```xml
<CreatedDatetime DisplayTimeZoneText="America/Phoenix">
  2015-06-08T15:20:14-07:00
</CreatedDatetime>
```

There are several standard lists of time zone names (IANA, Joda-Time, Olson, POSIX, etc.). The name used in the above example is from the Joda-Time list. Trading partners should agree on which time zone naming standard they will use.

**Container Elements**

Container elements are elements that can hold either related data elements or other container elements, never both.

The *root* or main container element of the MISMO V3 Schema is MESSAGE. It is the highest level in the MISMO architecture. The following image shows the MESSAGE container element, which holds ABOUT_VERSIONS and DEAL_SETS container elements.
Container elements with plural names always have a child element with the singular version of the name. For example:

- DEAL_SETS / DEAL_SET
- LOANS / LOAN
- PARTIES / PARTY
- INDEX_RULES / INDEX_RULE

In the example below the LOANS element has two LOAN elements. All repeating container elements have a SequenceNumber attribute, which may be used to identify the order of the two LOAN elements as shown in this XML example:

```xml
<DEAL>
  <LOANS>
    <LOAN SequenceNumber="1"/>
    <LOAN SequenceNumber="2"/>
  </LOANS>
</DEAL>
```

Every container element has an EXTENSION element that can be used to provide additional data not otherwise defined in the MISMO standard. In the following example, the MISMO BIRTH_INFORMATION element contains the mother’s maiden name, an existing MISMO data point, and is being extended to also include the grandmother’s maiden name, which is not a MISMO data point. The extended data is added to the EXTENSION / OTHER element structure.

```xml
<BIRTH_INFORMATION>
  <MothersMaidenName>Jennings</MothersMaidenName>
  <EXTENSION xmlns:abc="http://abcmortgage.com">
    <OTHER>
      <abc:GrandmothersMaidenName>Kelly</abc:GrandmothersMaidenName>
    </OTHER>
  </EXTENSION>
</BIRTH_INFORMATION>
```

See the “Extending the MISMO Standard” chapter of this I-Guide for more information on the use of the EXTENSION element.

Certain containers can appear in more than one location. These are sometimes referred to as “generic or reusable” containers. The container and its contents have a specific meaning that is further refined based on its location within the model. For example, the ADDRESS container element captures the address data element, street name, city name, state code, postal code, etc. When the ADDRESS container element appears in the PROPERTY container element, it represents a property address. When the ADDRESS
element is contained within an LIABILITY_HOLDER element, it represents the address of borrower’s creditor.

Cardinality

The cardinality of an element defines the number possible occurrences of that element. As discussed in the previous section, some data containers may be repeated multiple times within a MISMO data file. The MISMO Reference Model Schema defines the cardinality of each element. The attributes that specify cardinality are minOccurs (minimum number of elements) and maxOccurs (maximum number of elements). If the minOccurs value or maxOccurs value is not specified they have a default value of “1” (single occurrence).

The XML sample below is part of the MISMO Schema definition for the ADDRESSES container element. The ADDRESS child element is defined with a cardinality of minOccurs="0" and maxOccurs = “unbounded”, which indicates that ADDRESS is optional but may occur an unlimited number of times within ADDRESSES. The EXTENSION element has a cardinality of minOccurs="0", which indicates that it is optional and may only appear once (since maxOccurs is assumed to be “1” if it is not specified.

```xml
<xsd:complexType name="ADDRESSES">
  <xsd:sequence>
    <xsd:element name="ADDRESS" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="EXTENSION" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Data Relationships

Providing a list of mortgage industry data elements and their definitions is an important feature of the MISMO Reference Model. Equally important is how the Reference Model Schema defines the relationships between those data elements. Source and target data elements in a relationship are generally referred to as end points. The following types of relationships are supported in the Reference Model.

One-to-One Relationship

A One-to-One Relationship is one in which any given instance of the source element may only be associated with one instance of the target element, and no two instances of the source element are associated with the same instance of the target element.

For example: Each instance of the source PARTY / INDIVIDUAL element may be
associated with one instance of the target NAME element, which is not associated with any other instances of the PARTY / INDIVIDUAL element.

INDIVIDUAL 1  ➔  NAME 1
INDIVIDUAL 2  ➔  NAME 2

Most One-to-One Relationships are established in the MISMO Reference Model by containment. Containment means that there is a direct parent/child relationship between the elements. In the example above the INDIVIDUAL element contains a NAME element, which itself contains the individual’s name components – First Name, Middle Name, Last Name, and others. Other One-to-One Relationships may be needed which cannot be made using containment. For example, relating two data elements that are in different container elements. These types of relationships are sometimes referred to as “pointing relations” and are defined using “XLink Relationships”, which are discussed later in this chapter.

**One-to-Many Relationship**

A One-to-Many Relationship is one in which one instance of the source element may be associated with one or more instances of the target element and no two instances of the source element are associated with the same instance of the target element.

For example: Each instance of the source BORROWER element may have multiple instances of the target EMPLOYER element, each of those is not associated with any other instances of BORROWER. Even if both borrowers on a loan may have the same employer, the values of some the employer data elements may be different, such as their income amounts, start dates, and positions.

BORROWER 1  ➔  EMPLOYER 1
BORROWER 1  ➔  EMPLOYER 2
BORROWER 2  ➔  EMPLOYER 3
BORROWER 2  ➔  EMPLOYER 4

Most One-to-Many Relationships are also established in the MISMO Reference Model by containment. Other One-to-Many Relationships may be defined using “XLink Relationships”.

**Many-to-Many Relationship**

A Many-to-Many Relationship is one in which one instance of the source element may be associated with multiple instances of the target element and multiple instances of the source element may be associated with the same instance of the target element.
For example: Each BORROWER element may be associated with multiple ASSET elements, and multiple BORROWER elements may be associated with the same ASSET element.

![Diagram of BORROWER and ASSET relationships]

Many-to-Many Relationships are defined using “XLink Relationships”.

**Reflexive Relationship**

A Reflexive Relationship is a one-to-one, one-to-many, or many-to-many relationship between instances of the same element, i.e., the source and target elements are of the same type.

For example: An instance of the INDIVIDUAL element may be associated with another instance of the INDIVIDUAL element to represent a marriage relationship.

Reflexive Relationships are defined using “XLink Relationships”.

**Irreflexive Relationship**

An Irreflexive Relationship is the opposite of a Reflexive Relationship. It is a one-to-one, one-to-many, or many-to-many relationship between instances of different elements, i.e., the source and target elements are of different types.

For example: The One-to-One INDIVIDUAL to NAME relationship, One-to-Many BORROWER to EMPLOYER relationship, and Many-to-Many BORROWER to ASSET relationships discussed earlier are all examples of Irreflexive Relationships.

**Rules for XLink Relationships**

**Overview**

The previous section identified types of relationships and the methods MISMO uses to define relationships – either by containment or by pointing. Whenever we need to identify relationships between elements that are not directly contained by another, we use Xlink Relationships. MISMO has added the appropriately named RELATIONSHIPS container structure as a means of specifying those relationships.

The RELATIONSHIPS container is a child of MESSAGE, DEAL_SETS, DEAL_SET, DEALS, DEAL, and DOCUMENT. The location in the Reference Model of the RELATIONSHIPS container is determined by the XML instance and its structure. The RELATIONSHIPS container used MUST be at the lowest structural level that is a parent or sibling of the elements being joined.
Examples include:

- The RELATIONSHIPS container as a child of a DEAL_SETS container links DEAL_SETS/PARTY data to DEAL_SETS/DEAL_SET/DEALS/DEAL/LOANS/LOAN data.

- The RELATIONSHIPS container as a child of a DOCUMENT container links all the information about document signatures in the DOCUMENT/SIGNATORIES/SIGNATORY container to the actual DOCUMENT/DEAL_SETS/DEAL_SET/DEALS/DEAL/PARTY container representing the person that signed the document.

- The RELATIONSHIPS container as a child of a DEAL container links all the information for a DEAL/LOANS/LOAN, such as asset and liability information, to individual borrowers under DEAL/PARTIES/PARTY container.

The MISMO Reference Model uses an XML specification called XLink to define data relationships in a MISMO message that are not naturally expressed by the MISMO container hierarchy. The full XLink standard is defined by the W3C at [http://www.w3.org/TR/xlink](http://www.w3.org/TR/xlink). At this time MISMO is only using a limited set of the XLink attributes.

One of the XLink attributes, arcrole describes the type of relationship between source data and target data. The XLink from attribute identifies the source data point or container. The XLink to attribute identifies the target data point or container. The XLink label attribute is an identifier that is attached to data point or container elements that will be used to express an arcrole relationship. The MISMO RELATIONSHIP container elements hold the XLink arcrole, from and to attributes that describe the data relationships.

In the simplified example below, a CREDIT_RESPONSE has returned CREDIT_SCORE data for BORROWER parties in a loan DEAL. XLink label attributes are added to each CREDIT_SCORE element and each PARTY element’s ROLE element. The MISMO RELATIONSHIP element identifies which credit score is associated with each borrower.
RELATIONSHIP Element Types “From” and “To” End Point Conditions

Each “from” and “to” attribute value in a RELATIONSHIP element must satisfy the following conditions:

- **Sufficiency** – Each endpoint must unambiguously represent the data that is being linked in the relationship.
- **Uniqueness** – Each instance of endpoint must represent a different instance of the relationship.

For example, to represent the relationship where a party is a borrower on a loan, link the elements LOAN and ROLE. That relationship meets the rules of **sufficiency** and **uniqueness**.
LOAN is sufficient to represent an instance of a loan because all the elements under LOAN work together to describe the same loan. If we go any higher in the model, we would fail to identify a loan. We could not use the DEAL element, because it may contain multiple loans and we would not know to which one of them the relationship is being applied.

LOAN is unique because it represents one specific instance of the LOAN container. Any other sibling instance would be a different loan. If we go any lower, the endpoint would not be unique. For example, we could not use LOAN_IDENTIFIER because all sibling occurrences of LOAN_IDENTIFIER refer to the same loan.

ROLE is sufficient to identify a party as a borrower (by setting the ROLE_DETAIL element’s PartyRoleType attribute value to “Borrower”). If we go any higher in the model, we would fail to identify the role. For example, we could not use the PARTY element, because it may contain multiple roles and we would not know to which of them the relationship is being applied.

The ROLE element is unique. Any given party will only have one ROLE element of type borrower. If we want to say that a party is a borrower on a loan, we need to use one specific instance of the ROLE element. Pointing to any other sibling instance would mean a different role. If we go any lower, the endpoint would not be unique. For example, we could not use EMPLOYER because all sibling instances of EMPLOYER refer to the same borrower.

RELATIONSHIP Element Attributes: from, to and arcrole

Each instance of the RELATIONSHIP element must contain at least three XLink attributes: from, to, and arcrole. Since these attributes are part of the XLink standard’s “namespace”, in a XML message they must contain the namespace prefix that the MISMO schema assigns for them. By default the schema usually assigns xlink: as the namespace prefix, to differentiate these XLink attributes from MISMO data elements and attributes.

Example:

```xml
```

RELATIONSHIP Element “Arcrole” Naming Convention

The architectural rules for the use of XLink are detailed in MEG 0036, Use of XLink to Manage Relationships, available at the link below:


Each arcrole value will be composed as follows:

```
{URN} {FromName} ‘_’ {VerbPhrase} ‘_’ {ToName}
```
• **URN** – for arcroles defined by MISMO, the URN string value is `urn:fdc:mismo.org:2009:residential`/ For arcroles not defined by MISMO the “extended” URN is the string “urn:fdc:” + domain name of the extending organization + “:” + year of first use + “:” + optional qualifier + separator character “:” or “/”. Example: `urn:fdc:AcmeLending.com:2013:mortgage`

• **FromName** – the element name that is pointed to by xlink:from attribute.

• **VerbPhrase** – is defined according to the type of relationship. For Irreflexive Relationships (i.e. FromName is different from ToName), the value of VerbPhrase is ‘IsAssociatedWith’.

• **ToName** – the element name that is pointed to by the xlink:to attribute.

Example of a “MISMO V3.3.0” arcrole value:

```
xlink:arcrole="urn:fdc:mismo.org:2009:residential/ASSET_IsAssociatedWith_ROLE"
```

Example of an “extended” arcrole value:

```
xlink:arcrole="urn:fdc:AcmeLending.com:2013:mortgage/PARTY_IsSubsidiaryOf_PARTY"
```

**RELATIONSHIP Element “Arcrole” FromName/ToName Value Order**

When the relationship is connecting two container elements, the arcrole value used for the FromName must precede the value of the ToName alphabetically.

For example, if a relationship is being established between ROLE and LOAN elements, LOAN must be the FromName and ROLE must be the ToName, since LOAN precedes ROLE alphabetically.


Invalid arcrole value: "urn:fdc:mismo.org:2009:residential/ROLE_IsAssociatedWith_LOAN"

When the relationship is connecting a container element to a data point element, the FromName is always the container name and the ToName is always the data point element name.

For example: "urn:fdc:mismo.org:2009:residential/DATA_SOURCE_IsAssociatedWith_GrossLivingAreaSquareFeetNumber"
CHAPTER 4: MAJOR ELEMENTS OF THE SCHEMA

This chapter covers the major elements in the MISMO Version 3 schema. The beginning sections discuss the four root elements MESSAGE, DEAL¹, DOCUMENT and PARTY. The last section focuses on the LOAN element and related concepts: Loan Role, Loan State, and structures storing data related to Index Rules, Adjustment Rules, and Interest Rate or Payment changes.

MESSAGE Element

The MESSAGE element is the root, or primary, element of the MISMO message.² This image shows the top two levels of the MISMO MESSAGE data structure. The second-level element DEAL_SETS holds sets of loan data, or deals. The DOCUMENT_SETS element includes printable/viewable documents as well as the related data that populates the documents.

¹ Even though the DEAL element is listed in the Reference Model schema as one of the three “root” elements, MESSAGE is always used as the root element when exchanging data between business partners. However, DEAL or DOCUMENT can be used as a “root” element of an XML instance document when using it internally for analysis, reporting or archiving.

² The data captured in MESSAGE data set can be the message payload of any SOAP or REST transaction. In a SOAP transaction, MESSAGE is the content of the SOAP body element. In a REST transaction, MESSAGE is the document that is in the PUT, GET or DELETE HTTP protocol methods.
In general, the DEAL_SETS container, as the only child of MESSAGE, is used for data-centric exchanges and the DOCUMENT_SETS container, as the only child of MESSAGE, is used for document-centric exchanges. There are certain cases where related documents are included in a data centric message. In these cases, there may be both a DEAL_SETS element and a DOCUMENT_SETS element as children of a MESSAGE container; in these cases RELATIONSHIP containers are used link the data and associated documents. An example is a data-centric service request for underwriting that would have attached the documents for the borrowers’ authorizations to obtain data from the IRS.

The RELATIONSHIPS element holds the XLink attributes and the SYSTEM_SIGNATURES element enables the use of W3C XML digital signatures for creating tamper-evident electronic seals around the MESSAGE and the DOCUMENT elements.

**MESSAGE Attributes**

Every MISMO data element can have additional XML “attributes” that either provide additional identifiers for the element, or that further describe an element’s usage. The first two attributes of MESSAGE identify the MISMO Reference Model version (example: 3.3.0) and Logical Data Dictionary version (example 3.3.0.0) used in a particular MISMO XML message instance.
The diagram above like others in this guide were generated from the MISMO XML Schema using an XML-aware editor. Note that it includes the definitions for each data element and attribute. These are the same as the definitions from MISMO Logical Data Dictionary.

**ABOUT_VERSIONS / ABOUT_VERSION Elements**

In the MISMO Reference Model, the ABOUT_VERSION elements are defined as child elements of the MESSAGE, DEAL and DOCUMENTS elements. How ABOUT_VERSION is used is dependent on its location.

When used within the MESSAGE element, ABOUT_VERSION captures the version number of the software, platform, or specification used to generate the message payload and the date and time that the message was generated.

**DEAL_SETS Element**

DEAL_SETS contains information about one or more DEAL_SET elements. The DEAL_SETS element may be a child of the MESSAGE or the DOCUMENT element. When the DEAL_SETS is a direct child of MESSAGE, each DEAL_SET can be used for transmitting a group of related loans such as set of loans being transferred to or from a loan servicer or other entity, or a pool of loans that forms a mortgage-backed security (MBS).
When the DEAL_SETS container is a child of a DOCUMENT element, there are two uses. A single DEAL_SET may be used to represent the data of the document. Alternatively, multiple DEAL_SET containers may be used to represent a group of data on a single document (such as a report on a pool of loans). The DEAL_SET_SERVICE elements contain information about services related to deal sets, such as servicing transfers, workout evaluations, loan registrations or securities rating services.

PARTY elements list information about the business entities or individuals providing or receiving services related to deal sets. RELATIONSHIP elements identify how data points in the DEAL_SETS structures are related to each other. VERIFICATION_DATA contains information that may be used as a cross check to verify the full content of the DEAL_SET is present, such as a Loan Total Count or to summarize the content such as Escrow Balance Total Amount.
**DOCUMENT_SETS Element**

Individual related DOCUMENT elements are grouped together as a DOCUMENT_SET. DOCUMENT_SETS can hold one or more DOCUMENT_SET elements. See the “DOCUMENT Element” section of this chapter for more information on its use.

When common document data or image files are being provided for multiple documents in a non-MISMO or foreign format, the FOREIGN_OBJECTS element is used to either contain the shared content directly or identify the location of the document on a server or web site.
RELATIONSHIPS / RELATIONSHIP Elements
The RELATIONSHIP elements in the MESSAGE element link DEAL_SET elements to DOCUMENT_SET elements. The use of the RELATIONSHIPS element was covered earlier in this guide in the “Rules for XLink Relationships” section of the “Reference Model – XML Schema” chapter.

SYSTEM_SIGNATURES / SYSTEM_SIGNATURE Elements
The SYSTEM_SIGNATURES element holds electronic signatures that provide a tamper seal of one or more parts of a MISMO message. The tamper seal provides a way to detect if a specified data section within the MISMO message has been changed since the date and time that the electronic seal was originally created. The SYSTEM_SIGNATURE element uses a security technology called the X.509 digital certificate. More information is provided in the XML Digital Signature Implementation Guide on the MISMO web site (www.mismo.org).

DEAL Element
The DEAL element captures information for a single mortgage transaction. A deal may involve a single loan or multiple loans, such as a first lien refinance of a mortgage and a second lien home equity line of credit.

Even though the DEAL element is listed in the Reference Model schema as one of the three root elements, MESSAGE is always used as the root element when exchanging data between business partners. However DEAL can be used as a root element of an XML instance document when using it internally for analysis, reporting or archiving.

On the next page is a diagram of the DEAL data structure. The elements contained by DEAL can hold information about the assets, collaterals, expenses, liabilities, loans, parties (borrowers, brokers, closing agents, etc.), and services (appraisals, credit reports, title, etc.) related to the mortgage deal.

DEAL Attributes
When the DEAL element is used as the root element of an XML instance, its first two attributes are used to identify the MISMO Reference Model version (example: 3.3.0) and Logical Data Dictionary version (example 3.3.0.0) that was used to define the data and structures in a particular MISMO XML message instance.
The root element for all
interactions that apply to the
use cases of a single deal. A
Deal may contain multiple
loans or loan applications.

DEAL

VERSIO NS 3.4 & LATER

This diagram outlines the Major Elements of the Schema in MISMO V3 General Information Guide.
REFERENCE Element

Normally DEAL data is included directly in its child elements, ASSETS, COLLATERALS, LOANS, and so on. In some cases it may be necessary or preferred to refer to data related to a deal by referencing a server location or web site address using a URL (Universal Reference Locator). The data related to this option is stored in the REFERENCE element.

For an individual DEAL instance, either a REFERENCE can be used or the normal ASSETS, COLLATERALS, LOANS, etc. set of elements can be used, not both. In the XML Schema this is referred to as a choice group.

ABOUT_VERSIONS / ABOUT_VERSION Elements

The ABOUT_VERSION element is available for use as a child element of the MESSAGE, DEAL and DOCUMENTS elements. How ABOUT_VERSION is used is dependent on its location.

When used within the DEAL element, ABOUT_VERSION provides information about the software version or entity-specific specification that is generating the DEAL data, plus the date and time that the DEAL data was generated.

Here are a few examples of how ABOUT_VERSION can be used within DEAL:

- In the Fannie Mae and Freddie Mac Uniform Loan Delivery Data (ULDD) specification, the ABOUT_VERSION element is used to identify the version number of the ULDD Implementation used to create the loan delivery XML file.
- A Loan Origination System (LOS) software platform used by a lender can use ABOUT_VERSION to identify the LOS vendor and the software version.
- A service provider like a credit bureau can use ABOUT_VERSION to identify which version of their software created the credit report data.
**ASSETS Element**

ASSETS captures data about a collection of assets that may be used to determine the credit-worthiness of the borrowers. Each child ASSET element captures information about a single asset, including its cash or market value, its description, the asset holder, and verification documentation related to the asset. Assets can be associated with a borrower using the RELATIONSHIP elements. Relating assets to a borrower is probably the most common usage. However, the model supports the ability to relate an asset to any other party specified by the Party Role Type.

For owned property assets, additional information may be collected such as lien amounts, rental income, maintenance expenses, etc. In V3.0 and V3.1 the ASSETS container could hold either ASSET elements or OWNED_PROPERTIES elements as shown in the diagram above.
In V3.2 and later, ASSETS was restructured to only contain ASSET elements. If the asset was an owned property, the additional property information would be contained in the OWNED_PROPERTY child element of ASSET.

**COLLATERALS Element**

The COLLATERALS element captures data about a collection of assets that are designated as collateral for the loans of this deal. For most mortgage loans the primary collateral used to secure the loan is the subject property, but this structure has the flexibility to support other collateral arrangements.

In all MISMO versions, the COLLATERALS element can contain one or more COLLATERAL elements. The two diagrams below highlight significant structure change between the V3.0 and V3.1 COLLATERAL structure and one used in V3.2 and later.
In V3.0 and V3.1 a single COLLATERAL element could contain data about the properties that are the subject of the loan, plus multiple other assets or owned properties being pledged as collateral for the deal. This structure is represented in the diagram shown below.
In V3.2 and later, a single COLLATERAL element could contain either data about the subject property or data about other pledged assets such as owned property or another type of asset as depicted in the diagram below. In V3.2 each COLLATERAL record only contains data about a single asset or property.

NOTE: The property and valuation data structures are being covered in more detail in an Implementation Guide being prepared by the Property Information & Valuation Services Workgroup. When it is completed it will be available on the “Implementation Guidelines” page of the MISMO web site (www.mismo.org).
COMMUNICATION EVENTS / COMMUNICATION EVENT Elements

These elements were added in V3.4.0 to collect information about one or more communications related to a loan deal. This structure captures and tracks data about issues, their follow-up and their resolution and the participants involved in the communication event.

DEAL DETAIL Elements

The DEAL DETAIL element was added in Version 3.4.0 to capture data related to a loan deal, such as whether the deal is eligible to be considered a “First Time Home Buyer” category.

EXPENSES / EXPENSE Elements

The EXPENSES element captures a group of expenses. Each EXPENSE element captures the type of expense, such as alimony, child support, separate maintenance, job-related (child care, union dues, etc.) expenses and the monthly amount of the expense.
LIABILITIES / LIABILITY Elements

The LIABILITIES element holds groups of liabilities. Each child LIABILITY element holds creditor name and address, account number, monthly payment amount, unpaid balance and other data for an individual liability. Like assets, the liabilities are most commonly related to one or more borrowers but may be linked to any other party identified by the Party Role Type.

LOANS Element

This element holds a one or more LOAN elements that are part of a mortgage deal between a mortgage lender and one or more borrowers. More detailed coverage of the contents of the LOAN element appears later in this chapter.

The LOANS element also has a COMBINED_LTVS, which can contain one or more records of the calculated combined loan to value (LTV) ratio percentages calculated at different times during the life of the loan. For example: If there are 2 loans, a first and second, on a single property. The individually calculated LTV (the ratio of the amount of the loan to the property value) for each loan would be carried within an instance of the LOAN container. The COMBINED_LTV container would carry the combined LTV (the ratio of the sum of both loans to the property value). The COMBINED_LTV element can also capture the Home Equity Combined LTV.
PARTIES / PARTY Elements

The PARTY elements provide information about the various parties related to this DEAL data set. The parties can include the borrower(s), seller, lender, broker, closing agent, title agent, etc. There are multiple locations in the Reference Model for PARTY elements. More detailed information about the PARTY element and which location in the Reference Model to use for a given purpose is provided later in this chapter.

RELATIONSHIPS / RELATIONSHIP Elements

The RELATIONSHIP elements directly under the DEAL element are used to identify data relationships within an individual DEAL data set. For example, it could be used to identify which borrowers are associated with a particular asset, liability or expense. The use of the RELATIONSHIPS element was covered earlier in this guide in the “Rules for XLink Relationships” section of the Reference Model – XML Schema chapter.
SERVICES / SERVICE Elements

The SERVICES element contains data for mortgage services related to a DEAL data set. The following diagram of the SERVICE element shows the services available in MISMO V3.3.0, plus new services and elements that were added in V3.3.1 and V3.4.0.
SUPPORTING RECORD SETS / SUPPORTING RECORD SET Elements

The SUPPORTING RECORD SET elements were added in V3.4.0 and contain data collected for a purpose such as evidence of compliance with a regulation.

DOCUMENT Element

The DOCUMENT element contains an electronic file that reports one or more aspects of a business transaction. It can contain the image of a document as well as information about the document and the data represented on the document.

Document Concepts

A document in electronic-record form is an eye-readable file. It has several functions:

- When a document is signed, it can show that the signer acknowledges its contents.
- When the document is a contract it can bind the signer to the terms of that contract, and the document becomes evidence to other readers that the signer agreed to those terms.
- Because electronic documents can be easily altered, there are special processes to prove that a document has not been altered after it was signed.
- A set of documents together can provide evidence that the proper processes have been carried out in a mortgage transaction, and that all parties agreed to the same set of facts.
**DOCUMENT Attributes**

Similar to the MESSAGE and DEAL elements, the first two attributes of the DOCUMENT element identify the MISMO Reference Model version (example: 3.3.0) and Logical Data Dictionary version (example 3.3.0.0) that define the data and structures in a particular MISMO XML MESSAGE element.

**REFERENCE Element**

Instead of containing a document and its data directly, a DOCUMENT element may just identify the external location of the document. This is done using the REFERENCE / LocationURL element which contains data points that “refer” to the document’s location by referencing a server location or web site address using a URL (Universal Reference Locator), as shown in the XML sample below.
For an individual DOCUMENT instance, **either a REFERENCE can be used or the normal DEAL_SETS, MAP, VIEWS, etc. set of elements can be used, not both.**

**UNKNOWN_VERSION3_DOCUMENT Element**

The UNKNOWN_VERSION3_DOCUMENT element captures document content that is not part of the namespace or version of the MISMO Version 3 schema. This document is either contained directly (using this structure’s EmbeddedContent XML element), or indirectly (using this structure’s ObjectURL element).

**AUDIT_TRAIL Element**

The AUDIT_TRAIL element contains entries that show the document’s history, including corrections, additions, signings, validation, voiding, and editing of the document, including who edited the document, date and time of the edit, and the system that it was edited on.

Each entry should have a corresponding SYSTEM_SIGNATURE element to create a digital tamper-evident seal over the audit entry as well as the affected portions of the document.

More information about its implementation is provided in implementation guides in the eMortgage Specifications section of the MISMO web site (www.mismo.org).

**DEAL_SETS Element**

A DOCUMENT element can contain a DEAL_SETS element that captures data related to the instance of DOCUMENT. If a DEAL_SETS element is present, it must contain the document’s data and must be a child of the DOCUMENT element. Only the data elements contained in the document need to appear in the DEAL_SETS structure however additional data that is not viewable on the document may be included. When tamper-sealed with a digital signature, a document can provide both eye-readable and machine-readable evidence of its history.

For example:

- If the DOCUMENT element holds a closing document, then the child elements DEAL and LOAN of DEAL_SETS element are populated.
- If the document is a credit report, then the child elements of DEAL, SERVICE and CREDIT_RESPONSE of the DEAL_SETS element are populated.
- If the document involves a mortgage-backed security covering a pool of loans, then the child elements of the DEAL_SET element are populated.
**MAP Element**

A MAP element contains or points to style sheets for generating or populating document VIEWS, based on the contents of the DEAL_SETS data and/or the SIGNATORIES data. The MAP data can verify that the data in the target VIEW element matches the data provided in the DEAL_SETS and SIGNATORY elements.

More information about its implementation is provided in implementation guides in the eMortgage Specifications section of the MISMO web site (www.mismo.org).

**RELATIONSHIPS / RELATIONSHIP Elements**

The RELATIONSHIP elements directly under the DOCUMENT element identify data relationships within a DOCUMENT data set. For example, RELATIONSHIP can identify which signatories are associated with a particular document view or link a signatory to a party listed in the DEAL_SET data. The use of the RELATIONSHIPS element was covered earlier in this guide in the “Rules for XLink Relationships” section of the Reference Model – XML Schema chapter.

**SIGNATORIES / SIGNATORY Elements**

Each SIGNATORY element contains information regarding the type of electronic signature used to represent a legally binding signature of a specific party named in the document. Other data within this element may contain more detailed data about the signing event, such as its location, date and time as well as information about the notary certificate used in the notarial act that the notary memorializes in the document.

**SYSTEM_SIGNATURES / SYSTEM_SIGNATURE Elements**

This element is used to hold electronic signatures that represent a “tamper seal” of one or more parts of a DOCUMENT instance. The “tamper seal” provides a way to detect if a specified data section within the document has been changed since the date and time that the electronic seal was originally created. The SYSTEM_SIGNATURE element uses a security technology called the X.509 digital certificate. More information about its implementation is provided in a “Digital Signatures” document in the eMortgage Specifications section of the MISMO web site (www.mismo.org).

**VIEWS / VIEW Elements**

This section of a DOCUMENT instance contains the actual “views” of the document that would appear on screen or a printed output. These contain what is necessary to build one or more visual representations of this DOCUMENT. Each DOCUMENT element must include at least one VIEW element. Views can represent different methods of representing the data, or they can represent snapshots of specific points in time, in various stages of completion. A common example of two “views” are before and after a document is signed.
Depending on the type of document, the views can either be organized as VIEW_FILES or VIEW_PAGES. There are additional data elements that describe areas of a “view” that contain interactive fields, a notary signature, recording endorsement, stakeholder signatures or witness signatures.

More information about its implementation is provided in implementation guides in the eMortgage Specifications section of the MISMO web site (www.mismo.org).

ABOUT_VERSIONS / ABOUT_VERSION Elements

In the MISMO Reference Model, the ABOUT_VERSION elements are defined as child elements of the MESSAGE, DEAL and DOCUMENTS elements. How ABOUT_VERSION is used is dependent on its location.

When used within the DOCUMENT element, ABOUT_VERSION is used to provide version information about the software version and/or entity-specific specification that is generating the DOCUMENT data, such as the date-time the DOCUMENT was generated.

DOCUMENT_CLASSIFICATION Element

Data elements in this section contain information about a specific instance of the document that describes its type and also its use, as opposed to its contents. Because documents can serve such critical functions, it is important to consistently identify and classify each document in a set so that it can be easily found and used at any point in the future. It is further subdivided as follows.

- DOCUMENT_CLASSES - One or more specific document types that differentiate this document from other document types. There are multiple occurrences because a single readable document image may be classified in more than one way, and it is useful to search for a single document by any of the listed types.
- DOCUMENT_CLASSIFICATION_DETAIL – Contains the non-repeatable data about an instance of a document such as the publisher and version of the document.
- DOCUMENT_USAGES - Apart from being classified, a document may be used in multiple ways and multiple processes, as defined by trading partners who are sending and receiving sets of documents. NOTE: DOCUMENT_USAGES is only present in MISMO Versions 3.1 and later.

NOTE: A separate Document Classification I-Guide is being produced that provides more detail on this topic. It will be available on the MISMO web site (www.mismo.org).

Document Profiles

Use of the DOCUMENT element is defined by profiles, which determine how and when the containers described above are used. The profiles help to efficiently manage data exchange between multiple internal and external computer systems – while also preserving the human need to read, understand and analyze the information being
exchanged.

The three MISMO v3 profiles are:

**Basic:** The document receiver only needs the view, information related to signatures (if the document is signed) and the document type or name.

**Retrievable:** The receiver wants to access data from the document and there is not a requirement to automatically verify that the data matches what is presented in the viewable image. The data in the Retrievable profile may or may not include all of the information contained in the view.

**Verifiable:** The document receiver wants to verify that the data matches the information contained in the view. The data in the Verifiable profile must include all of the information included in the view.

Each profile determines which sections of the MISMO v3 DOCUMENT model are expected to be in the document. This allows computer systems to be programmed to interpret and read the various potential sections of the document. The profiles become more extensive, from Basic to Verifiable.

A summary of the sections used by each profile and MISMO V3 Reference Model requirements for each profile are provided in the table below. Optional means the section may or may not be present. Required sections must be present and prohibited sections must not be present. Conditional sections may be present depending on the presence of other sections. For instance, the Audit Trail Section in the Basic and Retrievable profiles is required if a Computer System Signature is present.

<table>
<thead>
<tr>
<th>Document Sections in the MISMO v3 DOCUMENT model</th>
<th>Basic</th>
<th>Retrievable</th>
<th>Verifiable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>About Versions:</strong> Identify the SMART document profile.</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td><strong>Audit Trail:</strong> Logs events that occur during the lifecycle of the document.</td>
<td>Conditional</td>
<td>Conditional</td>
<td>Required</td>
</tr>
<tr>
<td><strong>Deal Sets:</strong> Provides the data for the document.</td>
<td>Conditional</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td><strong>Document Classification:</strong> Describes the types, purposes and uses of the document including the document name.</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td><strong>Relationships:</strong> Provides associations between the different document sections.</td>
<td>Optional</td>
<td>Optional</td>
<td>Conditional</td>
</tr>
<tr>
<td><strong>Signatories:</strong> Provides data about executed document signatures, electronic or otherwise.</td>
<td>Optional</td>
<td>Conditional</td>
<td>Conditional</td>
</tr>
</tbody>
</table>
System Signature: Provides digital signatures over some or the entire document to provide proof that no tampering has occurred.

<table>
<thead>
<tr>
<th></th>
<th>Optional</th>
<th>Optional</th>
<th>Required</th>
</tr>
</thead>
</table>

Templates or Mapping: Relates data from other sections of the document to the viewable image.

<table>
<thead>
<tr>
<th></th>
<th>Prohibited</th>
<th>Prohibited</th>
<th>Required</th>
</tr>
</thead>
</table>

View: Provides the visual representation of the document.

<table>
<thead>
<tr>
<th></th>
<th>Required</th>
<th>Required</th>
<th>Required</th>
</tr>
</thead>
</table>

1 If a digital signature is provided in the SYSTEM_SIGNATURE element, then there MUST be at least one entry in AUDIT_TRAIL_ENTRIES. Otherwise, the use of the AUDIT_TRAIL_ENTRIES container is OPTIONAL.

2 If information about signatures for stakeholders, notaries, and/or witnesses is provided in the SIGNATORIES container, then the DEAL_SETS container MAY be used to convey the signer’s information in a corresponding PARTY element.

3 If the document has signatures for stakeholders, notaries, and witnesses, then the information about those signatures MUST be provided in the SIGNATORIES container.

4 If the document has signatures or signature placeholders, then the RELATIONSHIPS container MUST be used to link the SIGNATORY element and/or the VIEW_FIELD container to the corresponding PARTY container.

PARTY Element

The PARTY container element collects information about a single party that is involved in the mortgage related transaction. There must be a separate PARTY container for each party for whom information is required.

The MISMO Reference Model allows collections of PARTY elements to be documented at several locations within the Reference Model to support different types of transactions. For example, if data related to pools of loans are being transmitted then the PARTIES associated with pool-level data can be included at either the DEAL_SETS or DEAL SET level. If data and documents related to pools of loans are being transmitted then the PARTIES associated with pool-level data and documents should be included at the MESSAGE level to allow for linking between the pool level data and the individual documents in the DOCUMENT_SETS container.

Parties involved with an individual mortgage deal would be listed in the PARTIES container structure at the DEALS or DEAL level. Parties related to an individual document would be listed in the PARTIES container structure within the DEALS or DEAL level as a child of DOCUMENT.

Parties related to mortgage services are generally included in a PARTIES element within those transaction structures (i.e. CREDIT_RESPONSE, FLOOD_RESPONSE, MI_VALIDATION_RESPONSE, TITLE_RESPONSE, and VALUATION_RESPONSE). Parties represented on a document are generally included in a PARTIES element within the DEAL

---

49
level under the DOCUMENT element. In general, the PARTY elements should be used at
the lowest level in the Reference Model at which the entities to which they are linked by
RELATIONSHIPS are all contained.

In the MISMO Reference Model, it is possible for each PARTY to play multiple roles. A
Party may be either an INDIVIDUAL or a LEGAL_ENTITY. Because many entities and
individuals play a role in the loan origination, servicing, and delivery processes, multiple
PARTY containers will likely be delivered for each DEAL - for example, Borrower,
Appraiser, Appraiser Supervisor, Loan Originator, and Loan Origination Company. If
there is a Primary Borrower and a Co-Borrower, a separate PARTY container would be
sent for each of them.

Whether a party is an INDIVIDUAL or a LEGAL_ENTITY, the PARTY element also provides
ADDRESS, ROLE and TAXPAYER_IDENTIFIER containers to hold this type of data.

**REFERENCE Element**

Normally the PARTY data is included directly in its child elements, INDIVIDUAL or a
LEGALENTITY, and the ADDRESS, ROLE and TAXPAYER_IDENTIFIER elements. In some
cases it may be necessary or preferred to refer to party data in another part of the
message or at an external server location or web site address. The data related to this
option is stored in the REFERENCE element. For an individual PARTY instance, either a
REFERENCE can be used or the normal INDIVIDUAL or LEGALENTITY, etc. set of
elements can be used, not both. In the XML Schema this is referred to as a choice
group.

When the PARTY data is being referenced in another location within the same XML
document, the LocationLabelValue element is used to identify the XPath and the label
value assigned to the PARTY element record that contains the full set of data.
The following XML PARTY element samples demonstrate the use the REFERENCE element. This sample shows a PARTY element that is located within a DEAL element. Note that the PARTY element’s xlink:label attribute contains a label value.

```
<DEAL>
  <PARTIES>
    <PARTY xlink:label="Party-0001">
      <LEGAL_ENTITY>
        <LEGAL_ENTITY_DETAIL>
          <FullName>ABC Mortgage</FullName>
        </LEGAL_ENTITY_DETAIL>
      </LEGAL_ENTITY>
    </PARTY>
  </PARTIES>
</DEAL>
```

In some use cases the same party data may need to be used within the MESSAGE / DEAL_SETS element to identify the requesting party. Rather than repeating the party data, the PARTY element within DEAL_SETS can contain a REFERENCE element that points to the PARTY element above within the DEAL element.

```
<DEAL_SETS>
  <PARTIES>
    <PARTY>
      <REFERENCE>
        <LocationLabelValue>/MESSAGE/DEALSETS/DEAL_SET/DEALS/DEAL/PARTIES/PARTY[@xlink:label = 'Party-0001']</LocationLabelValue>
      </REFERENCE>
    </PARTY>
  </PARTIES>
</DEAL_SETS>
```

**INDIVIDUAL Element**

The INDIVIDUAL container element is used to provide the type of information that is appropriate for a Party that is an individual as opposed to a legal entity. The person’s name and contact points (phone numbers, email addresses, etc.) as well as aliases (previous names such as a maiden name). Under the IDENTIFICATION_VERIFICATION element, the BIRTH_INFORMATION element is used to record the city, state, province or country of the individual’s birth. The IDENTITY_DOCUMENTATION element is available to record information about documents that identify a person such as driver’s license or a passport.
LEGAL_ENTITY Element

This element holds information specific to a legal entity, such as the company name, “Doing Business As” names, and contact information for individuals within a company. The LEGAL_ENTITY_DETAIL element can also store the GlobalLegalEntityIdentifier. See the About the Legal Entity Identifier section of this guide for more information.

If an individual within a company has a key mortgage transaction role, such as a loan officer, broker, appraiser, etc., these individuals should also have their own PARTY records with their INDIVIDUAL information and should use a relationship to link the individual to the legal entity.

In general, it is not recommended to solely use the CONTACTS container to represent individuals of a legal entity, as the purpose of the container is to describe methods of communication and identify individuals that may be contacted for a legal entity. It is recommended to create PARTY containers for each individual that is associated with the legal entity, then connect the two parties using a RELATIONSHIP element using the appropriate arcrole (see Data Relationships section of this guide for more information).
ROLES / ROLE Elements

The MISMO Reference model allows one or more roles to be defined for an individual or a legal entity, using the ROLE / ROLE_DETAIL / PartyRoleType data point element. For some roles, the MISMO Reference Model provide additional container elements to capture data specific to that role type.

For example, for the PartyRoleType value of “Appraiser”, the example below shows a ROLE / APPRAISER container element to identify the appraiser’s company name, their license type and the detailed license information.

```xml
<PARTY xlink:label="Appraiser-1">
    <INDIVIDUAL>
        <NAME>
            <FirstName>Arnold</FirstName>
            <LastName>Smith</LastName>
            <MiddleName>P</MiddleName>
        </NAME>
    </INDIVIDUAL>
    <ROLES>
        <ROLE>
            <APPRAISER>
                <APPRAISER_DETAIL>
                    <AppraiserCompanyName>
                        Acme Appraisers Inc.
                    </AppraiserCompanyName>
                </APPRAISER_DETAIL>
            </APPRAISER>
            <LICENSES>
                <LICENSE>
                    <APPRAISER_LICENSE>
                        <AppraiserLicenseType>
                            LicensedResidentialAppraiser
                        </AppraiserLicenseType>
                    </APPRAISER_LICENSE>
                </LICENSE>
                <LICENSE_DETAIL>
                    <LicenseIdentifier IdentifierOwnerURI="www.grec.state.ga.us">279249</LicenseIdentifier>
                    <LicenseIssuingAuthorityName>
                        Georgia Real Estate Appraisers Board
                    </LicenseIssuingAuthorityName>
                    <LicenseIssuingAuthorityStateCode>GA</LicenseIssuingAuthorityStateCode>
                </LICENSE_DETAIL>
            </LICENSE>
        </ROLE>
    </ROLES>
</PARTY>
```
For the PartyRoleType of “Borrower”, there is a ROLE / BORROWER container structure for holding other borrower data that may be used during the qualification process. The sample XML below depicts a V3.3.0 structure for the Borrower Party.

```xml
<PARTY SequenceNumber="1" xlink:label="Borrower-1">
  <INDIVIDUAL>
    <NAME>
      <FirstName>JONATHAN</FirstName>
      <LastName>CONSUMER</LastName>
    </NAME>
  </INDIVIDUAL>
  <ROLES>
    <ROLE>
      <BORROWER>
        <BORROWER_DETAIL>
          <BorrowerBirthDate>1983-02-15</BorrowerBirthDate>
          <BorrowerClassificationType>Primary</BorrowerClassificationType>
          <MaritalStatusType>Married</MaritalStatusType>
        </BORROWER_DETAIL>
        <DECLARATION>
          <DECLARATION_DETAIL>
            <CitizenshipResidencyType>USCitizen</CitizenshipResidencyType>
            <IntentToOccupyType>Yes</IntentToOccupyType>
          </DECLARATION_DETAIL>
        </DECLARATION>
        <GOVERNMENT_MONITORING>
          <GOVERNMENT_MONITORING_DETAIL>
            <GenderType>Male</GenderType>
            <HMDAEthnicityType>NotHispanicOrLatino</HMDAEthnicityType>
          </GOVERNMENT_MONITORING_DETAIL>
          <HMDA_RACES>
            <HMDA_RACE>
              <HMDARaceType>White</HMDARaceType>
            </HMDA_RACE>
          </HMDA_RACES>
        </GOVERNMENT_MONITORING>
        <RESIDENCES>
```

54
<RESIDENCE>
  <ADDRESS>
    <AddressLineText>10655 N BIRCH ST</AddressLineText>
    <CityName>BURBANK</CityName>
    <PostalCode>91502</PostalCode>
    <StateCode>CA</StateCode>
  </ADDRESS>
</RESIDENCE>

<RESIDENCES>
</RESIDENCE>

</BORROWER>

<ROLE_DETAIL>
  <PartyRoleType>Borrower</PartyRoleType>
</ROLE_DETAIL>

</ROLE>

</ROLES>

<TAXPAYER_IDENTIFIERS>
  <TAXPAYER_IDENTIFIER>
    <TaxpayerIdentifierType>SocialSecurityNumber</TaxpayerIdentifierType>
    <TaxpayerIdentifierValue>000223333</TaxpayerIdentifierValue>
  </TAXPAYER_IDENTIFIER>
</TAXPAYER_IDENTIFIERS>

</PARTY>

GOVERNMENT_MONITORING Element (V3.3.1 & Earlier)

The GOVERNMENT_MONITORING container element stores the borrower’s gender, race and ethnicity data. The XML snippet below shows a typical data set that would be present for Version 3.3.1 and earlier.

<BORROWER>
  <GOVERNMENT_MONITORING>
    <GOVERNMENT_MONITORING_DETAIL>
      <GenderType>Male</GenderType>
      <HMDAEthnicityType>HispanicOrLatino</HMDAEthnicityType>
    </GOVERNMENT_MONITORING_DETAIL>
    <HMDA_RACES>
      <HMDA_RACE>
        <HMDARaceType>White</HMDARaceType>
      </HMDA_RACE>
    </HMDA_RACES>
  </GOVERNMENT_MONITORING>
</BORROWER>
GOVERNMENT MONITORING Element (V3.4 & Later)

For Version 3.4 and later, additional containers and structures were added to the GOVERNMENT_MONITORING element, to capture additional gender, race and ethnicity data being requested for HMDA compliance.

The updated structure still contains all of the previous data elements, but can now capture more detail. For example, if a borrower identifies the HMDAEthnicityType as **HispanicOrLatino**, the new structure also allows for further identification using the HMDAEthnicityOriginType values: **Cuban, Mexican, PuertoRican** or **Other**.

```xml
<BORROWER>
  <GOVERNMENT_MONITORING>
    <GOVERNMENT_MONITORING_DETAIL>
      <GenderType>Male</GenderType>
      <HMDAEthnicityCollectedBasedOnVisualObservationOrSurnameIndicator>true</HMDAEthnicityCollectedBasedOnVisualObservationOrSurnameIndicator>
      <HMDAEthnicityType>HispanicOrLatino</HMDAEthnicityType>
      <HMDAGenderCollectedBasedOnVisualObservationOrNameIndicator>false</HMDAGenderCollectedBasedOnVisualObservationOrNameIndicator>
      <HMDARaceCollectedBasedOnVisualObservationOrSurnameIndicator>false</HMDARaceCollectedBasedOnVisualObservationOrSurnameIndicator>
    </GOVERNMENT_MONITORING_DETAIL>
    <HMDA_ETHNICITY_ORIGINS>
      <HMDA_ETHNICITY_ORIGIN>
        <HMDAEthnicityOriginType>Cuban</HMDAEthnicityOriginType>
      </HMDA_ETHNICITY_ORIGIN>
    </HMDA_ETHNICITY_ORIGINS>
    <HMDA_RACES>
      <HMDA_RACE>
        <HMDA_RACE_DETAIL>
          <HMDARaceType>White</HMDARaceType>
        </HMDA_RACE_DETAIL>
      </HMDA_RACE>
    </HMDA_RACES>
  </GOVERNMENT_MONITORING>
</BORROWER>
```

If a borrower identifies the HMDARaceType as **Asian**, the new structure allows additional identification using the HMDARaceDesignationType values: **AsianIndian, Chinese, Filipino, Japanese, Korean, Vietnamese** or **Other**. If the borrower identifies the HMDARaceType as **NativeHawaiianOrOtherPacificIslander**, additional HMDA values for the HMDARaceDesignationType element are: **AsianIndian, GuamanianOrChamorro, NativeHawaiian, Samoan** or **Other**.
The XML snippet below shows a sample of borrower demographics with the HMDARaceType of **Asian**, and the HMDARaceDesignationType values of **Korean**.

```xml
<BORROWER>
  <GOVERNMENT_MONITORING>
    <GOVERNMENT_MONITORING_DETAIL>
      <GenderType>Male</GenderType>
      <HMDAEthnicityType>NotHispanicOrLatino</HMDAEthnicityType>
    </GOVERNMENT_MONITORING_DETAIL>
    <HMDA_RACES>
      <HMDA_RACE>
        <HMDA_RACE_DESIGNATIONS>
          <HMDA_RACE_DESIGNATION>
            <HMDARaceDesignationType>Korean</HMDARaceDesignationType>
          </HMDA_RACE_DESIGNATION>
        </HMDA_RACE_DESIGNATIONS>
        <HMDA_RACE_DETAIL>
          <HMDARaceType>Asian</HMDARaceType>
        </HMDA_RACE_DETAIL>
      </HMDA_RACE>
    </HMDA_RACES>
  </GOVERNMENT_MONITORING>
</BORROWER>
```

If a borrower identifies the HMDARaceType as **AmericanIndianOrAlaskaNative**, the name of the individual’s enrolled or principal tribe can be entered in the HMDARaceTypeAdditionalDescription element, as shown below.

```xml
<BORROWER>
  <GOVERNMENT_MONITORING>
    <GOVERNMENT_MONITORING_DETAIL>
      <GenderType>Female</GenderType>
      <HMDAEthnicityType>NotHispanicOrLatino</HMDAEthnicityType>
    </GOVERNMENT_MONITORING_DETAIL>
    <HMDA_RACES>
      <HMDA_RACE>
        <HMDA_RACE_DETAIL>
          <HMDARaceType>AmericanIndianOrAlaskaNative</HMDARaceType>
          <HMDARaceTypeAdditionalDescription>Miccosukee</HMDARaceTypeAdditionalDescription>
        </HMDA_RACE_DETAIL>
      </HMDA_RACE>
    </HMDA_RACES>
  </GOVERNMENT_MONITORING>
</BORROWER>
```
LOAN Element

The LOAN element is one of the core elements of the mortgage-related data set. While the child container elements and data points are defined in the MISMO LDD, there are a few concepts that will be covered in more detail in this section of the chapter. A single mortgage DEAL may need to contain data about one or more loans. For example, in addition to the “subject loan”, data may also be needed about an existing loan that is being refinanced. This section explains various use cases and how they are implemented using multiple LOAN container elements.

LOAN ROLE Attribute

The MISMO Reference model allows multiple LOAN elements to be included. The LOAN element is qualified with an attribute that describes the “role” of each instance of LOAN. Here are the definitions for each of the enumerated values for the LoanRoleType attribute:

- "SubjectLoan" - The loan that is the object of the transaction, upon which the receiving business partner will take some action.
- "RelatedLoan" - A loan linked to the subject loan by virtue of being collateralized by the same property. Examples include the loan being refinanced, or a HELOC or other subordinate lien collateralized by the same property as the subject loan.
- "HistoricalLoan" - A loan linked to this transaction via past relationship to the borrower and mortgage type. Does not have a direct bearing on the Related Loan or Subject Loan.
LOAN_STATE Element

The MISMO Reference model also allows multiple LOAN elements that show the state of the loan data at different points during the lifetime of the loan. For example, it may be useful to know what loan data was on file at the time of loan closing, compared with the loan data at the time of a modification. The LOAN_STATE child element of LOAN has several data points that identify the state of a loan at a particular date or a date and time:

- **LoanStateDate** - Specifies the date for the "Loan State Type".
- **LoanStateDatetime** - Specifies the date and time for the "Loan State Type". (Version 3.3 and later)
- **LoanStateType** - Identifies the state at a point in time for the data included in this instance of the LOAN data. Here are the valid values:
  - "AtClosing" - A snapshot of the loan data at the completion of the closing process. This is sometimes referred to as "original".
  - "AtConversion" - For loans with a conversion option, a snapshot of the loan data at the time the conversion features become effective (e.g., biweekly to monthly payments; adjustable to fixed rate amortization).
  - "AtEstimate" - A snapshot of the loan data at the point in time when a loan estimate is disclosed. (Version 3.3 and later)
  - "AtModification" - For loans which undergo term modifications not originally specified in the note, a snapshot of the loan data at the time the new note terms become effective.
  - "AtRelief" - For loans subject to payment relief, a snapshot of the loan data at the time the relief is initiated. (Version 3.3 and later)
  - "AtReset" - For balloon mortgages with a reset feature, a snapshot of the loan data on the balloon maturity date at the time the borrower exercises the reset option to modify and extend the balloon note.
  - "AtTransfer" - A snapshot of the loan data as of the effective date of the servicing transfer. (Version 3.3 and later)
  - "AtTrial" - A snapshot of the loan data at the initiation of a trial period for a workout modification. (Version 3.3 and later)
  - "AtUnderwriting" - A snapshot of the loan data at the point at which the underwriting recommendation is made. (Version 3.3 and later)
  - "Current" - A snapshot of the loan data as of the "Loan State Date".
LOAN ROLE – LOAN STATE Use Cases

Non-Modified Loans Delivered to Investor

For loans being delivered for sale to an investor, the majority of the LOAN element data will most likely be submitted with a LoanStateType of “AtClosing”. This is the data that was valid at the time the loan was closed, so the LoanStateDate would be the same as the NoteDate value.

A second LOAN element container with a LoanStateType value of “Current” could also be submitted if that loan is seasoned and certain values may have changed since closing. The LoanStateDate would be set to the date that the data was retrieved from the submitter’s system.

For both LOAN elements the LoanRoleType is “SubjectLoan”.

| Sample LOAN Containers Usage Data for Non-Modified Loans Delivered to an Investor |
|---------------------------------|---------------------------------|
| **LOAN Element 1**              | **LOAN Element 2**              |
| Subject Loan / At Closing       | Subject Loan / Current          |
| Data about the original loan:   | Data valid at the time of loan delivery: |
| • Underwriting Data             | • Current Balances, Rates,     |
| • Product Derivation           |   Option Status, Payment       |
| • Product Features             |   Status                     |
| • Note Terms                   | • MI and Credit Enhancements  |
| LoanStateDate = NoteDate       | • Escrow Details              |
|                                | • Transaction Details         |
|                                | • Program Identifiers         |
|                                | LoanStateDate = Date data     |
|                                | retrieved from submitter’s system |
**Modified Loans Delivered to Investor**

If the loan has been modified prior to delivery, the majority of the LOAN element data will most likely be submitted with a LoanStateType of “AtModification”. Data specific to the loan modification would be put in the MODIFICATION container element. Since the LOAN element would include data values at the time of the modification, the LoanStateDate value would be the same as the LoanModificationEffectiveDate value.

A minimal set of data could be delivered in a second LOAN element container with a LoanStateType of “AtClosing”, with only the data necessary to identify the original loan product and note terms. This is the data that was valid at the time the loan was closed, so the LoanStateDate value would be the same as the NoteDate value.

A third LOAN element would include the data in its current state and would have LoanStateType of “Current” and the LoanStateDate set to the date that the data was retrieved from the submitter’s system for delivery. Since the loan had been modified before delivery, the LOAN_DETAIL element’s MortgageModificationIndicator should have a value of “true”. This LOAN container could also include information about the delivery transaction, subsequent loan servicing, and other data that was updated after the loan modification.

<table>
<thead>
<tr>
<th>Sample LOAN Containers Usage Data for Modified Loans Delivered to an Investor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LOAN Element 1</strong></td>
</tr>
<tr>
<td><strong>Subject Loan / At Closing</strong></td>
</tr>
<tr>
<td>Subset of data about the original loan:</td>
</tr>
<tr>
<td>• Product Derivation</td>
</tr>
<tr>
<td>• Note Terms</td>
</tr>
<tr>
<td>LoanStateDate = NoteDate</td>
</tr>
</tbody>
</table>

| **LOAN Element 2**                     |
| **Subject Loan / At Modification**     |
| Data about the modified loan:          |
| • Underwriting Data                    |
| • Product Derivation                   |
| • Product Features                     |
| • Note Terms                           |
| • Modification Information             |
| LoanStateDate = LoanModificationEffectiveDate |

| **LOAN Element 3**                     |
| **Subject Loan / Current**             |
| Data valid as of time of loan delivery:|
| • Current Balances,                     |
|   Rates, Option Status,                |
|   Payment Status                       |
| • MI and Credit Enhancements           |
| • Escrow Details                       |
| • Transaction Details                  |
| • Program Identifiers                  |
| MortgageModificationIndicator = “true” and LoanStateDate = Date data retrieved from submitter’s system |
Converted Loans

If the loan has been converted from either biweekly to monthly payments or from an adjustable to a fixed note rate, it originally would have had a LOAN_DETAIL element with a ConvertibleIndicator set to “true”. The original note information would be submitted in a LOAN element with a LoanStateType value of “AtClosing”. This is the data that was valid at the time the loan was closed, so the LoanStateDate value would be the same as the NoteDate value.

A second LOAN element container, with a LoanStateType of “AtConversion”, holds information that has changed as a result of the conversion, such as the loan terms. The LoanStateDate would be the same as the LatestConversionEffectiveDate (located in the RATE_OR_PAYMENT_CHANGE_OCCURRENCE container element).

A third LOAN element would include the data in its current state such as current balances, option status, payment status, escrow details, etc. and would have LoanStateType of “Current”. The LoanStateDate set to the date that the data was retrieved from the submitter’s system for delivery. Since the loan had been converted before delivery, the RATE_OR_PAYMENT_CHANGE_OCCURRENCE element’s ConvertibleStatusType should have a value of “Exercised”.

| Sample LOAN Containers Usage Data for Converted Loans Delivered to an Investor |
|-------------------------------|-------------------------------|-------------------------------|
| LOAN Element 1 | LOAN Element 2 | LOAN Element 3 |
| **Subject Loan / At Closing** | **Subject Loan / At Conversion** | **Subject Loan / Current** |
| Original data about the subject loan: | Data about the converted loan: | Data valid as of time of loan delivery: |
| • Underwriting Data | • Product Derivation | • Current Balances, Rates, Option Status, Payment Status |
| • Product Derivation | • Note Terms | • MI and Credit Enhancements |
| • Product Features | | • Escrow Details |
| • Conversion Rules | | • Transaction Details |
| • Note Terms | | • Program Identifiers |
| ConvertibleIndicator = ‘true’ and LoanStateDate = NoteDate | LoanStateDate = LatestConversionEffective Date | ConvertibleStatusType = “Exercised” and LoanStateDate = Date data retrieved from submitter’s system |
Reset Balloon Loans

If the loan being delivered was originated as a balloon loan, at closing it would have had a LOANDETAIL element with a BalloonIndicator value of “true”. The original balloon note information would be submitted in a LOAN element with a LoanStateType of “AtClosing”. This is the data that was valid at the time the loan was closed, so the LoanStateDate value would be the same as the NoteDate value.

If the balloon reset option has been exercised, a second LOAN element container, with a LoanStateType value of “AtReset” holds information that has changed as a result of the balloon reset, such as the loan terms. The LoanStateDate value would be the same as the BalloonResetDate value (located in the RATE_OR_PAYMENT_CHANGE_OCCURRENCE container element).

A third LOAN element would include the data in its current state such as current balances, option status, payment status, escrow details, etc. and would have LoanStateType of “Current”. The LoanStateDate set to the date that the data was retrieved from the submitter’s system for delivery. Since the loan had been converted before delivery, the LOANDETAIL element’s BalloonResetIndicator should have a value of “true”.

| Sample LOAN Containers Usage Data for Reset Balloon Loans Delivered to an Investor |
|---------------------------------|------------------|---------------------------------|
| **LOAN Element 1**             | **LOAN Element 2** | **LOAN Element 3**             |
| Subject Loan / At Closing      | Subject Loan / At Reset | Subject Loan / Current         |
| Original data about the subject loan: | Data about the reset loan: | Current data valid as of time of loan delivery: |
| • Underwriting Data            | • Product Derivation | • Current Balances, Rates, Option Status, |
| • Product Derivation          | • Note Terms       | Payment Status                  |
| • Product Features            |                  | • MI and Credit Enhancements    |
| • Note Terms                  |                  | • Escrow Details                |
| BalloonIndicator = ‘true’ and  | LoanStateDate =   | • Transaction Details           |
| LoanStateDate = NoteDate      | BalloonResetDate  | • Program Identifiers           |
|                                |                  | BalloonResetIndicator = “true”  |
|                                |                  | and LoanStateDate = Date        |
|                                |                  | data retrieved from submitter’s system |

63
Second Lien Subject Loan with Related First Lien Loan

If the subject loan is a second lien all of the data would be submitted in a LOAN element with a LoanRoleType of “SubjectLoan” and a LoanStateType of “AtClosing”. This is the data that was valid at the time the loan was closed, so the LoanStateDate value would be the same as the NoteDate value of the second lien loan.

The data about the related first lien would be delivered with a LoanRoleType of “RelatedLoan” and a LoanStateType of “AtClosing”. The LoanStateDate would be the same as the NoteDate of the first lien loan.

A third LOAN element would include the data about the second lien loan in its current state such as current balances, option status, payment status, escrow details, etc. and would have LoanStateType value of “Current”. The LoanStateDate set to the date that the data was retrieved from the submitter’s system for delivery.

<table>
<thead>
<tr>
<th>Sample LOAN Containers Usage Data for Second Lien Subject Loan with Related First Lien Loan Delivered to an Investor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LOAN Element 1</strong></td>
</tr>
<tr>
<td><strong>Subject Loan / At Closing</strong></td>
</tr>
<tr>
<td>Original data about the second lien subject loan:</td>
</tr>
<tr>
<td>• Underwriting Data</td>
</tr>
<tr>
<td>• Product Derivation</td>
</tr>
<tr>
<td>• Product Features</td>
</tr>
<tr>
<td>• Note Terms</td>
</tr>
<tr>
<td>LienPriorityType = “SecondLien” and LoanStateDate = NoteDate of second lien</td>
</tr>
</tbody>
</table>

| **LOAN Element 2**                                      |
| **Related Loan / At Closing**                          |
| Original subset of data about the first lien related loan: |
| • Product Derivation                                   |
| • Note Terms                                           |
| LienPriorityType = “FirstLien” and LoanStateDate = BalloonResetDate |

| **LOAN Element 3**                                      |
| **Subject Loan / Current**                             |
| Current subset of data valid as of time of subject loan delivery: |
| • Current Balances, Rates, Option Status, Payment Status |
| • MI and Credit Enhancements                            |
| • Escrow Details                                       |
| • Transaction Details                                  |
| • Program Identifiers                                  |
| LoanStateDate = Date data retrieved from submitter’s system |
Loans with Concurrent Secondary Financing

If more than one concurrently closing lien exists on the subject property, the original “AtClosing” data could be submitted in the first LOAN container element, with the “Current” data submitted in a second LOAN container element. “Current” data on additional subordinate liens could be sent in one (or more) LOAN container elements with a LoanRoleType of “Related” and a LoanStateType of “Current”.

<table>
<thead>
<tr>
<th>Sample Loan Containers Usage Data for Concurrently Closing Secondary Financing when a First Lien is being Delivered</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LOAN Element 1</strong></td>
</tr>
<tr>
<td><strong>Subject Loan / At Closing</strong></td>
</tr>
<tr>
<td>Original data about the first lien subject loan:</td>
</tr>
<tr>
<td>• Underwriting Data</td>
</tr>
<tr>
<td>• Product Derivation</td>
</tr>
<tr>
<td>• Product Features</td>
</tr>
<tr>
<td>• Note Terms</td>
</tr>
<tr>
<td>LienPriorityType = “FirstLien” and LoanStateDate = NoteDate of first lien</td>
</tr>
</tbody>
</table>
**ADJUSTMENT Element**

The ADJUSTMENT element within LOAN, holds child elements that describe how the rate and payment structure of a loan can change. It contains data that specifies the rules for adjustments and conversions to the interest rate and principal and interest payment for the life of the loan as specified in the Note. It also includes provisions to capture the modifications to interest rate and principal and interest payment for each adjustment period. It can also specify the rules for selecting the appropriate index upon which the adjustments may be based.

![Diagram]

Each of the ADJUSTMENT child elements has a similar structure; one to specify index rules, one to articulate the rules in effect for the life of the conversion option or loan, one to identify the rules applicable to the “First” and “Subsequent” change periods and one to communicate associated rules that are also in effect for specified time periods. The RATE_OR_PAYMENT_CHANGE_OCCURRENCE container holds data points that communicate values resulting from execution of each of the applicable rules.

**INDEX RULE**

The INDEX_RULE element contains data points specifying the index used to govern changes in the interest rate or principal and interest payment. The data points it contains include the index source, lookback period, rounding rules, calculation methods, and index averaging rules. While multiple Index Rules can be identified for each potential adjustment to a loan (conversion, adjustable rate, or adjustable payments), typically there is only one.
Example: Use of INDEX RULE for an ARM

The interest rate adjustment is based on a 1-year LIBOR index published daily in the *Wall Street Journal* with a lookback period of 25 days. The index source and lookback period are known at the time of origination and are communicated as “Index Rules”.

<table>
<thead>
<tr>
<th>Data Point</th>
<th>Definition</th>
<th>Valid Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index Source Type</td>
<td>Specifies the type and source of index to be used to determine the interest rate at each adjustment.</td>
<td>LIBOROneYearWSJ Daily</td>
</tr>
<tr>
<td>Interest And Payment Adjustment Index Lead Days Count</td>
<td>The number of days prior to an interest rate effective date used to determine the date for the index value when calculating a new interest rate on a loan.</td>
<td>25</td>
</tr>
</tbody>
</table>

Here is how the same data would be represented in an XML format.

```xml
<INDEX_RULE>
  <IndexSourceType>LIBOROneMonthWSJDaily</IndexSourceType>
  <InterestAdjustmentIndexLeadDaysCount>25</InterestAdjustmentIndexLeadDaysCount>
</INDEX_RULE>
```

LIFETIME ADJUSTMENT RULE Elements

The various Lifetime Adjustment Rule elements hold the rules that are in place for the duration of the conversion option or life of the adjustable rate loan.

- **CONVERSION_ADJUSTMENT_LIFETIME_ADJUSTMENT_RULE**: For convertible mortgages, the lifetime rule applies for the full duration of the conversion option and includes data about option length, extendibility, repeatability, margin, rate caps, schedule, and type (biweekly to monthly or adjustable to fixed rate).
- **INTEREST_RATE_LIFETIME_ADJUSTMENT_RULE**: For ARMs, the lifetime rule applies for the full duration of the loan and includes data about: rate caps, first rate change date, calculation type, rounding and truncation.
- **PRINCIPAL_AND_INTEREST_PAYMENT_LIFETIME_ADJUSTMENT_RULE**: For loans with adjustable payments, this lifetime rule applies for the full duration of the loan and includes data about: payment caps, final recast instructions, number of times the note term can be extended, payment calculation methods, and number of payments between adjustments.
**Example:** Use of INTEREST_RATE_LIFETIME_ADJUSTMENT_RULE for an ARM

A loan that closed on January 15, 2010 has with a 5-year initial fixed interest rate of 5.5% with a 5/2/6 cap structure and a margin of 2.6%. There is no lifetime floor rate. The CeilingRatePercent is calculated as the initial note rate of 5.5% plus the lifetime cap of 6% (the “6” of the 5/2/6). The FirstRateChangePaymentEffectiveDate begins with the end of the 5-year initial fixed period.

<table>
<thead>
<tr>
<th>Data Point</th>
<th>Definition</th>
<th>Valid Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ceiling Rate Percent</td>
<td>The stated maximum percentage to which the interest rate can increase over the life of the loan.</td>
<td>11.5</td>
</tr>
<tr>
<td>First Rate Change Payment Effective Date</td>
<td>The due date of the payment at the first calculated interest rate change. To arrive at the actual (true) date that interest begins to accrue at the changed rate one payment period is subtracted if interest is paid in arrears.</td>
<td>2015-03-01</td>
</tr>
<tr>
<td>Interest Rate Rounding Percent</td>
<td>The percentage to which the interest rate is rounded when a new interest rate is calculated. This field is used in conjunction with Interest Rate Rounding Type, which indicates how rounding should occur.</td>
<td>0.125</td>
</tr>
<tr>
<td>Interest Rate Rounding Type</td>
<td>Indicates how the interest rate is rounded when a new interest rate is calculated for an ARM change. The interest rate can be rounded Up, Down, or to the Nearest Factor. This field is used in conjunction with Per Change Interest Rate Rounding Factor, which indicates the percentage to which the rounding occurs.</td>
<td>Nearest</td>
</tr>
<tr>
<td>Margin Rate Percent</td>
<td>The number of percentage points to be added to the index to arrive at the new interest rate.</td>
<td>2.60</td>
</tr>
</tbody>
</table>

**CONVERSION OPTION PERIOD ADJUSTMENT RULES**

This set of rules can repeat as required to specify the rules governing each conversion period associated with a loan (if there is more than one). Each conversion period is defined by the values in ConversionOptionPeriodEffectiveDate, ConversionOptionPeriodExpirationDate, and ConversionOptionPeriodType.

**PER CHANGE ADJUSTMENT RULES**

This set of rules is expected to repeat twice, with each rule identified by the value of AdjustmentRuleType as either “First” or “Subsequent”. If the value of AdjustmentRuleType is “First”, then the various Per Change Adjustment Rule element types consists of instructions governing the initial interest rate or payment change. If the value of AdjustmentRuleType is set to “Subsequent”, then the instance of the Per
Change Adjustment Rule consists of instructions governing the interest rate changes that follow the “First” rate or payment change. Typically, the “Subsequent” adjustment rules remain in place once they become effective.

- **INTEREST_RATE_PER_CHANGE_ADJUSTMENT_RULE**: If the ARM has an initial fixed period (AdjustmentRuleType value of “First”), the data in this element governs the rate change that commences at the end of the fixed rate period. It specifies for this initial rate change, rate change maximums and minimums, calculation method, and rule duration. When this element has a AdjustmentRuleType value of “Subsequent”, it provides data parameters for all following rate changes.

- **PRINCIPAL_AND_INTEREST_PAYMENT_PER_CHANGE_ADJUSTMENT_RULE**: When the AdjustmentRuleType is “First”, this element governs the initial payment change period. It specifies payment percentage and dollar increase and decrease maximums and minimums, calculation method, and rule duration (among other things). When this element has a AdjustmentRuleType value of “Subsequent”, it provides data parameters for all following rate changes.

**PERIODIC ADJUSTMENT RULES**

This set of rules can be used for interest rate or payment adjustments and can repeat, with each rule identified by the value of AdjustmentRuleType as either “First” or “Subsequent”.

- **INTEREST_RATE_PERIODIC_ADJUSTMENT_RULE** - When the AdjustmentRuleType value is “First”, this element contains instructions governing the establishment of the initial periodic base rate. It specifies the initial periodic base rate, effective date, and the selection date for next periodic base rate, and the upper and lower limits for the interest rate relative to the base rate. When the AdjustmentRuleType value is “Subsequent”, this element contains instructions governing the periodic base rates that follow the initial base rate. It provides instructions for establishment, duration, and use of all following periodic base rates.

- **PRINCIPAL_AND_INTEREST_PAYMENT_PERIODIC_ADJUSTMENT_RULE** - When the AdjustmentRuleType value is “First”, this element contains instructions governing the first recast period for negatively amortizing loans, as well as the annual payment increase cap. When the AdjustmentRuleType value is “Subsequent”, this element contains instructions governing subsequent recast periods.
Example: Use of INTEREST_RATE_PER_LIFETIME_ADJUSTMENT_RULE for an ARM.

A loan that closed on January 15, 2010, has with a 5-year initial fixed interest rate and adjusts annually thereafter with a 5/2/6 cap structure.

The first instance of the rule captures the interest rate caps for the initial adjustment period (the “5” of the 5/2/6) which begins 5 years after the note date at the conclusion of the fixed rate period, and remains in effect for one year.

**INTEREST RATE PER CHANGE ADJUSTMENT RULE — Instance #1**

<table>
<thead>
<tr>
<th>Data Point</th>
<th>Definition</th>
<th>Valid Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjustment Rule Type</td>
<td>Specifies whether the occurrence of the adjustment is the first change or a subsequent change.</td>
<td>First</td>
</tr>
<tr>
<td>Per Change Maximum Decrease Rate Percent</td>
<td>The maximum number of percentage points by which the rate can decrease from the previous interest rate.</td>
<td>5.0</td>
</tr>
<tr>
<td>Per Change Maximum Increase Rate Percent</td>
<td>The maximum number of percentage points by which the rate can increase from the previous interest rate.</td>
<td>5.0</td>
</tr>
<tr>
<td>Per Change Rate Adjustment Effective Date</td>
<td>The date when the Interest Rate Per Change Adjustment Rule first becomes applicable. The Rule remains in effect unless another Rule with a later date is present on the loan.</td>
<td>2015-02-01</td>
</tr>
<tr>
<td>Per Change Rate Adjustment Frequency Months Count</td>
<td>The number of months between rate adjustments, if the interest rate on the subject loan can change.</td>
<td>12</td>
</tr>
</tbody>
</table>

The second instance of the INTEREST_RATE_PER_CHANGE_ADJUSTMENT_RULE captures the interest rate caps for all subsequent adjustment periods (the “2” of the 5/2/6). This rule becomes effective at the conclusion of the initial adjustment period. It is applied annually thereafter and stays in effect for the remaining life of the loan.

**INTEREST RATE PER CHANGE ADJUSTMENT RULE — Instance #2**

<table>
<thead>
<tr>
<th>Data Point</th>
<th>Valid Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjustment Rule Type</td>
<td>Subsequent</td>
</tr>
<tr>
<td>Per Change Maximum Decrease Rate Percent</td>
<td>2.0</td>
</tr>
<tr>
<td>Per Change Maximum Increase Rate Percent</td>
<td>2.0</td>
</tr>
<tr>
<td>Per Change Rate Adjustment Effective Date</td>
<td>2016-02-01</td>
</tr>
<tr>
<td>Per Change Rate Adjustment Frequency Months Count</td>
<td>12</td>
</tr>
</tbody>
</table>

The following is an XML representation of the ADJUSTMENT structure for the ARM
example used in earlier tables.

<ADJUSTMENT>
  <INTEREST_RATE_ADJUSTMENT>
    <INDEX_RULES>
      <INDEX_RULE>
        <IndexSourceType>LIBOROneMonthWSJDaily</IndexSourceType>
        <InterestAdjustmentIndexLeadDaysCount>25</InterestAdjustmentIndexLeadDaysCount>
      </INDEX_RULE>
    </INDEX_RULES>
    <INTEREST_RATE_LIFETIME_ADJUSTMENT_RULE>
      <CeilingRatePercent>11.5</CeilingRatePercent>
      <FirstRateChangePaymentEffectiveDate>2015-03-01</FirstRateChangePaymentEffectiveDate>
      <InterestRateRoundingPercent>0.125</InterestRateRoundingPercent>
      <InterestRateRoundingType>Nearest</InterestRateRoundingType>
      <MarginRatePercent>2.60</MarginRatePercent>
    </INTEREST_RATE_LIFETIME_ADJUSTMENT_RULE>
    <INTEREST_RATE_PER_CHANGE_ADJUSTMENT_RULES>
      <INTEREST_RATE_PER_CHANGE_ADJUSTMENT_RULE>
        <AdjustmentRuleType>First</AdjustmentRuleType>
        <PerChangeMaximumDecreaseRatePercent>5.0</PerChangeMaximumDecreaseRatePercent>
        <PerChangeMaximumIncreaseRatePercent>5.0</PerChangeMaximumIncreaseRatePercent>
        <PerChangeRateAdjustmentEffectiveDate>2015-02-01</PerChangeRateAdjustmentEffectiveDate>
        <PerChangeRateAdjustmentFrequencyMonthsCount>12</PerChangeRateAdjustmentFrequencyMonthsCount>
      </INTEREST_RATE_PER_CHANGE_ADJUSTMENT_RULE>
      <INTEREST_RATE_PER_CHANGE_ADJUSTMENT_RULE>
        <AdjustmentRuleType>Subsequent</AdjustmentRuleType>
        <PerChangeMaximumDecreaseRatePercent>2.0</PerChangeMaximumDecreaseRatePercent>
        <PerChangeMaximumIncreaseRatePercent>2.0</PerChangeMaximumIncreaseRatePercent>
        <PerChangeRateAdjustmentEffectiveDate>2016-02-01</PerChangeRateAdjustmentEffectiveDate>
        <PerChangeRateAdjustmentFrequencyMonthsCount>12</PerChangeRateAdjustmentFrequencyMonthsCount>
      </INTEREST_RATE_PER_CHANGE_ADJUSTMENT_RULE>
    </INTEREST_RATE_PER_CHANGE_ADJUSTMENT_RULES>
  </INTEREST_RATE_ADJUSTMENT>
</ADJUSTMENT>
RATE OR PAYMENT CHANGE OCCURRENCES

Unlike all the containers ending in “Rule” discussed above, the data in this container is not known at time of origination. Instead, the data points in this container capture the status or results of the execution of the rules. Because the execution of the rules occurs after the loan has closed, the RATE_OR_PAYMENT_CHANGE_OCCURRENCE container is communicated in the “Current” LOAN container.

- **Convertible Loans** - For convertible mortgages, the RATE_OR_PAYMENT_CHANGE_OCCURRENCE container communicates data about the execution of the conversion option including the status of the conversion option, the last date the conversion option was exercised, and the next date the conversion option can be exercised.

- **Adjustable Rate Loans** - For ARMs, the RATE_OR_PAYMENT_CHANGE_OCCURRENCE container communicates data about the execution of the specified AdjustmentRuleType, including values of index, interest rate, and the adjustment on the adjustment date, and the next rate adjustment date.

- **Adjustable Payment Loans** - For loans with adjustable payments, the RATE_OR_PAYMENT_CHANGE_OCCURRENCE container communicates data about the execution of the specified AdjustmentRuleType, including values of the payment, next payment change date, and payment cap dates.

- **Balloon Loans** - For balloon loans that have exercised the reset option, the RATE_OR_PAYMENT_CHANGE_OCCURRENCE container communicates the Balloon Reset Date.

**Example:** Use of RATE_OR_PAYMENT_CHANGE_OCCURRENCE for an ARM

A loan that closed on January 15, 2010, has with a 5-year initial fixed interest rate. The lender delivered the mortgage to a GSE (Fannie Mae or Freddie Mac) on January 25, 2010.

The loan is still in its initial fixed rate period when it is delivered to the GSE. Therefore, the NextRateAdjustmentEffectiveDate is the PerChangeRateAdjustmentEffectiveDate of the first INTEREST_RATE_PER_CHANGE_ADJUSTMENT_RULE, the beginning of the adjustment period.

<table>
<thead>
<tr>
<th>Data Point</th>
<th>Definition</th>
<th>Valid Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NextRateAdjustmentEffectiveDate</td>
<td>The date on which the next interest rate adjustment goes into effect.</td>
<td>2015-02-01</td>
</tr>
</tbody>
</table>
CHAPTER 5: EXTENDING THE MISMO STANDARD

In the MISMO V3 Reference Model, every container element includes one EXTENSION element. The EXTENSION container provides maximum flexibility to a given version of a MISMO message by establishing a structured framework for adding data. Organizations can take advantage of XML Namespaces to place needed data points that are not yet defined in the MISMO standard, or proprietary data points that would never be included in the MISMO standard, into a message for exchange with business partners. The definition of the EXTENSION container makes it possible to add additional content yet still validate against the MISMO standardized Schema.

At the end of this chapter there are several examples of common methods of extending the MISMO Standard using both EXTENSION and Namespace concepts. Much of the content for this chapter was derived the MISMO Engineering Guideline – Extending the Standard (MEG0025), which is available on the MISMO web site at the link below:


Goals of the MISMO Extension Model

The XML standard that is the core of the MISMO Reference Model is extensible by its nature. In fact, “eXtensible” is the “X” of XML. MISMO’s Version 3 architecture provides an orderly way of extending a MISMO message to hold additional data needed for mortgage processing that is not included in the MISMO Reference Model. The MISMO Architecture Workgroup had specific goals for the MISMO Extension Model:

- Allow any container element to be extended.
- Separate the MISMO provided files from the files you need to edit in order to add extensions.
- Allow extensions from multiple organizations to coexist.
- Allow simultaneous schema validation of MISMO and Extension content.

Guidelines for Using the EXTENSION Element

All MISMO container elements will include one optional EXTENSION element and will have a cardinality of zero to one. The MISMO Schema defines the EXTENSION element as allowed to contain an unlimited number of xs:any elements. This means that it acts as a “wild card” and it can have any name and any type of XML content. The MISMO Engineering Guideline - MEG0025 has recommendations for the use of the EXTENSION element so that its use is consistent with other element structures used in the MISMO Schema.

Each xs:any element may have its own namespace, which means that data belonging to multiple namespaces can be contained within a single EXTENSION element. The namespace(s) used for the EXTENSION element should be declared either in the MESSAGE root element or if the namespace is only used once it could be declared in the
EXTENSION element.

Differences between EXTENSION element use in V3.2 and later and earlier MISMO versions are discussed below, along with rules for adding data point elements and container elements and XML examples of extended data elements.

**EXTENSION Element – Version 3.2 and Later**

Beginning with MISMO Version 3.2, the definition of the EXTENSION element was modified to add a MISMO child element and an OTHER child element as shown in the diagram below.

- **MISMO** – this child element of EXTENSION allows data points or containers from a later MISMO version to be added to a version that is currently in use. For example, if a lender’s system is currently using MISMO Version 3.2, but they need to use a few data points that were added to Version 3.3. They could add these data points to the EXTENSION / MISMO element to give them the use of these Version 3.3 data points while still using Version 3.2. Any data points or containers added to the EXTENSION / MISMO element would remain under the MISMO namespace.

- **OTHER** – this child element is used for any other non-MISMO data points or containers and would use the namespace declaration of the entity that added the data.
EXTENSION Element – Versions 3.0 and 3.1

The EXTENSION element used in MISMO Versions 3.0 and 3.1 did not have MISMO and OTHER child elements. Extended elements are added directly under the EXTENSION element and have their own namespace declarations.

Adding Data Point Elements

Extension data points should only be added to an EXTENSION element of a container element that holds other data points. In the example below, the V3.2 PROPERTY_OWNER element schema definition contains only data point elements. Therefore, only data point elements should be added to the PROPERTY_OWNER / EXTENSION element.
In the V3.2 XML snippet below, two extended data points were added. The first one, `PropertyOwnedSinceDate`, is a MISMO data point that was added to the MISMO Reference Model in a fictional, future V3.9 version that a business partner would like to use in their current V3.2 messages. By adding this data point to the `EXTENSION / MISMO` element, the current V3.2 message will still validate against the V3.2 Reference Model Schema.

A second data point, `OtherPropertiesOwnedCount`, is a non-MISMO data point used by a fictional “ACME Mortgage” that they would like to include in their V3.2 messages. Since this is an ACME data point, they must assign it to the “acme” namespace using the `xmlns:acme` namespace declaration as shown in the `EXTENSION` element below. This namespace declaration could also be placed in the MISMO root `MESSAGE` element. Note that the extended data point, `acme:OtherPropertiesOwnedCount`, is marked with the acme: namespace prefix when it is placed in the `OTHER` element.

```xml
<PROPERTY_OWNER>
  <OwnershipPercent>100</OwnershipPercent>
  <RelationshipVestingType>JointTenants</RelationshipVestingType>
  <EXTENSION xmlns:acme="www.acmeMortgage.com">
    <MISMO>
      <PropertyOwnedSinceDate>2013-02-15</PropertyOwnedSinceDate>
    </MISMO>
    <OTHER>
      <acme:OtherPropertiesOwnedCount>5</acme:OtherPropertiesOwnedCount>
    </OTHER>
  </EXTENSION>
</PROPERTY_OWNER>
```

**Adding Container Elements**

Extension containers should only be added to an `EXTENSION` element of a container element that holds other data points. In the example below, the V3.2 `SERVICE_PRODUCT` element schema definition contains only container elements. Therefore, only other container elements should be added to the `SERVICE_PRODUCT / EXTENSION` element.
In the V3.2 XML snippet below, a new container was added to the MISMO SERVICE/Product container. SERVICE_COMPLIANCE_RATING is a non-MISMO container element a fictional “Ajax Compliance Ratings” that they would like to include in their V3.2 messages. Since this is an Ajax container, they must assign it to the “ajax” namespace using the xmlns:ajax namespace declaration as shown in the EXTENSION element below. This namespace declaration could also be placed in the MISMO root MESSAGE element. Note that the Ajax data container and its data points are marked with the ajax: namespace prefix when they are placed in the OTHER element.

```xml
<SERVICE_PRODUCT>
  <EXTENSION xmlns:ajax="www.AjaxComplianceRatings.org">
    <OTHER>
      <ajax:COMPLIANCE_RATING>
        <ajax:ComplianceDate>2012-02-13</ajax:ComplianceDate>
      </ajax:COMPLIANCE_RATING>
    </OTHER>
  </EXTENSION>
</SERVICE_PRODUCT>
```

Adding Schema Validation for EXTENSION Elements

In the XML samples used previously, all of the extended data will validate against the MISMO Reference Model Schema since all EXTENSION child elements are declared as an xs:any type. But no further validation on the extended data is done unless schema validation is added for the extended data. For example, there would be no checks to confirm that the extended data element names were spelled correctly or that the date fields were in a specific date format.

This section discusses two options for adding schema validation for extended data and the advantages and disadvantages of each.

**Approach 1: Modified Schema Files Using xs:redefine**

The first approach leaves all of the original MISMO schema files “as is”, but involves creating one or more new schema files that define the extended content that is to be added to the MISMO EXTENSION / OTHER element. Then a copy of the MISMO “root” schema is created and modified to import the extended content schemas files and declare their namespaces. The final step is to add a xs:redefine element to specify which EXTENSION element the new schema structure is to be added to.

The main risk of this approach that not all platforms will properly recognize structures subject to the xs:redefine when performing validation.

**Approach 2: Modifying MISMO Extension Schema Files**

The second approach is similar to the first one in that it involves creating one or more new schema files that define the extended content that is to be added to the MISMO
EXTENSION / OTHER element. Then a copy of the MISMO “root” schema is created and modified to import the extended content schemas files and declare their namespaces. But instead of using the xs:redefine element to redefine the EXTENSION content, the actual MISMO Extension schema block is modified.

This approach will support schema validation in most platforms, but requires changes to the published MISMO schemas to support validation of the extended content.
CHAPTER 6: IMPLEMENTING MISMO: PROGRAMMING GUIDANCE

For many MISMO implementations, it is necessary to write custom applications or services, either to read and interpret MISMO messages from other systems or trading partners, or to generate messages for outside consumption.

While most application developers are already familiar with how to read and create XML files using application development frameworks and programming languages such as C# (.NET) or Java, MISMO messages often present unique challenges. By virtue of the sheer volume of information that can be packed into a MISMO message, there are certain techniques that developers and application architects should be aware of in order to help them create MISMO solutions that achieve acceptable performance and maintain the integrity of the underlying data.

**Reading and Processing MISMO XML**

This section covers best practices and trade-offs to consider when reading or consuming a MISMO XML file that was created by another system or organization.

**Using an XML Object Model**

The most common method of reading or consuming an XML file within application code is through the use of an XML Object Model. Using this technique, developers typically use tools that are built into their application development framework to load the entire XML file into memory. Once the file is loaded, the developer can locate and read containers, elements, and attributes using these same tools.

Developers using the Microsoft .NET framework and programming in either the C# or Visual Basic .NET languages use what is known as the XmlDocument object, while Java developers use the Document Object Model (DOM) Parser.

The primary advantage of using this method to consume XML is that it is the simplest and most familiar to developers. When tasked with extracting well-defined pieces of information from an XML Document (such as the names of all borrowers), most developers can quickly write code that will locate and read the data from within the file.

For the vast majority of application development scenarios, using an XML Object Model to read MISMO XML files is the best approach.

The main disadvantage to using an XML Object Model is the overhead incurred when consuming very large files. In cases where MISMO XML files are so large that memory usage or processing speed becomes unacceptable, developers might need to explore alternative methods of reading the XML file. These other methods are discussed in the Performance Considerations section below.
Validating Against a Schema

When using an XML Object Model, XML files are typically validated only so far as to make sure that they are well-formed XML files. That is, the files are checked to make sure they contain matching opening and closing tags for elements, allowable names for elements and attributes, and the like.

The MISMO Reference Model, which is an XML Schema, defines what the content of the XML file should look like. It specifies the names for containers and elements within the XML file, as well as the overall parent/child structure for MISMO messages.

Most application development frameworks such as .NET and Java also provide the capability to validate a document against an XML schema, such as the MISMO Reference Model. When using this technique, the developer can ensure that a MISMO XML message contains both correctly constructed XML as well as a valid MISMO structure.

Performance Considerations

While the ideal programming best practice would be to use an XML Object Model to load a MISMO XML message and then validate it against the MISMO XML schema, performance and computational power must be taken into account, especially for larger files.

Validating XML against a schema can be a resource-intensive step. In cases where the MISMO XML files being consumed are coming from a source that you trust to provide well-formed messages, developers might consider skipping the schema validation step in order to improve performance.

In cases where files are so large as to adversely affect memory usage or processing speed, programming techniques that do not load the entire XML file at one time are another option. The XMLReader object in .NET and SAX parser in Java are two tools that developers can use to read XML files using a forward-only read-only mechanism. Under this model, applications can only process XML files in one direction – from beginning to end. This method is more difficult and less flexible for programmers, but has the potential to greatly reduce processing time and memory usage.

Creating MISMO Data XML

This section includes practical advice and considerations for developers creating MISMO XML messages within application code. Many of the techniques, such as using an XML Object Model, are similar to those used in reading files.

Use an XML Object Model or Other Built-in XML Support

The most important piece of advice for developers generating MISMO, or any other type of XML message or file, is never to create these files using text manipulation techniques,
such as string concatenation. Although easy, these techniques provide no assurance that the XML generated is a valid MISMO message, or even well-formed XML.

MISMO messages should usually be generated using the same XML Object Model tools discussed previously. This technique allows developers to generate the entire document or message in memory using programming methods that make it easy to assemble an XML document in the correct structure. XML Object Models also ensure that the end result is well-formed XML.

Additionally, most XML Object Model tools also include the option to validate against an XML schema, such as the MISMO Reference Model. When feasible to use, this option provides assurance that the XML files produced are valid MISMO messages.

**Use a Template File**

To increase the performance of file generation, consider starting with a template file, or skeleton XML file that already contains as much of the required structure as possible. This will also minimize the possibility of introducing incorrect XML.

**Handling Large Sets of Data**

In scenarios where massive amounts of data must be packaged within a MISMO message for transfer between trading partners, some members of the MISMO community have suggested the practice of using references to external data in place of inline data.

A frequently cited scenario is a case where one vendor must deliver a file containing information on a very large number of loans (one million, for example) to another organization, such as Fannie Mae, Freddie Mac, or other investors or servicers.

The suggested alternative to creating a file that is many gigabytes (or more) in size is to include only absolute minimal information on each loan in the MISMO file. Instead of a full set of loan data, a reference is specified for each loan that allows the consuming system to retrieve full details on each loan as the file is processed.

**Transforming Other XML Data Formats to MISMO**

Some systems must convert other XML formats, such as earlier versions of MISMO, into the most current MISMO standard XML, or vice versa. You can use XSLT to manage this process.

XSLT uses an XML-based language called XSL, or Extensible Stylesheet Language, to specify instructions for reading one type of XML document and generating another type of XML.

Since the HTML code that defines the structures of web pages is itself a form of XML, XSLT is commonly used to transform pure data XML representations, such as MISMO messages, into HTML that can be viewed in a web browser.
Although a full discussion of XSLT is beyond the scope of this document, many good resources are available on the internet for learning this technique.

**Sending and Receiving MISMO XML**

One final consideration for custom application development activities involving MISMO is how to appropriately handle sending and receiving MISMO XML messages between systems.

Since the MISMO standard is based upon XML, which is widely used throughout the entire information technology field, there exists a wealth of information on best practices for sending and receiving this sort of data, whether by SOAP or REST web services, message-oriented middleware (MOM), and other techniques.

Given the sensitive nature of much of the information within mortgage-related data messages, probably the most important thing to ensure is that all messages are transmitted securely. Since XML is a form of plain text that is easily readable by humans, messages containing sensitive data must be properly encrypted when transmitting over communication technologies and protocols.

Additional steps should also be taken to detect any tampering that might occur to messages while in transit. Tamper-evident seal technology and its use in MISMO implementations is referenced in the *XML Digital Signature Implementation Guide* ([www.mismo.org](http://www.mismo.org)).

For more information about secure data transfer, see the “Security Principles” appendix of this guide.

**Conclusion**

Although custom applications and services vary widely in their usage scenarios, a careful examination of functional and performance requirements, weighed against the various options for handling MISMO data in code, should allow for creating solutions that meet needs and perform well.
CHAPTER 7: VERSION 3 SMART Doc®

This chapter provides an overview of the MISMO SMART Doc® specification and provides summaries of some of the I-Guides that are available to support the specification.

Overview

The SMART Doc® specification is defined by MISMO to provide a consistent, standard electronic document format for the mortgage industry. SMART is an acronym for Securable, Manageable, Archiveable, Retrievable, and Transferable, the key attributes that the eMortgage Workgroup determined were necessary for efficient eMortgage process flow.

SMART Doc® Version 1.02 was based on XML DTD, and is the currently-accepted standard format for eNote delivery to Fannie Mae and Freddie Mac. Approximately 300,000 eNotes have been registered on the MERS® eRegistry as of June 2013, and virtually all of them are in the SMART Doc® V1.02 Category One format, which requires an XHTML View section and an XML Data section, and linking statements that link each piece of data in the View (what the Borrower sees and signs) to a corresponding piece of data in the Data section (for lights-out back-end processing).

In the MISMO Version 3 Reference Model structure, a separate SMART Doc® specification is no longer necessary. The SMART Doc® structure is inherently contained within the Reference Model XML schema, via the DOCUMENT structure. However, for broad industry understanding, we call this new structure SMART Doc® Version 3 (SDV3). SDV3 addresses many of the challenges that were inherent in the Ver. 1.02 specification, and supports any View format (for example, Adobe PDF, Microsoft Word, image files, RTF, etc.) A single SDV3 document can contain multiple Views, such as pre- and post-signing, pre- and post-recording, and other useful process flow points.

General information about Version 3 SMART Docs and Frequently Asked Questions (FAQs) are available on the MISMO website.3

The V3 SMART Doc® I-Guides

The MISMO eMortgage Work Group is creating several I-Guides that cover specific topics related to MISMO Version 3 Docs in much more detail. Here is a list of Version 3 SMART Doc® I-Guides that are either currently available, or are under development.

SMART Doc® PDF I-Guide
Portable Document Format (PDF) is an open, device-independent standard for

representing documents and document interactions created by Adobe Systems and published by the International Organization for Standardization (ISO) [ISO 32000-1:2008]. This document provides the guidelines and requirements for creators and consumers of MISMO SMART Docs® in conjunction with the PDF specifications above to allow interoperability in the context of complex business workflows.

**XML Digital Signature I-Guide**

The XML Signature Syntax and Processing specification [XML-DSIG] is an XML syntax and processing model for digital signatures defined by the World Wide Web Consortium (W3C). XML digital signatures can be applied to any digital content, including text and XML. An XML signature may be applied to the content of one or more resources.

An XML digital signature is commonly used to create a tamper-evident seal around an entire electronic resource, which invalidates any changes to any part of that resource after the XML digital signature has been applied. This approach is useful when the signed resource is not supposed to be changed in subsequent steps of its lifecycle.

In addition to that, the XML Signature Syntax and Processing specification also allows for using XPath filtering to selectively sign any portion of a resource, while leaving other portions unsigned and open for changes. This approach is useful for a collaborative document that is incrementally modified by multiple participants during its lifecycle, before becoming a final, immutable record. Selective signing allows each participant in the process to verify that there have been no changes to the document prohibited by tamper-evident seals applied by previous participants; a participant can then perform and seal the changes that are pertinent to their function in the process and forward the document for further processing by other participants.

The MISMO V3 reference model enables the use of XML digital signatures for creating tamper-evident electronic seals around the MESSAGE and the DOCUMENT elements. It also enables tamper-evident seals to be used in conjunction with stakeholder, witness, and notary signatures represented in the SIGNATORY element of a DOCUMENT.

Although the XML Signature Syntax and Processing specification is a mature and widely-adopted technology, many considerations need to be taken into account when creating signatures that need to be interoperable between multiple participants using different systems and platforms.

This document provides the guidelines and requirements for creators and consumers of signed MISMO MESSAGES and DOCUMENTS to allow interoperability in the context of complex mortgage business workflows.

**Zip Archive Implementation Guide**

A MISMO V3 MESSAGE or DOCUMENT can be represented as one self-contained XML instance document, or as multiple resources spread out across different networks. This
document describes how to take a MESSAGE or DOCUMENT that is composed by multiple files and store all its pieces into a standalone ZIP archive.

Earlier SMART Doc® I-Guides

There are several earlier SMART Doc® I-Guides that provide valuable information about eMortgages, eMortgage closings, eVaults, and eSigned PDFs and other topics. Summaries of a few of these documents is included here. Check the eMortgage Specification section of the MISMO web site (www.mismo.org) for current versions.

eMortgage Guide

The MISMO eMortgage guide is intended to educate the mortgage industry community about the business and (high-level) technical aspects of eMortgage implementations and their potential benefits. It provides guidance on how to get started with your own eMortgage system implementation, and describes general industry-acceptable guidelines for eMortgages that mortgage industry participants may elect to apply across related mortgage processes. This guide provides general information about the legal framework surrounding eMortgage implementation. It is educational in nature and is not intended as legal advice. Professional advice should be sought in connection with any specific efforts to implement eMortgages. This guide applies to all MISMO versions of SMART Doc® implementations.

eMortgage Closing Guide

The MISMO eMortgage Closing Guide describes and explains general electronic closing concepts, definitions, and voluntary guidelines. It is intended to provide general guidelines for electronic closing platforms and/or services. This guide provides general information about the legal framework surrounding electronic closing implementation. It is educational in nature and is not intended as legal advice. Professional advice should be sought in connection with any specific efforts to implement electronic closing.

This guide is not intended to be a technical implementation guide. It also does not provide information about any specific company’s internal processes, patented concepts, business logic, algorithms, or other proprietary details nor is it intended to affect the existing obligations (contractual or otherwise) between business partners.

eRecording – Electronic Document Formatted Recordable Instruments

The eRecording guide was jointly produced by MISMO and PRIA (Property Records Industry Association). This document describes the business requirements to allow for the eRecording of electronic recordable instruments in common electronic document formats. It also includes business functions, a process flow diagram, and project assumptions, constraints, and dependencies.

As the adoption of electronic documents increases, the need for and use of electronic
recording (eRecording) in the various recording jurisdictions in the country is also growing. Concurrent with this requirement will be the eRecording of the real estate industry’s electronic recordable instruments. Therefore, the goal of this document is to define the requirements for electronic instruments to be electronically recorded by applications and systems in public land records offices.

High level business requirements and benefits:

- Prepare electronic instruments so that they are electronically recordable.
- Upload or transmit electronic instruments to closing systems and/or various other platforms.
- Execute recordable electronic instruments on closing systems and/or various other platforms.
- Deliver recordable instruments electronically from closing systems to various eRecording applications and systems in public land records offices.
- Process electronic instruments including fee and payment information.
- Enable the return of electronic document submissions (recorded or rejected) to the eClosing platforms, or enable the closing platforms to retrieve electronic document submissions (recorded or rejected).

The eRecording of electronic instruments provides the following benefits to county recorders, mortgage lenders, and closing agents:

- Improved ability to manage real estate document volumes,
- Operational cost savings and productivity gains,
- Improved data accuracy and reduced errors,
- Effective and efficient use of resources, and
- Timely recording of and more accurate land records.

**eSigned PDF Guidelines**

This guidance focuses on Portable Document Format (PDF) v. 1.6. PDF is a file format for representing documents in a device independent and resolution independent format. PDF files are self-contained, they include the text, fonts, bitmap images, and vector graphics that compose the document. PDF files are not dependent on the application software, hardware, or operating system originally used to create or view the document. A PDF file will render exactly the same regardless of its origin or destination. Adobe’s licensing of the PDF specification allows anyone to create applications that read and write PDF files without having to pay royalties to Adobe Systems. Adobe has a number of patents relating to the PDF format, but licenses them on a royalty-free basis for use in developing software that complies with the PDF specification. This availability of PDF has encouraged its widespread adoption by many industries through de facto standardization, formal adoption or mandates, and the development of industry-specific guidance.
eMortgage Vaulting Guide

The purpose of this document is to provide information and guidance for evaluating, implementing, and relying on electronic vaults for storing electronic documents, and in particular, the eNote SMART Doc® Version 1.02. The term, “electronic vault” is typically used in the mortgage industry today to identify a system where electronic documents are stored for safekeeping. However, to date, there has been no comprehensive and authoritative source that defines requirements and responsibilities for implementing and operating an electronic vault on the behalf of lenders, warehouse lenders, or investors.

This is not intended to be a visionary document, but rather one that can be put to immediate use to provide guidance on finding answers to the most commonly asked questions for people interested in the requirements for storing electronic promissory notes (eNotes). The initial focus is on the eNote because the investor community has contributed considerable resources within MISMO to develop a standard uniform eNote. The processing of an eNote is also less dependent on the cooperation of third parties since it is not recorded in the county land records.
CHAPTER 8: UNIVERSAL LOAN IDENTIFIER

New Identification Requirements

The Dodd-Frank Wall Street Reform and Consumer Protection Act of 2010 (Pub. L. 111-203/H.R. 4137 (“Dodd-Frank Act”), as part of amendments made to the Home Mortgage Disclosure Act (“HMDA”), granted the Consumer Financial Protection Bureau (“CFPB”) the authority to require depository institutions to itemize statistical information about their mortgage loans using a “universal loan identifier” in a manner that the CFPB may deem appropriate.4

The CFPB’s Revised HMDA/Regulation C Final Rule specifies the use of a new Universal Loan Identifier (ULI), such that each loan has a unique number through the entirety of its life cycle. 5 The CFPB has a number of requirements regarding this new number and its usage, in order to maintain safeguards regarding consumer privacy, and to ensure the identifier’s value in reporting.

New requirements for the ULI, to be provided for each covered loan or application that can be used to retrieve the covered loan or application file:

- **Unique:** The number must be unique within the institution.
- **Non-sensitive:** The number should not include the applicant’s or borrower’s name or Social Security number, or any other borrower-identifying information.
- **Go forward only:** The new ULI requirement would apply only on a going-forward basis; previously reported loans should use the identifier that was previously reported.
- **Applies to a single loan:** Only one ULI should be assigned to any particular loan application or covered loan.
- **Continuous:** The same ULI should be assigned at loan application and remain with that transaction as it moves through the process to become a mortgage loan. The resulting mortgage loan should retain the same ULI number.
- **Consistent:** A financial institution shall use a ULI that was reported previously to refer only to the same loan or application for which the ULI was used previously.
- **Is Inherited in Purchase:** A financial institution that purchases a covered loan must use the ULI that was assigned by the financial institution that originated the covered loan.
- **One and only one:** A ULI number may not be re-used or recycled--
  - The ULI number may not be re-used or applied to another loan or mortgage loan application, even if the original transaction fell out of the mortgage

---

5 Under the Bureau’s Rule, new § 1003.4(a)(1)(i) addresses the universal loan identifier
application process.

- Refinancing or applications for refinancing should be assigned a different ULI than the loan that is being refinanced.
- A financial institution with multiple branches must ensure that its branches do not use a single ULI to refer to multiple covered loans or applications.

**CFPB Rule for Universal Loan Identifier: Format Requirements**

In addition to requirements and restrictions for usage, the new CFPB Rule includes instruction regarding the format of the new ULI to be used. The requirement for the ULI number incorporates a requirement to use the Legal Entity Identifier (LEI) as the first part of the new ULI number.

**About the Legal Entity Identifier**

The LEI portion of the ULI will also serve as a stand-alone requirement in the transmittal sheet and LAR in replacement of the current Reporter’s Identification Number (HMDA RID). Therefore, the LEI number will need to exist as its own data point(s), as well as a sub-part of any generated ULI number created by an organization.

The LEI is always a unique, 20-digit alphanumeric identifier associated with a single legal entity and is intended to serve as a uniform international standard for identifying participants in financial transactions. The LEI number is a product of a global initiative to uniquely identify participants in financial transactions. It is governed by the Global LEI Foundation (GLEIF), which was created by the Financial Stability Board and the Group of 20. It is currently planned or used in mandates across the world, to include that for reporting of Money Funds and certain derivative transactions in the US. The global registry ensures the data quality of the numbers and their metadata. There is one current registry in the United States, hosted at https://www.gmeiutility.org/. This utility was erected to support the CFTC requirements for money funds and derivatives. There are currently more than a dozen of these endorsed local utilities in operation across the world. The registries are operated on a cost recovery basis, with the current cost in the US for initial registration being $200 per LEI, with a $100 annual maintenance fee.

The expectations are that, over the course of time, these numbers will eventually be used to identify all parties to financial transactions. Information about these numbers and their metadata are publicly available through the utility/registry websites. Regulatory agencies may participate in data sharing in order to track and measure the activities of financial institutions across markets that are currently fragmented by region, product, and regulatory paradigm.

An LEI is always composed of 20 characters. (see chart to follow). The first four characters are that of the local registry that generated the specific LEI. It is followed by two zeroes. The next 12 characters are the number that is generated in that specific registry as belonging to a specific legal entity. The final two characters contain a check
The entire LEI is unique within the global system. Organizations should not use substrings of the LEI for identification, as this will not result in a unique number.

The check digit sequence used at the end is a mechanism designed to enable users to very quickly validate whether any particular number is valid. A number, especially a longer one, can become altered or garbled through copy or data entry errors, such as transposition, substitution, and circular shift errors. The practice of creating and using check digit sequences is based upon a standard distributed by the International Organization for Standardization (ISO), number ISO7064: Information technology—Security techniques—Check character systems. More information about the use of check digits can be found in the ULI formatting section below.

**Diagram: Internal Format of the Legal Entity Identifier**

The LEI is a 20-character field composed as follows:

**As published in ISO standard 17442**

| AN | AN | AN | AN | AN | AN | AN | AN | AN | AN | AN | AN | N | N |

**As currently specified by the Financial Stability Board LEI Implementation Group**

| N | N | N | N | 0 | 0 | AN | AN | AN | AN | AN | AN | AN | AN | AN | AN | AN | N | N |

- 1 – 4: Prefix allocated uniquely to each Local Operating Unit by the FSB
  - 5493 – DTCC/SWIFT
  - 5299 – WM Datenservice
  - 6354 – Irish Stock Exchange
  - 1392 – Palestine Securities Exchange
  - 7890 – Takasbank
  - 2534 – National Settlements Depository
  - 9695 – INSEE
- 5 – 6: Two characters reserved for expansion and set to ‘00’
- 7 – 18: Entity-specific portion generated and assigned by each Local Operating Unit
- 19 – 20: Two character check digit based on ISO/IEC 7064

---


7 ISO/IEC. Reference number ISO 17442:2012(E): Financial Services—Legal Entity Identifier (LEI)
About The Universal Loan Identifier

The Universal Loan Identifier (ULI) will be composed of the LEI plus a lender-generated number of up to 23 characters and a two-digit check digit sequence to create up to a 45 character ULI number required by the Rule.

Requirements for the format of the ULI⁸,⁹:

- The first 20 characters of the ULI shall begin with the financial institution’s Legal Entity Identifier¹⁰
- The LEI must be issued by:
  - A utility endorsed by the Global LEI ROC
  - A utility endorsed or otherwise governed by the GLEIF (or any successor of the GLEIF) after the GLEIF assumes operational governance of the global LEI system.
- Following the LEI, a sequence of up to 23 additional characters to identify the covered loan or application, which:
  - May be letters, numerals, or a combination of letters and numerals;
  - Must be unique within the financial institution; and
  - Must not include any information that could be used to directly identify the applicant or borrower (For the purposes of this requirement, borrower-identifying information includes, but is not limited to, the applicant’s or borrower’s name, date of birth, Social Security number, official government-issued driver’s license or identification number, alien registration number, government passport number, or employer or taxpayer identification number.¹¹)
  - Ends with a two-character check digit
- The resulting ULI can be up to 45 characters in length

If an institution’s loan number is less than 23 characters in length, the institution may choose to left-pad the number with zeroes to create a number that is 23 characters long, or simply leave the number as is, without padding. The validity of the check digit operation will not be affected by either choice. However, if an institution foresees needing to expand the number’s length at some time in the future, left padding would allow expansion in the future without affecting the integrity of numbers already issued.

---

⁸ Final Rule § 1003.4(a)(1)
¹⁰ Proposed § 1003.5(a)(3)
¹¹ Final 1003.4(a)(1)(i)-2
The resulting ULI would look like this:

![Diagram of ULI](image)

The final check digit sequence function should be applied to the entire 43-character identification string, to include the LEI and Lender Loan number segments.

### A Few Words on Check Digits

As mentioned in the section on the LEI, a check digit sequence is used at the end of a number to enable users to very quickly validate whether the number is valid. The check digit process is based on an ISO standard, and the same process used for the LEI will now also be applied to the ULI.

Validating and creating a check digit sequence are performed in very similar manners; it is essentially a mathematical operation performed on the number, based on a translation of the number’s characters according to a prescribed scheme. The check digit works in the manner described below.

First, each digit of the ULI is assigned a number according to the following table.

- A = 10  F = 15  K = 20  P = 25  U = 30
- B = 11  G = 16  L = 21  Q = 26  V = 31
- C = 12  H = 17  M = 22  R = 27  W = 32
- D = 13  I = 18  N = 23  S = 28  X = 33
- E = 14  J = 19  O = 24  T = 29  Y = 34
  Z = 35

Next, the check character system is applied, MOD 97-10. Modulo is essentially dividing the resulting alpha-translated number from the chart above by 97—the modulo result is the remainder from that division.
This can be done by using a MOD function (like the one in Excel for example) as “MOD (cell of entire ULI number, 97).” If the remainder is 1 (one), the number is valid. This system allows a user to validate any specific LEI as being a valid number without validating it against the registry. When using desktop tools, including Excel, to create your check digit or validate a ULI, be careful to consider the significant digit limitations of your chosen tool. (For instance, Excel cannot do an accurate modulo calculation on a 45-character string, especially with the alpha substitution.)

A similar process is used to create the check digit sequence for the number as well. After applying the same numbering grid as in the instructions above, two zeroes are added to the end of the number. Apply the mathematical function MOD (cell of ULI number with the two zeroes, 97). Subtract this result from 98. If the result is one digit, add a leading zero to make it two digits. This is the check digit sequence that should be used to create a full ULI according to the Rule (see example below).

Check digits are generally added to the end of a long number sequence in order to validate that any number that is received in the course of a transactional or reporting exchange is well formed and valid. The practice of adding a check digit sequence to the end of an identifier allows a user to perform a very straightforward mathematical function to the number, and very quickly ensure that the number is valid and has not been compromised due to transformation or transliteration of the number series through data entry or copy errors. For instance, a user interface could quickly perform a check digit calculation on an entered number, and determine that it was not valid, prompting the user to perform the necessary correction at the entry point. This type of function, conducted at key points in the business process, could enhance the data quality of the ULI as it is passed through the loan origination, fulfillment or other transactional processes.

This process should ideally be used to validate any ULI number that is received between two systems, typed into an interface, or received from another party as part of a loan purchase. Using this mathematical operation appropriately and in the right points in the process can ensure the data quality and accuracy of the ULI that is ultimately reported to the CFPB or other regulatory agencies.
Some Examples on Check Digits

Creating Your ULI Example  *(lengths shortened for illustration)*

- Assemble your LEI and Lender Loan number pieces:
  
  124700ABD47    |    4960ZF123

- Add two zeroes to the end of it:
  
  124700ABD474960ZF12300

- Convert it using the “decoder ring” table:

  A = 10  F = 15  K = 20  P = 25  U = 30
  B = 11  G = 16  L = 21  Q = 26  V = 31
  C = 12  H = 17  M = 22  R = 27  W = 32
  D = 13  I = 18  N = 23  S = 28  X = 33
  E = 14  J = 19  O = 24  T = 29  Y = 34  Z = 35

  124700101113474960351512300

- Divide by 97, keeping the remainder

  Remainder of
  \[(124700101113474960351512300) \div 97\]

  = 95

- Subtract that remainder from 98 (or use the modulo function)

  98 – 95 = 3

- Replace the two end digits of your initial string with these two digits (if the result is one digit, add a leading zero to make it two digits)

  124700ABD474960ZF12303

  *YOUR NEW ULI*
MISMO XML Representation of Legal Entity Identifier

Beginning with MISMO V3.3.0, the Legal Entity Identifier (LEI) value is stored in a GlobalLegalEntityIdentifier child element of the LEGAL_ENTITY_DETAIL element.

The XML sample below shows how to represent a LEI for MISMO 3.3.0 and later:

Version 3.3.0 and Later

```xml
<PARTY>
  <LEGAL_ENTITY>
    <LEGAL_ENTITY_DETAIL>
      <FullName>ABC Lender, NA</FullName>
      <GlobalLegalEntityIdentifier>10Bx939c5543TqA1144M</GlobalLegalEntityIdentifier>
    </LEGAL_ENTITY_DETAIL>
  </LEGAL_ENTITY>
</PARTY>
```

MISMO XML Representation of Universal Loan Identifier

The Universal Loan Identifier (ULI) is represented in the MISMO XML format in a similar manner to other loan identifier types. Beginning with MISMO V3.4.0 the enumerations for the LoanIdentifierType element were expanded to include a value of UniversalLoan for use with Universal Loan Identifiers. Earlier MISMO versions use a value of Other for the LoanIdentifierType and the LoanIdentifierTypeOtherDescription value would be set to UniversalLoan.

Below are examples, showing how to represent a ULI for MISMO 3.4.0 and for earlier versions:

Version 3.4.0 and Later

```xml
<LOAN_IDENTIFIER>
  <LoanIdentifier>124700ABD474960ZF12303</LoanIdentifier>
  <LoanIdentifierType>UniversalLoan</LoanIdentifierType>
</LOAN_IDENTIFIER>
```

Versions 3.0.0 thru 3.3.1

```xml
<LOAN_IDENTIFIER>
  <LoanIdentifier>124700ABD474960ZF12303</LoanIdentifier>
  <LoanIdentifierType>Other</LoanIdentifierType>
  <LoanIdentifierTypeOtherDescription>UniversalLoan</LoanIdentifierTypeOtherDescription>
</LOAN_IDENTIFIER>
```
**APPENDIX A: EVOLUTION OF THE MISMO STANDARDS**

This section provides an overview of the evolution of the MISMO Standards from Version 1 through Version 3.

**MISMO Version 1 - The Electronic Loan Package**

The MISMO participants developed the Version 1 architecture with the goal to create a model of an electronic loan package. This model categorized the data needed to process a single mortgage loan. The diagram below shows the high-level structure of a complete set of loan-package data.
In MISMO Version 1, each mortgage process area only used the MORTGAGEDATA sub-categories that included the data that was needed. For example:

- **Loan Application** - the initial loan application might use data from the APPLICATION, BORROWER, EXPENSE, INCOME, LOAN and PROPERTY elements.
- **Credit Reporting** - a credit reporting bureau might use data from the BORROWER, CREDITREQUEST, CREDITREPORT and CREDITSCORE elements.
- **Underwriting** - the underwriting process could also use ARMPAYMENTADJUSTMENT and ARM_RATE_ADJUSTMENT elements plus the data from the loan application process.

After the loan was processed, the data from all the processes could be combined into a single loan package.

As the MISMO Version 1 data structure was being built, each data point was assigned a name and a definition. These were organized into a Logical Data Dictionary (LDD). The MISMO Version 1 LDD contained 790 data points. Today’s Version 3 LDD has most of those and has expanded to around 4,000 data points.

Credit bureaus and mortgage insurers were some of the early adopters of the MISMO Version 1 standards, but MISMO participants were also expressing a need for transaction standards that would support the request and response of mortgage services, such as credit reporting, flood certification, property appraisal, title services, and automated underwriting. This led to the development of MISMO Version 2.

**MISMO Version 2 - Transactions**

For MISMO Version 2, the standards shifted towards transactional formats and away from the single loan package architecture of MISMO Version 1. Individual transaction standards were created for an expanding number of mortgage process areas such as automated underwriting, credit reporting, appraisals, flood certifications, mortgage insurance, and title insurance. Although the different transaction standards reused many of the same structures, they were organized to optimize each transaction.

While MISMO Version 1 used MORTGAGEDATA as the main or “root” element name, MISMO Version 2 transactions used different root elements, depending on their purpose. For example, the Automated Underwriting Standard (AUS) used LOAN_APPLICATION as the root element. The Credit Request transaction used the REQUEST_GROUP element as its root element. The Credit Response transaction used the RESPONSE_GROUP element as its root element.

Despite the structural differences, the data elements in each transaction were originated from the same MISMO Logical Data Dictionary that was used in Version 1. MISMO continued to add new data points and definitions as the scope of the data needed grew.
The following three examples show the structural differences of each MISMO Version 2 transaction.

**MISMO V2 Automated Underwriting Request Transaction**
When each of the MISMO work groups developed their transactions, they made the content of each data container match their processing needs. For example, the type of PROPERTY information needed for an automated underwriting transaction can differ greatly from that needed for a property appraisal transaction or a flood certification. There were 14 variations of the PROPERTY data container content. This meant that a loan originator who generated transactions for a variety of services needed to be aware of the differences in the MISMO PROPERTY data container for each transaction type.

Many lenders and mortgage service companies adopted MISMO Version 2. As this acceptance grew, MISMO decided to expand the usability of the MISMO standards by making the data content more uniform across all process areas. This led to the development of MISMO Version 3.
MISMO Version 3 – Reference Model, Messages, Deals, Documents

When MISMO began developing MISMO Version 3, there were several key requirements:

- Continue to use and expand the MISMO Logical Data Dictionary. Also expand its ability to identify deprecated data and their new equivalents, and how relationships between data containers are labeled.
- Create a common Reference Model that includes the common set of data attributes, data containers, and data relationships that are defined in the Logical Data Dictionary. This common model resolves the differences that existed between the transactional structures that existed in Version 2.
- Publish MISMO Version 3 using an industry standard “schema” format, which is supported by nearly all computer processing platforms. Previous MISMO versions used an older industry standard called a Document Type Definition (DTD). The schema format has a greatly expanded feature set, including “namespaces” - a feature which allows data from different sources to be combined in the same message.
- Store data as XML “elements” similar to method used in MISMO Version 1. This allows several new features that are not possible when data is stored as XML “attributes” as implemented in MISMO Version 2. These new features include the ability to encrypt individual data points, the ability to express some data in different languages, and the ability to express monetary data in a variety of currency types.
- Continue to support the use of MISMO for transactions that became widely adopted in MISMO Version 2.
- To meet this requirement, MISMO specified the use of the same root element name for all transactions. In Version 3, this element is MESSAGE. MISMO users can use MESSAGE for a variety of purposes, including: requesting a mortgage service, responding to a mortgage service request, or transferring a set of loan or mortgage-document data, just to name a few.
- Provide a uniform way of representing a document across all MISMO Version 3 Messages that includes support for PDFs, HTML and TIFF (fax) images of documents. MISMO Version 3 also provides the ability to include an audit trail for document changes and standard methods for mapping data to the document view.
- Add the ability to apply electronic signatures and tamper-evident seals to the documents, a function the mortgage industry has widely accepted to increase the confidence in the fidelity of electronic data.
- Add the ability to identify data fields that contain non-public personal
information that require special handing or encryption.

- Add the ability to support a loan, a deal with multiple loans, or multiple deals (deal sets), and the ability to describe and contain all of the data related to a single mortgage-backed security or group of securities. Although MISMO Version 1 provided a structure for holding data related to a single loan, Version 3 has expanded this capability greatly. This means that investors can view and analyze the content of a mortgage backed security data set, which will help increase their confidence and viability in this financial instrument.

- Provide the common reference model with a standard method of extending the standard with data that is not currently defined in the MISMO standard. This can be either industry-wide data that will be part of a future MISMO version, or company-specific data that would only be exchanged with specific business partners. This common extension method will allow for customizations to the standard, while still allowing the data to be validated using the MISMO standard reference model.

The following diagram shows the basic structure of a MISMO Version 3 MESSAGE. The main chapters of this guide describe this structure in more detail.
APPENDIX B: SECURITY PRINCIPLES

This chapter provides general guidance about security principles. 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.\textsuperscript{12} 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]\textsuperscript{13}

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 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

\textsuperscript{12} NIST SP 800-63
\textsuperscript{13} NIST SP 800-199
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]^{14}

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.^{15}

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).

---

^{14} NIST SP 800-199

^{15} 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.

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.

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 or entities (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 contains a description, business use case, security requirements, and possible security recommendations. The possible security recommendations 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.
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.

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

3DES
Also referred to as triple DES, 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 second key, and the resulting cipher text is again encrypted with a third key).

AES
Short for Advanced Encryption Standard, a symmetric 128-bit block data encryption algorithm.

Asymmetric encryption
In deploying asymmetric data encryption, the use of public and private key pairs to encrypt and subsequently de-crypt the data is required. Use of these pairs of keys can provide for both confidentiality and authentication.

Digital signature
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.

DMZ
Short for demilitarized zone, a computer or small sub-network 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.

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 \textit{plain text}; encrypted data is referred to as \textit{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 \textit{between} local area networks (LANs) \textit{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 \textit{hash values} for accessing data or for security. A hash value (or simply \textit{hash}), also called a \textit{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 \textit{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.

PGP  
Pretty Good Privacy, abbreviated as *PGP*, 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.

Pharming  
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.

PKI digital certificate  
Short for *public key infrastructure*, 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.

RSA public keys  
A 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.

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.

<table>
<thead>
<tr>
<th>Secure FTP</th>
<th>See SFTP/SSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security domain</td>
<td>The subset of data required by a party to fulfill their respective portion of the request.</td>
</tr>
<tr>
<td>SFTP/SSH</td>
<td>In computing, <strong>Secure Shell</strong> or <strong>SSH</strong> 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 <strong>secure FTP or SFTP</strong>.</td>
</tr>
<tr>
<td>SSL</td>
<td><strong>SSL</strong>, Short for <strong>Secure Sockets Layer</strong>, 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>Symmetric encryption</td>
<td>In deploying data encryption, the use of a key to decrypt the data is required. When utilizing a symmetric algorithm, both the sender and receiver of the data share a common key.</td>
</tr>
<tr>
<td>TLS</td>
<td>Short for <strong>Transport Layer Security</strong>, a protocol that guarantees privacy and data integrity between client/server applications</td>
</tr>
</tbody>
</table>
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.