SMART DOC®
IMPLEMENTATION GUIDE
2.0

May 15, 2019

The following is applicable to the SMART DOCUMENT® IMPLEMENTATION GUIDE, Version 2.0, dated May 15, 2019.

Chapter 1.1: Introduction

This chapter provides some basic information about what is included in the Implementation Guide (I-Guide) along with instructions on how to use it

Version 2.0

Revision History

Version	Date	Change
2.0	02/12/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Overview

The Implementation Guide (I-Guide) describes the creation and life cycle of a SMART Doc®. It is not a step-by-step "how to" handbook on how to create a single SMART Doc but rather it is a reference resource providing business context and technical guidance along with illustrative examples for a variety of implementations of SMART Docs. The I-Guide is complementary to the SMART Doc 1.0 specification.

The SMART Doc Specification was designed to address a wide range of possible implementations. Implementation choices are inherent to this flexible design. This I-Guide describes requirements and considerations for implementations of the SMART Doc that are not specifically addressed in the specification.

This version of the I-Guide addresses the implementation of electronic Notes. Future versions of this guide will include other document types such as documents that are recorded.

The I-Guide consists of 10 Sections:

Section 1 provides introductory materials for the I-Guide, including this chapter.

Section 2 is divided into chapters pertaining to a particular phase or state in the life cycle of a SMART Doc.

Section 3 describes how to link the Data and View sections of a Category 1 SMART Doc, and how to convert the data, if the representations differ.

Section 4 covers common types of signatures within a SMART Doc.

Section 5 provides requirements for the view section and for implementing different types of views, such as tagged XHTML views and image views.

Section 6 describes the various categories of SMART Docs.

Section 7 covers the required data for the Data sections of specific document types, such as the Note.

Section 8 describes transporting SMART Docs and includes packaging and enveloping.

Section 9 describes specific requirements for the audit trail

Section 10 provides a list of references to external documents that are mentioned in this Implementation Guide.

Appendix A of Version 1 of this Implementation Guide provided a Glossary. This has been superseded by the eMortgage Glossary, which is available at http://www.mismo.org/Documents/MISMO/Documents/eMortgageGlossary_09212016.docx.

Appendix B of Version 1 provided sample SMART Doc files for an eNote in each state in the lifecycle. They were not very useful in PDF format, so the samples in XML format have been included in the Zip Archive distribution file for this Version 2 of the Implementation Guide.

Appendix C of Version 1 provided sample SMART Doc files for Category 1 – 5. They were not very useful in PDF format, so the samples in XML format have been included in the Zip Archive distribution file for this Version 2 of the Implementation Guide.

Appendix D of Version 1 provided examples of an encoded package and a SMART Doc package. They were not very useful in PDF format, so the samples in XML format have been included in the Zip Archive distribution file for this Version 2 of the Implementation Guide.

Appendix E of Version 1 provided various DTDs needed for SMART Doc implementation. They were not very useful in PDF format, so the files in DTD format have been included in the Zip Archive distribution file for this Version 2 of the Implementation Guide.

Each chapter is subdivided into sections identified by headings designed to allow the user of the I-Guide to find information quickly and easily.

Below is a list of the sections included in each chapter along with a high-level description of the type of information found in each section:

Version: A number used to identify each published revision to the Chapter

Relevant Specifications: A list of the versions of the MISMO specifications to which the chapter provides guidance.

Overview: A one- to two-paragraph description of contents of the subject I-Guide chapter.

Pre Conditions: A description of any specific information or process that must exist before the application of the chapter's guidance. For example, the pre-conditions might include the state of the SMART Doc (such as Signable) and the document categories (1 or 2 only).

Post Conditions: A description of specific information or processes that exist after the application of the chapter's guidance. For example, the post-conditions might include the state of a SMART Doc (Signed) or the addition of text electronic signatures.

Business Context: The high-level business case and requirements that the implementation will satisfy.

Scenario: A brief storyboard or process flow for which this specific chapter of implementation guidance applies.

Technical Guidance: The detailed technical guidance for implementation, including XML illustrations, specific steps or instructions to follow, and required attributes.

Checklist: A checklist of items to review in order to complete the implementation or process step.

XML Structures: A listing of the XML structures used.

Known Issues: This section discusses or identifies any known issues that may arise from the application of the guidance included in the chapter.

Other References: This section refers the reader of the chapter to other related documents or I-Guide chapters needed to better understand the implementation guidance included in the chapter.

Not every chapter will include all of these sections and there may be chapters that include special sections pertaining only to the topic addressed.

Examples, Checklists and Notes

The I-Guide chapters use three formatting tools to enhance their readability and value – Examples, Checklists, and Notes:

Examples are used to further illustrate or clarify a particular implementation point.

Checklists are provided near the end of each I-Guide chapter as a summary of the mandatory implementation steps or required processes.

Notes are marked at relevant points in a step, example, or XML structure to indicate optional steps or processes that the implementer may choose in order to satisfy their specific business process or requirements.

Please note that the I-Guide includes examples for illustrative purposes only. Although an example may show a particular data or signature type, there may be a number of options for you to implement based on your business process.

For instance, the examples in the I-Guide show parsed name and address data fields. However, you may elect to implement the name and address unparsed.

The eNote is used as an example throughout this version of the I-Guide for several reasons:

The eNote is a transferable record and may be the most important document in a mortgage loan package
The eNote is the SMART Doc upon which the industry has done the most work and developed the greatest expertise
The eNote is the SMART Doc that is likely to be the most familiar to the users of the I-Guide since a Note is part of each mortgage transaction and has few differences from jurisdiction to jurisdiction.

Although this I-Guide includes some information on Recordable and Recorded documents, future editions will include more information specific to security instruments and other specific document types as defined by the industry.

Definitions

Some of the terms used throughout the I-Guide are specific to the world of SMART Docs and electronic mortgages. Because electronic mortgages and SMART Docs are new concepts to most of us, we have included a list of the more commonly used terms along with their definitions in the Glossary, which is in Appendix A.

Additional Guidance Related to SMART Documents

In addition to the SMART Doc Implementation Guide, it is recommended that implementers consult The Standards and Procedures for electronic Records and Signatures (SPeRS) document. Sponsored by the Electronic Financial Services Council (EFSC), the SPeRS document provides a wealth of information on the execution of electronic financial transactions. It is not a technical manual and as such complements the SMART Doc Implementation Guide. The SPeRS standards are applicable to a broad range of electronic financial transactions, including eMortgages.

SPeRS provides guidance on delivering information and obtaining signatures as well as recommendations on how a system should interact with end users. The standards are based on legal requirements and best practices. They are intended to enhance the legal reliability and sufficiency of systems. More information is available at http://www.spers.org.

Special Considerations for Electronic Mortgage Documents

MERS® eRegistry Requirements

To comply with eSignature laws and to ensure that the single authoritative copy of each eNote can be identified, the industry is working to establish a National eNote Registry, known as the MERS® eRegistry. The National eNote Registry is being developed and will be maintained by MERS. The Registry concept requires that eNotes include special language – an eNote Clause – that points a holder to the registry, which will provide information on the controller of the transferable record. See chapter 5.4 on the specific language required for eNotes. The controller will identify the single authoritative copy of its eNotes. Please see chapter 8.3, The MERS® eRegistry and SMART Docs, for further information on the National eNote Registry.

Individual Trading Partner Requirements

Many trading partners may have additional SMART Doc specifications beyond those illustrated in this I-Guide. Consult your trading partner's requirements before implementing SMART Docs as each implementation may vary.

Investor' Requirements

A secondary market investor may have additional SMART Doc or eNote specifications beyond those illustrated in this I-Guide. You should review and understand your investor's requirements before implementing SMART Docs as each investor's requirements may vary.

Tamper Evident Signature Requirements

As with a paper note, an eNote must be free of unauthorized alterations in order to stave off challenges to its validity. Once all data and electronic signatures have been applied to an eNote, a tamper-evident seal must be applied to the view of the document by means of a digital signature. For more detail see chapter 4.3 on Tamper Evident Signatures.

Feedback and Comments

As we learn more about SMART Docs we will continue to improve and revise this I-Guide to incorporate what we've learned. We need your help to make sure that this I-Guide includes the best possible implementation guidance.

Please send us an e-mail at info@mismo.org if you have:

-- Suggestions on ways to improve it
-- Other implementation approaches you'd like to see included, or
-- Corrections to the content

We will try to incorporate your feedback and comments when we publish updates and revisions to this I-Guide.

Chapter Highlights

Chapter 1.1 – Introduction: this chapter.

Chapter 2.1 – Unpopulated SMART Doc: illustrates the steps to create Header, an unpopulated Data section, and an unsigned and unpopulated View section of a SMART Doc. It also provides instructions for initiating the Audit Trail section.

Chapter 2.2 – Populating a SMART Doc: includes instructions for adding data and mapping that data to a view so that there is a complete and readable document ready for further processing. The chapter provides technical guidance for populating the SMART Doc's Header, Data, and View sections.

Chapter 2.3 – Making a SMART Doc Signable: describes how to prepare a SMART Doc for electronic signing by adding the XML elements to both the Header and View sections. The chapter includes step-by-step technical guidance for adding the SIGNATURE_MODEL, SIGNATURE_TARGET, SIGNATURE_SECTION, and SIGNATURE_AREA elements to the document as well as updating the document state and audit trail.

Chapter 2.4 – Signed: describes how to transition a Signable SMART Document to a Signed SMART Doc. A signed SMART Doc contains the signatures of all signers (with the exception of the Recorder) and is tamper-evidence sealed, i.e., digitally signed. The chapter covers how relevant portions of the HEADER, VIEW, and AUDIT_TRAIL sections are updated as individual signatories apply their signatures. Additionally, the chapter describes implementation steps after all signatures have been applied in the SIGNATURES section.

Chapter 3.1 -- Linking the data with the view for XHTML Documents: covers requirements for linking the data section of a SMART Doc to an XHTML view. It specifically covers the <MAP> section and <ARC> elements with the data section and unique identifiers within the view section

Chapter 3.2 -- Converting data to different representations in the View: covers implementation information that exists in different forms in the data section and the view section. For instance, the chapter describes the use of the CONVERT tag and a mask to show a numeric dollar amount (100000) in the Data section as text (One Hundred Thousand Dollars) in the view.

Chapter 3.3 -- One to Many Values: describes how to implement data values that have two or more representations in the view.

Chapter 3.4 -- Operators: describes how to implement several input data fields to a single output presentation field or one data input field to many output presentation fields. Additionally view constructs such as radio buttons or checkboxes are discussed. The primary elements discussed in his chapter are the <CONVERT>, <AND> and <OR> elements.

Chapter 4.1 -- Electronic Borrower Signatures: describes how to create signed documents by applying electronic text and image signatures for Borrowers. The chapter does not include implementation guidance on digital signatures, which are discussed separately in Chapters 4.2 and 4.3 (See below).

Chapter 4.2 -- Digital Borrower Signatures: explains how a digital signature should be applied to a SMART Doc for the purpose of creating a borrower's signature consistent with the W3C XML digital signature standards and how to create a View section that refers to the Borrower's digital signature.

Chapter 4.3 -- Tamper Evident Signatures: describes how to implement a tamper seal for SMART Docs in a manner that provides a high degree of message or document integrity

Chapter 4.4 -- Applying Digital Signatures for Authentication and Integrity Validation. This chapter describes how apply digital signatures to SMART Docs for both document authentication and integrity.

Chapter 5.1 -- Implementing Tagged Views: describes how to implement XHTML views and specific consideration and requirements for tagged views.

Chapter 5.2 -- Types of Image Views: and how to implement describes how to implement image views and specific consideration and requirements for image views.

Chapter 5.3 -- Requirements for XHTML Views: describes specific implementation requirements for XHTML views.

Chapter 5.4 -- Special Language for eNotes: describes requirements for the view specific to electronic notes.

Chapter 6.1 -- Document Categories: describes the characteristics of the various SMART Doc Categories. Within each Category's description are examples of probable uses for the various categories.

Chapter 7.1 -- Setting the correct document type in the HEADER: describes specific requirements in the header for the various document type (note, deed of trust, mortgage, etc.)

Chapter 7.2 -- Custom Data: describes implementation choices when adding custom organization specific data to SMART Docs.

Chapter 7.3 --Other DTDs: describes implementation choices when using DTDs other than the provided MISMO Closing DTDs and the PRIA DTD.

Chapter 7.4 -- Data Requirements for SMART Docs Fixed and Adjustable Notes: lists the data points for the uniform instruments for fixed and adjustable rate eNotes.
This chapter has been replaced with an Excel document that provides data mapping and format guidance for the eNote: SMARTDoc_1_02_eNote_Map_and_Format.xlsx, included in this Implementation Guide.

Chapter 8.1 – Packages: describes how to collect a set of SMART Docs and related files, such as signatures, into an electronic package.

Chapter 8.2 – Enveloping: requirements and implementation steps to place an eMortgage package into the MISMO Envelope for transport.

Chapter 8.3 -- MERS eNote Registry and SMART Docs: describes the MERS eNote Registry and SMART Doc requirements for the Registry.

Chapter 9.1 -- Audit trail: describes specific requirements regarding the implementation of the audit trial including the format of data and time stamps.

Appendices:

A	Glossary
B	Document Type Samples
B.1	Note Samples:
B.1.1	Sample Unpopulated 3200 Note (xml file also provided)
B.1.2	Sample Populated 3200 Note (xml file also provided)
B.1.3	Sample Signable 3200 Note (xml file also provided)
B.1.4	Sample Text Signed 3200 Note
B.1.5	Sample Image Signed 3200 Note (xml file also provided)
B.1.6	Sample Digitally Signed 3200 Note
B.1.7	Sample Tampersealed 3200 Note
B.1.8	Sample Notarized Signable 3200 Note
B.1.9	Sample Notarized Signed 3200 Note
B.1.10	Sample Notarized Tampersealed SMART Doc
C	Category Samples
C.1	Sample Category 1 3200 Note
C.2	Sample Category 2 3200 Note
C.3	Sample Category 3 3200 Note
C.4	Sample Category 4 3200 Note C.5 Sample Category 5 3200 Note

Chapter 2.1: Unpopulated SMART Docs

This chapter describes how to create a new SMART Doc® in the unpopulated state.

Version

2.0

Revision History

Version	Date	Change
2.0	02/12/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification v1.02

Overview

A single SMART Doc will progress through various states during its lifecycle. This section describes how to create the first state in the lifecycle, an unpopulated SMART Doc. An unpopulated document represents an unfilled form. It has a header, an audit trail and depending on the category, may have an unpopulated data section, and an unpopulated and unsigned view.

Pre Conditions

Document State: Not applicable
Document Categories: All

Post Conditions

Document State: Unpopulated

Business Context

The SMART Doc specification defines a standard for the representation of mortgage documents in an electronic format. A SMART Doc is defined as a single electronic document that binds together data and presentation along with other (optional) information needed to store electronic (and/or digital) signatures and allows for the application of a tamper evident seal to insure authenticity. A single SMART Doc will progress through various states during its lifecycle. Those stages are defined as Unpopulated, Populated, Signable, Signed, Recordable, Recorded, Exported, and Voided. Each of the states is described in a separate chapter in this I-Guide.

Not only can the SMART Docs be classified by the lifecycle stage, they are also classified into one of five categories depending on how much of the optional data/format is included in the document. The categories allow flexibility in how rigorous the implementation of a specific SMART Doc needs to be to accomplish the loan-processing objective. A description of the categories of SMART Docs is provided in Chapter 6.1, Document Categories.

A Category One document is the highest level that supports tamper evident sealing and links the data section to the presented view. A Category Two document is similar, except there is no separate "data" section and therefore can be no validation. A Category Three document is similar to a Category Two, however there is a separate data section and the viewing format of the document is image based as opposed to XHTML. Similarly, a Category Four document is essentially the same as a Category Three; however, it does not include a data section. Finally, a Category Five document is one that contains data but does not include a means to view the document (no image or XHTML). Please see Chapter 6.1 for details.

Furthermore, a set of SMART Docs, which may be collected together in an eMortgage Package, may consist of documents from more than one category. For example, a closing package may have a Category One Note document, while the other documents are represented by other categories. For information on eMortgage Packages, see Chapter 8.1.

This chapter describes the very beginning of the lifecycle, the unpopulated state (regardless of category). The unpopulated state can be thought of as simply the "empty form" or template. The View does not have any data, only blank lines that represent where the data will be applied. The unpopulated state MAY include an empty data section, ready for data, or it may not include a data section, i.e. it can be omitted entirely.

It should be noted that in some systems, a SMART Doc might be created as an unpopulated document with data. This effectively simultaneously builds an unpopulated and populated SMART Doc. It is not necessary to create every state independently; however, the audit trail must acknowledge each and every state transition as an entry as well as the attribution of who moved the document from state to state. If the SMART Doc is built directly in the populated state, the audit trail must include a record of the creation of an unpopulated state, immediately followed by a record of the populated state. The date and time of creation may be the same value. For information on implementation requirements for the <AUDIT_TRAIL> please see chapter 9.1. For technical details on populated documents see Chapter 2.2, "Populating a SMART Doc."

Scenario

The SMART Doc template is authored (for instance by a document preparation company) in the unpopulated state. Once created it can be used on any system that supports SMART Docs and can merge/add in transaction data. When a SMART Doc includes data, it is moved into a different state, known as a Populated SMART Doc. For information on creating SMART Docs with data, see Chapter 2.2 Populating SMART Docs.

Technical Guidance

An unpopulated SMART Doc must contain the `<SMART_DOCUMENT>` element and the following two sections:

- HEADER
- AUDIT_TRAIL

Depending on the category, the Unpopulated SMART Doc must also contain at least one of the following sections:

- DATA
- VIEW

See business context above for a description of categories, or the SMART Doc Specification for a more technical description.

Step One: Create the SMART_DOCUMENT Element

The `_ID` attribute is optional and is used to identify the SMART Doc. It is recommended to assign a unique XML identifier to the SMART Doc.

The `_ID` attribute should not be confused with the `PopulatingSystemDocumentIdentifier` attribute. This identifier MUST be unique at the time of population; that is when the document moves to the Populated state.

The `PopulatingSystemDocumentIdentifier` attribute is required and is an identification string for the document. The purpose of the unique identifier is to create a single, traceable instance of the document in the "populating" system.

```
<SMART_DOCUMENT MISMOVersionIdentifier="1.0"
PopulatingSystemDocumentIdentifier=" "
_ID="AnXML_ID">
```

The `PopulatingSystemDocumentIdentifier` is required. At the time of creating an unpopulated document, its value may be blank. Upon population, the system adding the data to the document would assign a unique identifier.

The `<SMART_DOCUMENT>` element is required for all SMART Doc categories.

Step Two: Create the Header Section

Create the <HEADER> section:

- Create the <HEADER> element
- Create a unique `_ID` for the <HEADER>. The `_ID` attribute is of attribute type ID in the DTD and its value must conform to XML naming syntax for ID attributes
- Create the required `<DOCUMENT_INFORMATION>` element
- `<_StateType>` attribute should be set to Unpopulated in `<DOCUMENT_INFORMATION>`
- Additionally, be aware that there are other required attributes in the `<DOCUMENT_INFORMATION>` element. The values of these attributes are dependent on the _Type attribute:
- For eNotes the _Type attribute must be set to “Note”. Other document type have other values. Please see Chapter 7.1, Setting the Correct Document Type in the HEADER, for further information
- `NegotiableInstrumentIndicator, _Type and MustBeRecordedIndicator`.

```
<HEADER _ID="FNMA_Sample_Header_3200">
   <DOCUMENT_INFORMATION _Type="Note"

_StateType="Unpopulated"

NegotiableInstrumentIndicator="True"

MustBeRecordedIndicator="False" />
</HEADER>
```

The <HEADER> element is required for all SMART Doc categories.

Step Three: Optional: Create the Data Section

The <DATA> element is not required for all SMART Doc categories. Please refer to the Specification for details. For information on creating SMART Docs with data, see Chapter 2.2: Populating SMART Docs. If you are not creating a data section, skip to Step 4.

If you are adding a `<DATA>` section, you must provide a unique identifier for the data section in the `_ID` attribute. The `_ID` attribute is of attribute type ID in the DTD and its value must conform to XML naming syntax for ID attributes then the minimal requirement in the Unpopulated state is:

```
<DATA _ID="FNMA_Sample_Data_3200">
      <MAIN>
            <LOAN MISMOVersionIdentifier="2.3">
                  </LOAN>
      </MAIN>
</DATA>
```

This section may contain the individual data elements with or without their values. In the unpopulated state, the individual elements are not required and may be left out. For example, you may include an empty borrower section for each borrower, or not include Borrower elements at all. If data elements are created with their values, then the Audit Trail must reflect the creation of the form in unpopulated state first, and also be moved directly to a "populated" state (as described in the Business Context section above).

```
<BORROWER BorrowerID="B1" _FirstName=""
_MiddleName="" _LastName=""
_HomeTelephoneNumber=""/>
```

Additionally, depending on the Category of SMART Doc you are creating you may also need to create the following two sections inside the `<DATA>` section:

- `<MAP>`: Contains the `<ARC>` elements that map the data element to its corresponding location in the `<VIEW>` and also controls the formatting of the data. Please refer to Section 3: Data Mapping and Conversions, for more details.
- `<CUSTOM>`: This element can be used to define any custom data elements that are outside the SMART Doc DTDs. The majority of the time, this will be left empty. Please note that if you use the
  `<CUSTOM>` element, anyone using your SMART Doc will need your DTD to support it. See Chapter 7.2 for information on customizing data.

The following is an example of a data section defined without any data defined for the execution date:

```
<DATA _ID="FNMA_Sample_Data_3200">
            <MAIN>
                  [...]
                  <EXECUTION _Date="" _City=""
_State=""/>
                  [...]
            </MAIN>
            <MAP
TargetIDREF="FNMA_Sample_View_3200">
                  …
                  <ARC
      DataLinkDescription="//EXECUTION/@_Date"
                  ViewLinkDescription="id(EXECUTION-
_Date)">
                  </ARC>
                  …
            </MAP>
            <CUSTOM>

<SYSTEM_SPECIFIC_DATA/>
                             </CUSTOM>
      </DATA>
```

For information on linking SMART Doc data with the view section, see Chapter 2.2, Populating SMART Docs and Section 3, Data Mapping and Conversions.

Linking, converting and mapping the data with the view for XHTML documents is covered in Section 3.

Step Four: Create the View Section

The `<VIEW>` section contains the visual representation of the document. If it is a tagged view, then it also contains the location where the data values are merged in. It is possible for a SMART Doc View to contain

a) an inline, tagged format (e.g. XHTML)
b) and inline encoded image (e.g. TIFF, PDF, JPG)
c) or an external, image (e.g. TIFF, PDF, JPG)
d) no view section at all

For the purposes of this document, we will refer to inline, tagged views only.

The `<VIEW>` element must contain an `_ID` attribute, which uniquely identifies the VIEW section of the SMART Doc as a digitally signed part of the document. The `_ID` attribute is of attribute type ID in the DTD and its value must conform to XML naming syntax for ID attributes. `<VIEW>` element MUST contain the `_MIMETypeDescription` attribute with the value "`text/html`" and a `_TaggedIndicator` attribute set to "`True`".

```
<VIEW _ID="FNMA_Sample_View_3200"
_MIMETypeDescription="text/html"
_TaggedIndicator="True">
            <html
```

```
xmlns="http://www.w3.org/1999/xhtml">
                 …
                 <span class="dataEntered"
id="EXECUTION_Date">_____, ___</span>
                 …
     </html>
</VIEW>
```

The `<VIEW>` element is not required for all SMART Doc categories. Please refer to the Specification for details.

Step Five: Create the Signature Information (Optional)

The `<SIGNATURE_MODEL>` contains information about the signers and the `<SIGNATURE_SECTION>` contains electronic signatures. The creation of signature line elements is not required in Unpopulated SMART Docs. However, to keep control of the placement of signatures, you may find it useful to include the signature elements without their respective values. For further information on adding signature lines and signature information, see the chapter on the signable state of the SMART Doc, Chapter 2.3.

Step Six: Create the Audit_Trail Section

The `<AUDIT_TRAIL>` contains a record of each operation performed on the document. For the Unpopulated state, the author should create an entry of the form:

```
      <AUDIT_TRAIL>
            <AUDIT_ENTRY _DateTime="2002-07-
30T20:30:50Z" _PerformedByName="Document Prep
Company" _ActionType="Unpopulated"/>
      </AUDIT_TRAIL>
```

The entry in the audit trail must be included for the unpopulated state, even if your system bypasses the unpopulated state and creates SMART Docs in the populated state.

The recommended format for the `_DateTime` attribute is Universal Coordinated Time (UTC). The `_ActionType` must indicate the state of the document. The `<AUDIT_TRAIL>` element is required for all SMART Doc categories. Further information about the `<AUDIT_TRAIL>` is provided in Chapter 9.

Step Seven: Validate the Document

There are two validation steps:

- XML validation against the relevant DTDs

- ƒ For XHTML tagged views, additional validation is required.
  The additional validation must ensure the `<DATA>` and the `<VIEW>` sections match. XML DTD validation will not catch these errors

Note: When validating <Data> and <View> sections, there is not always a one-to-one correlation. The view section may show a single data element, e.g. January 1, 2003 and the data section may have it divided into three data elements: Month, Day and Year. The ARC section defines the relationship between the single view element and the multiple data elements. So it is important to utilize the ARC section to properly validate.

For examples, please reference Section 3 for data mapping and conversions.

Checklist

9 Make sure that the root level `<SMART_DOCUMENT>` and the sections required for the document category you are creating are present `(HEADER, DATA, VIEW and AUDIT_TRAIL)`. The `<SIGNATURES>` section is optional in this state.

9 Check `<HEADER>` element _`StateType="Unpopulated"`

9 Check the `<HEADER>` section and verify the other required attributes exist.

9 If you are implementing a Category 1, 3, or 5 SMART Doc, confirm existence of the minimal `<MAIN>` section elements inside the `<DATA>` section (with empty elements for data that will eventually be merged into the document), if your document category requires it.

9 If you are implementing a Category 1 SMART Doc check the `<MAP>` section inside `<DATA>` and ensure the `<ARC>` elements map to their corresponding `<VIEW>` elements – this is required for external validation of the document, i.e. non-XML, or DTD validation

9 Confirm the `<VIEW>` section contains the fixed text and the data elements.

9 Confirm an `<AUDIT_TRAIL>` entry exists for the unpopulated state. If initial creation of the document bypasses the unpopulated state and moves directly to a populated state, this entry is still required. Check that the date and time are in UTC.

9 Validate document: Perform both XML DTD and external validation. External validation needs to reference the <ARC> section to determine the relationship between <DATA> and <View> elements being compared.

XML Structures Used

```
<SMART_DOCUMENT>
<HEADER>
<DATA> (<MAIN>, <ARC>, <MAP>)
<VIEW>
<AUDIT_TRAIL>
<SIGNATURE_SECTION>
```

Whether a specific XML element is used or not depends on the document category.

Known Issues

The SMART Doc may never “exist” as an unpopulated state in your system. It is not necessary to create every state independently; however, the audit trail must acknowledge each and every state transition as an entry as well as the attribution of who moved the document from state to state.

If the SMART Doc is built directly in the populated state, the audit trail must include a record of the creation of an unpopulated state, immediately followed by a record of the populated state. The date and time of creation may be the same value.

Other References

See Chapter 2.2, Populating SMART Docs, Chapter 2.3, Signable SMART Docs, and Section 3 Data Mapping and Conversions.

See Chapter 5.4, eNote Language in the View, for detail on the language needed for the eNote.

For all other references, see Chapter 10: References

Chapter 2.2: Populating a SMART Doc

This chapter describes the steps required to add data to an unpopulated SMART Doc®.

Version

2.0

Revision History

Version	Date	Change
2.0	02/15/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc® Specification v1.02 with references to Closing and PRIA DTD.

Overview

A single SMART Doc will progress through various states during its lifecycle. This section describes how to create the second state in the lifecycle, a populated SMART Doc. A populated document represents an unsigned form with all the values present. It has a header, an audit trail and depending on the category, may have a data section, and an unsigned view with data. This chapter will provide instructions on how to add data and map it to a view, if necessary.

Pre Conditions

Document State: Unpopulated Document
Categories: All

Post Conditions

Document State: Populated

Business Context

The SMART Doc specification defines a standard for the representation of mortgage documents in an electronic format. A SMART Doc is defined as a single electronic document that binds together data and presentation along with other (optional) information needed to maximize its performance. A single SMART Doc will progress through various states during its lifecycle.

This chapter describes the second state of the lifecycle, the populated state. The populated state can be thought of as the “unsigned form.” It has data values applied to the blank lines in the form. The populated state MAY include a data section, or it may not. In some categories of SMART Documents, the data section does not exist; however, the view section in the populated state will have values present.

It should be noted that, in some systems, a SMART Doc might be initially created as an unpopulated document with data. This effectively simultaneously builds an unpopulated and populated SMART Doc in one step. It is not necessary to create every state independently. However, the audit trail must acknowledge each and every state transition as an entry as well as the attribution on who moved the document from state to state. Therefore, if the SMART Doc is built directly in the populated state, the audit trail must include a record of the creation of an unpopulated state, immediately followed by a record of the populated state. The date and time of creation may be the same value. For information on implementation requirements for the <AUDIT_TRAIL>, see Chapter 9.1.

In some workflows, the signers of the SMART Doc may be known at the time of population (see Chapter 2.3 for information on creating a Signable SMART Doc). This scenario effectively builds an unpopulated, a populated and a signable SMART Doc. Again, the audit trail must include a record of each state transition, even if the states occurred at essentially the same time.

Primarily, document preparation companies will be involved with documents in this state. Any parties that produce or populate documents in the lending process may be implementers. This may include a lender/LOS, doc prep provider, or a closing agent.

Scenario

A practical example of usage is the loan document preparation process. Once the unpopulated SMART Doc is created, the LOS usually forwards the 'closing' data to a doc prep vendor/system to populate the document. The document may then be forwarded and reviewed by several parties. It may be updated, redrafted and resubmitted multiple times. The final populated SMART Doc should move to the signing table where the signers of the document are added, but not the actual signatures. This is process is addressed in Chapter 2.3, Making a SMART Doc Signable.

Technical Guidance

A populated SMART Doc must contain the `<SMART_DOCUMENT>` element and the following two sections:

- HEADER
- AUDIT_TRAIL

Depending on the category, the Populated SMART Doc may also contain the following sections:

- DATA
- VIEW

Step One: Change the Header Section

- Required: `<_StateType>` attribute should be set to `Populated` in <DOCUMENT_INFORMATION>

9

```
<HEADER _ID="FNMA_Sample_Header_3200">
   <DOCUMENT_INFORMATION _Type="Note"
                         _StateType="Populated"
           NegotiableInstrumentIndicator="True"
           MustBeRecordedIndicator="False"/>
</HEADER>
```

- Optional: If you are implementing a Uniform Instrument that has a form number, set the _FormNumberIdentifier to the appropriate value. For instance, if you are implementing the MULTISTATE FIXED RATE eNOTE Freddie Mac/Fannie Mae UNIFORM INSTRUMENT 3200e, populate the _FormNumberIdentifier as follows:
- `<DOCUMENT_INFORMATION _FormNumberIdentifier="3200e" />`

Step Two: Optional: Create and/or Populate the Data Section

The <DATA> element is not required for all SMART Doc categories. A SMART Doc may contain one and only one data section or no data section at all. If present, the data section consists of the mortgage information. Please refer to the Specification or Chapter 6.1 (SMART Doc Categories) for details. If a data section is not required in your implementation, skip to Step 4.

The data section for the SMART Doc is defined by the MISMO Origination Workgroup. The Origination workgroup encompasses three process areas: application, underwriting, and closing. The eClosing focus group's scope is to define business data used in the loan closing process. This includes all aspects of closing data including doc prep, escrow, and closing transactions. This data is integral to the eMortgage efforts. It is important to realize that there is NOT a single DTD for each document type; i.e., there is not a “Note DTD”, an “Addendum DTD”, etc. The mortgage information in the data section will vary depending on whether the document is an adjustable note or a fixed note, and which United States State the SMART Doc was created for. The MISMO closing DTD defines a large set of mortgage application and closing data.

The SMART Doc Data DTD references a full set of Closing and eRecording data. This is accomplished by referencing three separate DTDs:

MISMO Closing Version 2.3: This DTD contains the current Closing dataset, including AUS (Automated Underwriting System) updates.

PRIA (eRecording) Version 2.4: This is the PRIA version used in SMART Doc 1.02. Version 1.2 is still in use by some parties.

Common Version 1.0: This DTD contains the elements that are common to both the Closing and PRIA datasets.

SMART Docs can be configured to include other custom DTDs or to exclude any of the three default DTDs. See Chapter 7.3, Using Other Data DTDs for information on the SMART Doc's use of the Closing DTD and for using data defined in other DTDs with SMART Docs.

Step 2a: Create the <MAIN> element

The `<DATA>` element MUST contain a `<MAIN>` element. The `<MAIN>` element contains the XML elements corresponding to the appropriate elements in the SMART Doc Data DTDs.

If you are adding a `<DATA>` section, the minimal requirement is:

```
<DATA _ID="FNMA_Sample_Data_3200">
      <MAIN>
            <LOAN MISMOVersionIdentifier="2.3">
            </LOAN>
      </MAIN>
</DATA>
```

Step 2b: Create the <LOAN> element

If you are creating a Category 1 SMART Doc, then you must populate the SMART Doc `<DATA>` section with each and every data point that is displayed in the `<VIEW>` section. See Section 7, chapter 7.4 and higher for information on data requirements for data requirements for each eMortgage document type. For instance, Chapter 7.4 describes the data required for eNotes.

The following is an example of a complete data section for a Multi-state, form number 3200, eNote:

```
      <LOAN MISMOVersionIdentifier="2.3">
        <_APPLICATION>
          <LOAN_PRODUCT_DATA>
            <LOAN_FEATURES
ScheduledFirstPaymentDate="10/01/01"
LoanMaturityDate="09/01/2031"
OriginalPrincipalAndInterestPaymentAmount="763.02">
              <LATE_CHARGE _GracePeriod="15"
_Rate="4.000"/>
              <NOTE_PAY_TO _StreetAddress="P.O. Box
3050" _City="Columbia" _State="MD"
_PostalCode="21045-6050"/>
            </LOAN_FEATURES>
          </LOAN_PRODUCT_DATA>
          <MERS
MERS_MINNumber="123451234512345123"/>
          <MORTGAGE_TERMS NoteRatePercent="8.625"
PaymentRemittanceDay="1"
OriginalLoanAmount="96500.00"
LenderLoanIdentifier="04405355"/>
          <PROPERTY _StreetAddress="748 N. Main
Street" _City="Louisburg" _State="NC"
_PostalCode="27549"/>
          <BORROWER BorrowerID="B123456789"
_FirstName="Richard" _MiddleName="R."
_LastName="Bradley" _HomeTelephoneNumber="123-456-
7890"/>
        </_APPLICATION>
        <_CLOSING_DOCUMENTS>
          <EXECUTION _Date="08142001"
_City="Louisburg" _State="NC"/>
          <LENDER _UnparsedName="Columbia National
Incorporated"/>
        </_CLOSING_DOCUMENTS>
      </LOAN>
```

The Data section of a populated, Category One SMART Doc must contain the individual data elements with their values. If the unpopulated SMART Doc created a data section without values, simply add the data in this step. Otherwise you must create each and every data point referenced in the view.

Step 2c: Create the <MAP> Section

If the SMART Doc is classified as a Category One document (the view is tagged with XHTML), the map or link between the data section and the XHTML view must be present in a `<MAP>` element. See Section 3 for further information on implementing the MAP section. The data section may also contain `<CUSTOM>` element. See Chapter 7.2 for instruction on how to implement custom data.

Step Three: Optional: Populate the View Section

The `<VIEW>` element is not required for all SMART Doc categories. Please refer to the Specification or Chapter 6.1: SMART Doc Categories for details. If you are not creating a view section, skip to Step 5.

SMART Docs with a `<VIEW>` section contain the visual representation of the document. There are several choices for types of views. For information on types of views and specific requirements on the view section, see Section 5. It is possible for a SMART Doc to contain an inline, tagged (e.g.

XHTML), or inline encoded image or an external image (e.g. PDF) view. For the purposes of this document, we will refer to inline, tagged views only.

The following is an abbreviated example of a tagged View section:

```
<VIEW _ID="FNMA_Sample_View_3200"
      _MIMETypeDescription="text/html"
      _TaggedIndicator="True">
      <html xmlns="http://www.w3.org/1999/xhtml">
                [...]
                <span class="dataEntered"
                      id="EXECUTION_Date">
                  _____, ___</span>
                [...]
           </html>
</VIEW>
```

For Category One SMART Docs, unique identifiers must exist in the view section to connect mortgage data in the tagged view (XHTML) (`<span>` or `<div>` elements) to data in elements and attributes in the `<MAIN>` element. For further information on linking the data and the view, see Section 3 of this guide.

Step Four: Optional: Create the Signature Information

The `<SIGNATURE_MODEL>` contains information about the signers and the `<SIGNATURE_SECTION>` contains electronic signatures. The creation of signature lines is not required in Populated SMART Docs. However, to keep control of the placement of signatures, you may find it useful to include signature information in the Populated document. For further information on adding signature lines and signature information, see the chapter on the signable state of theSMART Doc, Chapter 2.3.

Step Five: Update Audit Section

The `<AUDIT_TRAIL>` contains a record of each operation performed on the document. For the Unpopulated state, the author should create an entry of the form:

```
      <AUDIT_TRAIL>
           <AUDIT_ENTRY
             _DateTime="2002-07-30T20:30:50Z"
_PerformedByName="Lender"
            _ActionType="Populated"/>
      </AUDIT_TRAIL>
```

The entry in the audit trail must be included for the populated state, even if your system bypasses the populated state and creates SMART Docs in the signable state.

The recommended format for the `_DateTime` attribute is Universal Coordinated Time (UTC). The `_ActionType` must indicate the state of the document. The `<AUDIT_TRAIL>` element is required for all SMART Doc categories. Further information about the `<AUDIT_TRAIL>` is provided in Chapter 9.

Step Six: Validate Document

Populated SMART Docs should be validated against all relevant DTDs (e.g., SMART Doc DTD, Closing DTD, etc.) to ensure compliant XML.

Additionally, SMART Docs that make use of tagged views (i.e., Categories One and Three) should be externally authenticated to ensure that all data mapping is valid.

Step Seven: Optional: Tamper-Seal the Document

The producer of the Populated document may wish to digitally sign (or tamper-seal) the Populated SMART Doc in order to provide an authentication mechanism for other recipients of the document.

Refer to Section 4, Implementing Signatures, for further instruction.

Checklist

- ✓ Header element `<DOCUMENT_INFORMATION>` changed to signed state `<DOCUMENT_INFORMATION_StateType="Populated">`
- ✓ If the SMART Doc is Category One or Three, review to see that required data elements and attributes are populated in <MAIN> data section. See section 7 for the data requirements for specific document types.
- ✓ If the SMART Doc is Category One, check the <MAP> section inside <DATA> and ensure the <ARC> elements map to their corresponding <VIEW> elements – this is required for the external validation of the document, i.e. non-XML validation. See Section 3 for further information on mapping the data and view sections.
- ✓ Check that the `<VIEW>` section contains the fixed text and the data elements with unique identifiers.
- ✓ For Category One SMART Docs, ensure that for every data value in the view there is a corresponding data element in the data section.
- ✓ Check that the `<AUDIT_TRAIL>` entry exists for the Populated state.
- ✓ Validate document XML and external validation.
- ✓ Optionally tamper-seal the Populated document.

XML Structures Used

<HEADER>
<DATA> (<MAIN>, <ARC>, <MAP>)
<VIEW>
<AUDIT_TRAIL>
<SIGNATURES_SECTION>

Whether a specific XML element is used or not depends on the document category.

Known Issues

There are some assumptions about the SMART Doc categories as well as other states of the SMART Doc that need to be considered and compared when reviewing how to populate a document.

The SMART Doc may never "exist" as a populated state in your system. You may bypass the populated state and the end result may be a Signable SMART Doc. Or you may be building categories of SMART Docs that do not have a <DATA> section but contain a view populated with data.

It is not necessary to create every state independently; however, the audit trail must acknowledge each and every state transition as an entry as well as the attribution of who moved the document from state to state.

If the SMART Doc is built directly in the Signable state, the audit trail must include a record of the creation of a populated state, immediately followed by a record of the Signable state. The date and time of creation may be the same value.

Other References

See Chapter 2.3, Signable SMART Docs.

See Section 3 for information on mapping the data section with tagged views.

See Section 5 for information on the types of allowable views and requirements in the view section.

See Section 7 for specific information regarding requirements for the data section and customizing data and data DTDs.

See Chapter 5.4, eNote Language in the View, for detail on the language needed for the eNote.

See Chapter 10: References for references to other documents.

Chapter 2.3: Creating a Signable SMART Doc®

This chapter describes how to prepare a SMART Doc for electronic signing.

Version 2.0

Revision History

Version	Date	Change
2.0	02/15/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

MISMO® SMART Doc Specification v1.02

Overview

This section describes the steps required to transition a Smart Document from the populated to the Signable state. The objective of this step is to designate all signers for a particular document, their respective roles, and the type of signature each signer will be using. When all of the elements have been added for the document to be in that condition, it is referred to as a “Signable” document.

Pre Conditions

Document State: Populated

Post Conditions

Document State: Signable

Business Context

In the paper world, it is easy for anyone to apply a distinctive mark to acknowledge and/or be bound to the terms of a paper document. Such marks, in most cases can be attributed to a person or entity, and tampering can be detected to a very limited extent through forensic methods. Creating legally binding and enforceable signatures for electronic documents presents some different challenges, but is similar to the paper world in that signers mark the document with some distinctive mark.

Borrowers, sellers, witnesses, or notaries using an electronic signature as defined by UETA (State Law) and E-SIGN (Federal Law) ACT of 2000 may sign MISMO SMART Docs. In the current paper-based process a paper document is signable as soon as it is created with required content and printed on paper – no additional "processing" is required in order to allow a human to physically sign the piece of paper. With electronic documents, however (and specifically, with SMART Docs), certain modifications must be made to the underlying document infrastructure so that it "knows" how to receive, display and track the electronic signature.

For SMART Docs that contain image views, such as a PDF, the Signable state simply provides information that the document is ready to accept signatures.

In tagged XHTML views, the signature area for the signer is linked with the following: the role of the signer, his or her name, the placement of the signature line, a place holder for the order of the signing, and a reference to the entire view that will be displayed at the time of signing.

This chapter describes the technical aspects of linking information about an individual signer (borrower, notary, etc.) with a specific area in a tagged view section.

Scenario

Prior to the signing ceremony, the populated document needs to become ready for signing. The SMART Doc eNote is processed by the controlling system to prepare it for electronic signatures.

Technical Guidance

Preparing a SMART Doc for signing requires that certain XML elements be present in both the `<HEADER>` and `<VIEW>` sections. While these elements may be included in the SMART Doc at any time prior to reaching the signable state, the steps below assume that none of the necessary elements have been included.

Step One: Add <SIGNATURE_MODEL> element

The `<SIGNATURE_MODEL>` element refers to areas to be signed in signable documents and to electronic or digital signatures for signed documents. This element is contained in <HEADER> and in turn contains one `<SIGNER>` element for each signer role. The `<SIGNER>` element contains information about the type of signature (e.g., text, image, etc.) and the location of the signature in the <VIEW>. You will need to add the following attributes to the `<SIGNER>` element

- Set the `_RoleType` attribute to the role of the signer. It is required and has an enumerated list of values: `Borrower, Lender, Notary, ClosingAgent, Recorder, Grantor, Assignor, TamperSealer, Witness, Trustee, Director,` or `Other.`

- Use The `_RoleTypeOtherDescription` when `Other` is selected as the value for the `_RoleType attribute`. Fill in an appropriate string reflecting the role of the signer.
- The `SignatureType` attribute is required. Specify the type of signature (electronic modality, HTML) the signer will be using. It has the list of values: `Text, DigitalSignature, Object, and Image`.
- The `_SignatureOrderNumber` attribute is optional in the DTD however it MUST be included. When the tamper evident seal is applied as the final signature, this attribute must be set to indicate that the tampersealer was the last signer. For more information on tampersealing documents, see Chapter 4.3, Tamperseal Signatures. The signing order of the document may not be known at the time the document is moved to the signable state. In this case you should include the attribute and leave it empty. This step will provide a placeholder.
- Set the `TargetsIDREFS` attribute to an ID attribute of the content to be signed, such as the <VIEW> element. See step 2 for information on creating specific signature targets.
- Set the `SignatureIDREF` attribute to the identifier of the placeholder for the signature, the `<SIGNATURE_LINE> element`. See step 6 for information on applying the signature line.
- Set the `AreaIDREF` attribute to an ID attribute in a `<SIGNATURE_AREA>` element to which the `<SIGNER>` element is related. See step 4 for information on the `<SIGNATURE_AREA>` element.
- Set the `SectionIDREF` attribute to an ID of the `<SIGNATURE_SECTION>` element associated with the Signer. See step 3 for information on creating signature sections.

The following is an XML document fragment for the `<SIGNATURE_MODEL>` element:

```
<SIGNATURE_MODEL>
  <SIGNER _RoleType="Borrower"
          SignatureType="Text"
          TargetsIDREFS="FNMA_Sample_View_3200"
          SignatureIDREF="Borrower1SignatureLine"
          _SignatureOrderNumber="1"
          AreaIDREF="Borrower1SignatureArea"
          SectionIDREF="BorrowerSignatures"/>
```

```
    </SIGNATURE_MODEL>
```

Optional: Step Two – Add <SIGNATURE_TARGET> element

The `<SIGNATURE_TARGET>` element is used to denote the area that was viewed prior to signing. You may choose to wrap the entire view with a `<SIGNATURE_TARGET>` element. However, you may find the `<SIGNATURE_TARGET>` element useful for referencing the view without the signature sections:

```
   <body>
      <SIGNATURE_TARGET _ID="SignedContent02">
          <h1>NOTE</h1>
              […]
          <p> WITNESS THE HAND(S) AND SEAL(S) OF THE
UNDERSIGNED </p>
       </SIGNATURE_TARGET>
       <SIGNATURE_SECTION _ID="BorrowerSignatures">
               […]
</SIGNATURE_SECTION>
          <pre class="footer">MULTISTATE FIXED RATE
eNOTE<span class="nobold">--  Single Family --</span>
Fannie Mae UNIFORM INSTRUMENT
Form 3200e 4/02</pre>
</body>
```

When there are multiple signers, you may want to reference the body of the view section separately from the signatures. The first signer did not "see" the second signer's signature when he or she signed. The second signer, however, did "see" the first signer's signature. The `<SIGNATURE_TARGET>` element used in conjunction with the `TargetsIDREF` attribute in the `<SIGNER>` can be useful to handle sequential signatures and signatures that may occur over time.

Step Three: Add <SIGNATURE_SECTION> element(s)

The `<SIGNATURE_SECTION>` element is used to denote a section of the document that contains a collection of signature placeholders or electronic non-digital signatures. It is recommended that you set up `<SIGNATURE_SECTION>` elements for all the signers of a particular role; e.g., all borrowers.

Step Four: Add <SIGNATURE_AREA> element(s)

The <SIGNATURE_AREA> element identifies the location of each individual signature and contains any formatting and display elements related to the signature. One <SIGNATURE_AREA> element should be included for each <SIGNER> element in <SIGNATURE_MODEL>.

```
<SIGNATURE_SECTION _ID="BorrowerSignatures">
            <SIGNATURE_AREA _ID="Borrower1SignatureArea">
[…]
            </SIGNATURE_AREA>
            <SIGNATURE_AREA _ID="Borrower2SignatureArea">
              […]
            </SIGNATURE_AREA>
            <SIGNATURE_AREA _ID="Borrower3SignatureArea">
              […]
            </SIGNATURE_AREA>
            <SIGNATURE_AREA _ID="Borrower4SignatureArea">
              […]
- Borrower </p>
            </SIGNATURE_AREA>
          </SIGNATURE_SECTION>
```

(Optional) Step Five: Add the element <SIGNATURE_ ABOVE_LINE>

Add the element <SIGNATURE_ABOVE_LINE> in the <SIGNATURE_AREA> element for each signer role. The purpose this element is to capture any text you may wish to include above the signature line.

Step Six: Add the <SIGNATURE_LINE>

Add the <SIGNATURE_LINE> element in the <SIGNATURE_AREA> element for each signer role.

```
<SIGNATURE_LINE _ID="Borrower1SignatureLine">
____________________________________
</SIGNATURE_LINE>
```

(Optional) Step Seven: Add the <SIGNATURE_BELOW_ LINE> element

Add the <SIGNATURE_BELOW_LINE> element for each signer role to the <SIGNATURE_AREA> element. This element contains the text of the signer's name, and other identifying information such as the borrower's home telephone number.

```
<SIGNATURE_SECTION _ID="BorrowerSignatures">
```

```
            <SIGNATURE_AREA _ID="Borrower1SignatureArea">
              <p class="right">
                <SIGNATURE_ABOVE_LINE/>
                <SIGNATURE_LINE
                 _ID="Borrower1SignatureLine">
                ____________________________________
                </SIGNATURE_LINE>(Seal) </p>
              <p class="right">
                <SIGNATURE_BELOW_LINE>
<span class="dataEntered"
id="BORROWER-_FirstName">Richard</span>
<span class="dataEntered"
id="BORROWER-_MiddleName">R.</span>
<span class="dataEntered"
                  id="BORROWER-_LastName">Bradley</span>
                - Borrower</SIGNATURE_BELOW_LINE>
              </p>
              <p class="right">
                <SIGNATURE_BELOW_LINE>
<span class="dataEntered"
id="BORROWER-_Telephone">
123-456-7890</span>
                </SIGNATURE_BELOW_LINE>
              </p>
            </SIGNATURE_AREA>
            <SIGNATURE_AREA _ID="Borrower2SignatureArea">
<p class="right">
                <SIGNATURE_ABOVE_LINE/>
                <SIGNATURE_LINE
                  _ID="Borrower2SignatureLine">
             ___________________________________________
              </SIGNATURE_LINE>(Seal)
    </p>
              <p class="right">
                <SIGNATURE_BELOW_LINE/> - Borrower </p>
            </SIGNATURE_AREA>
            <SIGNATURE_AREA _ID="Borrower3SignatureArea">
              <p class="right">
                <SIGNATURE_ABOVE_LINE/>
                <SIGNATURE_LINE
                 _ID="Borrower3SignatureLine">
             ___________________________________________
                 </SIGNATURE_LINE>(Seal) </p>
              <p class="right">
                <SIGNATURE_BELOW_LINE/> - Borrower </p>
            </SIGNATURE_AREA>
            <SIGNATURE_AREA _ID="Borrower4SignatureArea">
              <p class="right">
                <SIGNATURE_ABOVE_LINE/>
                <SIGNATURE_LINE
```

```
                    _ID="Borrower1SignatureLine">
                                    ____________________________________
               </SIGNATURE_LINE>(Seal) </p>
                <p class="right">
                  <SIGNATURE_BELOW_LINE/> - Borrower </p>
              </SIGNATURE_AREA>
</SIGNATURE_SECTION>
```

Step Eight: Update document state in the HEADER

The `_StateType` attribute in the `<DOCUMENT_INFORMATION>` element must be updated to the value Signable.

Step Nine: Add <AUDIT_TRAIL> entry

An entry must be made in the `<AUDIT_TRAIL>` section indicating that the document was moved to a Signable state. For example:

<AUDIT_ENTRY _ActionType="**Signable**"
_PerformedByName="**Closing Company**"
_DateTime="**2003-08-11T15:06:59Z**"/>

Checklist ✓ Ensure that `<SIGNATURE_MODEL>` section is included in `<HEADER>` and contains a `<SIGNER>` element for each individual who will sign the document.

- ✓ Ensure that `<VIEW>` contains one `<SIGNATURE_TARGET>` that indicates where signatures will be in the document.
- ✓ Ensure that all appropriate `<SIGNATURE_SECTION>`s (one per signer role) are present and that they include one or more `<SIGNATURE_AREA>` elements corresponding to each `<SIGNER>` that must sign in that section.
- ✓ Ensure that the placeholders for the signatures are represented by `<SIGNATURE_LINE>` elements
- ✓ Set `_StateType = Signable` in the `<DOCUMENT_INFORMATION>` element
- ✓ Add a record to `<AUDIT_TRAIL>` with `_ActionType = Signable`

XML Used `<DOCUMENT_INFORMATION>` Structures `<AUDIT_TRAIL>`

`<<SIGNATURE_TARGETVIEW>` >

`<SIGNATURE_MODEL>`
`<SIGNATURE_ABOVE_LINE>`

```
<SIGNATURE_LINE>
<SIGNATURE_BELOW_LINE>
```

Known Issues	None at this time.
Other References	See Section 3 for information on mapping the data section with tagged views. See Section 5 for information on the types of allowable views and requirements for the view section. See Section 7 for specific information regarding requirements for the data section and customizing data and data DTDs. See Chapter 8.3, National eNote Registry and SMART Docs for detail on the language needed for the eNote. See Chapter 10: References for all other references.

Chapter 2.4: Signed SMART Docs

This chapter describes how to create a new SMART Doc® in the signed state.

Version

2.0

Revision History

Version	Date	Change
2.0	02/06/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification
UETA
ESIGN

Overview

This section describes how to create a signed SMART Doc. A signed SMART Doc contains the signatures of all signers (with the exception of the Recorder) and is tamper-evidence sealed, i.e., digitally signed. The SMART Doc Specification provides mechanisms to specify who will sign the document, where the signature is applied, and what portion of the document content (VIEW) is associated with the signature(s). In addition, the tamper-evident seal captures the exact document in a mathematical formula and thereby provides a way to detect, at any time in the future, if the content has been modified after the SMART Doc is converted to the Signed state. The tamper-evident seal cannot prevent the SMART Doc from being modified; however, a comparison of a new tamper-evident seal with the original one will determine if the file has been modified in any way.

The transition from the Unsigned state to the Signed state occurs in two phases:

Phase 1 - Relevant portions of the HEADER, VIEW, and AUDIT_TRAIL sections are updated as individual signatories apply their signatures.

Phase 2 - After all signatures have been applied, the SIGNATURES section is updated with a tamper-evident seal. (See Chapter 4.3, Tamper-Evident Seal Signatures for more detail on the timing and execution of the tamper-evident seal.)

A document will typically move from the Signable to the Signed state. However, it is possible that the document becomes Signed directly from one of the previous SMART Doc states, like Unpopulated or Populated, skipping the Signable state. It is not necessary to create every state independently. However, the audit trail must acknowledge each and every state transition as an entry as well as the attribution of who moved the document from state to state. See Chapter 9.1 for further information on the AUDIT_TRIAL and its requirements. Please note that the remainder of this document is written with the assumption that the SMART Doc in question is transitioning from the Signable state to the Signed state.

Pre Conditions

Document State: Signable. Document Categories: All

Post Conditions

Document State: Signed

Business Context

This chapter describes the process of applying signatures to a SMART Doc by converting a Signable document into a Signed document. A document in the Signable state should contain all of the information needed before the actual signing occurs - the only tasks left at this stage are to apply the individual signatures and tamper-evident seal to the document. The tamper-evident seal provides a way to detect if the signed document contents were modified after a SMART Doc has been converted to a Signed state.

Scenario

The borrowers are seated at the closing table in front of a computer screen that displays the eNote ready to be signed. The SMART Doc will be signed using a compliant signing platform that updates the document as individuals apply their signatures. As individual signatures are applied, the system updates the appropriate elements in the document and provides user feedback that the signature has been applied. Once all signatures have been applied, the signing platform will apply a tamper-evident seal.

Before a SMART Doc can be signed, it must contain all of the information necessary for applying signatures. For the sake of this chapter, we assume that the document is in the Signable state and contains the following information that is required before the signing process can be carried out.

Technical Guidance

Valid <SIGNATURE_MODEL> with a <SIGNER> section for each signatory. This section provides the following crucial information:

- Who is signing and in what order: Specified by the _RoleType and _SignatureOrderNumber attributes
- Where the signature will appear in the document: Specified by the AreaIDREF and the SectionIDREF attributes, which point to the <SIGNATURE_AREA> and <SIGNATURE_SECTION> in the VIEW. The actual signature is referenced by SignatureIDREF which references where the signature is located.
- What type of signature: _SignatureType which can be one of <SIGNATURE_TEXT>, <SIGNATURE_IMAGE>, or <SIGNATURE_OBJECT> within the View but has a value of Text, Image, Digital Signature or Object.
- What document content (VIEW) is covered by the signature: Specified by the TargetsIDREFS attribute. Please note that this could contain more than one target content area
- Valid <SIGNATURE_SECTION> for each role used to sign the document. The information here needs to match the corresponding <SIGNER> elements in the <SIGNATURE_MODEL> section

Valid <SIGNATURE_AREA> for each signer that matches its appropriate <SIGNER> element. This section may contain the following elements:

- <SIGNATURE_ABOVE>: Data that will appear above the signature
- <SIGNATURE_BELOW>: Data that appears below the signature

representation

…and MUST contain the following element:

- <SIGNATURE_LINE>: Placeholder that will be replaced by the actual signature representation when the user signs the document

Valid TargetsIDREFS attribute of the <SIGNER> element specifies which parts of the document are being covered by individual signatures. These references must exist in the VIEW. The target area for the signature will typically be the whole VIEW, but is possible that small sections of the VIEW may be covered by the signature using the <SIGNATURE_TARGET> element. In this case, these references must exist in the VIEW.

Once the above pre-conditions are satisfied, the document is ready for the signing process. We will use the following document sample of a Signable SMART Doc as it is transformed into a Signed document:

```
<HEADER _ID="FNMA_Sample_Header_3200">
      ...
      <SIGNATURE_MODEL>
            <SIGNER _RoleType="Borrower"
SignatureType="Text"
TargetsIDREFS="FNMA_Sample_View_3200"
SignatureIDREF="B1SigLine" _SignatureOrderNumber="1"
AreaIDREF="Borrower1SignatureArea"
SectionIDREF="BorrowerSignatures"/>
      </SIGNATURE_MODEL>
</HEADER>
…
<VIEW _ID="FNMA_Sample_View_3200"
_MIMETypeDescription="text/html"
_TaggedIndicator="True">
      …
      <SIGNATURE_SECTION _ID="BorrowerSignatures">
            <SIGNATURE_AREA
_ID="Borrower1SignatureArea">
                  <SIGNATURE_ABOVE_LINE/>
                        <SIGNATURE_LINE
_ID="B1SigLine"> </SIGNATURE_LINE>
      <SIGNATURE_BELOW_LINE>
                        <span id="BORROWER-
_FirstName">Richard</span>
                        <span id="BORROWER-
_MiddleName">R.</span>
                        <span id="BORROWER-
_LastName">Bradley</span>    -
Borrower</SIGNATURE_BELOW_LINE>
            </SIGNATURE_AREA>
      </SIGNATURE_SECTION>
</VIEW>
```

```
<SIGNATURES/>
```

Note that <SIGNATURES> is an empty section in this state.

Step One: Update <SIGNATURE_AREA> section for each signer
This section will typically be complete except for the signature placeholder (see Step Two) and there should not be any work at this stage. However, business practices may sometimes require parts of the above or below lines to be modified at signing time.

Step Two: Update the signature placeholder for each signer
The placeholder for the signature is specified by the <SIGNATURE_LINE> element. When a signature is applied, it is replaced by the appropriate signature representation in the VIEW, <SIGNATURE_TEXT> in our example:

```
<SIGNATURE_TEXT_ID="B1SigLine">Electronically signed
by Richard R. Bradley on 2/1/2002 16:12:51
PST</SIGNATURE_TEXT>
```

Step Three: Update the Audit_Trail entry for each signer
Every time a signature is applied, a new <AUDIT_ENTRY> is placed in the <AUDIT_TRAIL> section with the _ActionType attribute set to Signed:

```
<AUDIT_ENTRY _DateTime="2002-07-31T18:07:32Z"
_PerformedByName="Borrower" _ActionType="Signed"/>
```

See chapter 9 for further information on the AUDIT_TRAIL.

Step Four: Change _StateType to Signed in the HEADER
Once all of the signatories have applied their signatures, the document state is changed to Signed in the <DOCUMENT_INFORMATION> element:

```
<DOCUMENT_INFORMATION _Type="Note"
_StateType="Signed"
NegotiableInstrumentIndicator="True"
MustBeRecordedIndicator="False"
_FormNumberIdentifier="3200"
SMARTDocumentCategoryType="1"/>
```

Step Five: Apply tamper-evident seal
Once the document contains all the signatures, its contents must not be modified or the signatures would become invalid. To guard against this, the document is tamper-evident sealed and the <SIGNATURES> section is updated with a digital signature of the target sections of the VIEW as specified in the <SIGNER> elements. A new <SIGNER> element with _RoleType of TamperSealer is added and contains all the sections of the document that are going to be included in computing the digital signature:

```
<SIGNER SignatureIDREF="TamperSealer01"
SignatureType="DigitalSignature"
TargetsIDREFS="FNMA_Sample_Header_3200
```

```
FNMA_Sample_Data_3200 FNMA_Sample_View_3200
SignedContent01 TamperSealer01"
_RoleType="TamperSealer" _SignatureOrderNumber="1"/>
```

The <SIGNATURES> section is then updated with the digital signature using the XML Signature Standard:

```
<SIGNATURES>
      <Signature Id="TamperSealer01">
            <SignedInfo>
                  …
                 <Reference
URI="FNMA_Sample_View_3200">
                  …
                  </Reference>
      …
      </Signature>
</SIGNATURES>
```

In the example above, the XML Signature contains a Reference element for each ID specified in the TargetsIDREFS and proves that the digital signature was applied to the document content viewed by the signers at the time of signing. Tampering with this content post-signing would break the tamperevident seal, allowing anyone to later validate the seal and see that the document had been modified. See chapter 4.3 for further information on tampersealing SMART Docs.

Step Six: Add the tamper-evident seal Audit Trail entry
Once the tamper-seal has been applied, the author should create an entry of the form:

```
            <AUDIT_ENTRY _ActionType="Signed"
_DateTime="2003-02-20T19:56:36Z"
_PerformedByName="eMD Signing Tool”
```

Please note that there is no specific _ActionType for the tamper-sealing action, and the Signed action is used for this purpose also.

Step Seven: Validate the Document There
are two validation steps:

XML Validation against the relevant DTDs
External validation of the tamper-evident seal

Checklist

- Make sure that the document is in the Signable state, i.e. contains the necessary information required to apply signatures
- For each signature applied, update the `<SIGNATURE_AREA>` element in the `<SIGNATURE_SECTION>`, if changes in the signatories have occurred between the time that the Signable state was created
- For each signature applied, replace the signature placeholder, SIGNATURE_LINE with the appropriate signature representation (SIGNATURE_TEXT, SIGNATURE_IMAGE, etc.)
- For each signature applied, update the `AUDIT_TRAIL` entry
- Once all the signatures have been applied, set the <HEADER> element `_StateType to Signed`
- Once all the signatures have been applied, tamper-evident seal the document content covered by the TargetsIDREFS attribute of each SIGNER element, and update the `<SIGNATURES>` section with the tamper-evident seal
- Apply the tamper-evident seal `<AUDIT_TRAIL>` entry

  Validate document: XML and external tamper-seal validation

XML Structures Used

```
<HEADER>
<SIGNATURE_MODEL>, <SIGNER>
<SIGNATURE_TARGET>
<SIGNATURE_SECTION>
<SIGNATURE_AREA>
<SIGNATURE_LINE>, <SIGNATURE_TEXT>,
<SIGNATURE_IMAGE>, <SIGNATURE_OBJECT>
<SIGNATURES>
<AUDIT_TRAIL>
```

Whether a specific XML element is used or not may depend on the document category.

Known Issues

The SMART Doc specification uses a modified version of the XHTML DTDs to allow for the signature elements to appear in the view.

Other References

See Chapter 10: References for references to other documents.

Chapter 3.1: Data Mapping

Data Mapping: This section covers how to link data to the XHTML View.

Version

2.0

Revision History

Version	Date	Change
2.0	02/06/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

MISMO® SMART Doc® Specification, Version 1.02

Overview

A SMART Doc is defined for our purposes as a single electronic document that binds together data and presentation along with other information needed to maximize its performance. The data section may be used by down stream processing systems to efficiently and quickly extract information from the SMART Doc. The view captures what was presented to the signer on the display device. The purpose of mapping data to the view is to provide a mechanism to validate that the contents of the view match the data used by automated systems when working with the document. This chapter describes how to map the data section in a SMART Doc to a Category 1, XHTML View. For conversions of information (different representations of information in the data and view sections) see Chapter 3.2, Data Conversions.

Pre Conditions

The state of the SMART Doc:
Document State: Any
Document Categories: 1

Post Conditions

The state of the SMART Doc:
Document State: Any
Document Categories: 1

Business Context

With the information and its presentation, and the relationship between the two, bound in a single immutable file, the integrity of the electronic data can be guaranteed. That is, this specification allows system validation to ensure that what the borrower sees and signs on the display device is the exact document that will be stored as a legal instrument. It also ensures that the data displayed on the screen will be the exact data used for downstream processing of the loan. We have designed the SMART

Document specification to unlock the greatest value from eMortgages: based on the SMART Doc specification, the document that the borrower sees on the screen can flow through a series of systems in a fully-automated fashion while ensuring data integrity and full quality control over the life of the loan.

The integrity of the document is maintained by linking the information in the data section to the same information presented in the view. Linking the information allows for the data to be represented in a form most useful for computer processing and for the view to meet the requirements needed for the presentation of the information. Validation that the data is consistent in the data section and the view section through linking provides an extra level of data integrity.

A tamper evident digital signature is applied after all signatures have been applied. This signature insures that the data, the view and the links between them have not been altered. The tamper evident wrap only provides evidence that the SMART Doc has been tampered with, it does not provide information on what was changed. Validation of the data against the view and the linkages may provide this information.

Scenario

The originating and receiving parties have a need to validate consistency in the data and view sections of each SMART Doc transferred. The originating system creates a SMART Doc with the appropriate linkages between the data and view. The receiving system validates that the linkages are valid. Note: This validation process is a separate process from XML validation to a DTD.

Technical Guidance

Mapping

The data section in a MISMO 1.0 SMART Doc classified as Category 1 MUST contain a mapping between fields in the XML data section <DATA> and the visual depiction of that data in the tagged XHTML view <VIEW>. The linkage is maintained in the <MAP> section, which is part of the <DATA> section. The map is maintained with a series of ARC elements. The <ARC> elements link data values from the XML <DATA> section to a variable data field in the <VIEW> section. Appropriate formatting conversions may be applied if necessary. Conversions are discussed in Chapter 3.2, Data Conversions.

An <ARC> must be created for each variable data field in the View. The ARC element maps the data using XPath. XPath is the W3C's general language specification for addressing parts of an XML document. The XPath expression is linked to a unique identifier in the view by using the <span> or <div> tags and the <span> or <div> tag's `id` attribute.

The <ARC> element has two attributes that maintain the link. The `DataLinkDescription` attribute is a valid XPath expression and is used to refer to a single, existing element in the <DATA> section. The `ViewLinkDescription` attribute refers to a unique XML ID that is found in a <span> or <div> element of the variable data element to be linked.

The identifier in the view must conform to an XML ID name:
a) must be a valid IDRef; and,
b) must not include characters that would prevent it from being used in an Xpath expression.

For readability, the value of the `id` attribute should resemble the XML data tag that it is associated with. The names of the unique identifiers used within the data and view sections have the following recommended naming convention, although any naming convention may be used:

1) The id name matches the XPath expression name;

2) The id name is the combination of ELEMENT_NAME followed by "-" followed by the attribute name;

3) If there are multiple data elements of the same name, the index number is included in the id.

Examples are provided below of this naming convention

Step 1: Create the XPath expression for the DATA element and the ID for the VIEW

The following XML fragment:

```
 <DATA _ID="FNMA_Sample_Data_3200">
    <MAIN>
      <LOAN MISMOVersionIdentifier="2.3">
        <_APPLICATION>
 […]
          <MORTGAGE_TERMS NoteRatePercent="8.625/>
          <BORROWER BorrowerID="B111111111"
_FirstName="Richard" _MiddleName="R."
_LastName="Bradley"/>
           <BORROWER BorrowerID="B222222222"
  _FirstName="Rachael" _LastName="Bradley"/>
</_APPLICATION>
[…]
     </LOAN>
    </MAIN>
    </DATA>
```

would have the following XPath expressions and IDs:

Xpath:
LOAN/_APPLICATION/MORTGAGE_TERMS/@NoteRatePercent
id:
MORTGAGE_TERMS-NoteRatePercent
Xpath: BORROWER[1]/@_FirstName
id: BORROWER[1]-_FirstName
Xpath: BORROWER[1]/@_LastName
id: BORROWER[1]-_LastName
Xpath: BORROWER[2]/@_FirstName
id: BORROWER[2]-_FirstName
Xpath: BORROWER[2]/@_LastName
id: BORROWER[2]-_LastName

The XPath expression:

```
    //MORTGAGE_TERMS/@NoteRatePercent
```

references the attribute `NoteRatePercent` of the `MORTGAGE_TERMS` element by using the "@" to denote an attribute and "/" to denote the hierarchy in the XML. See the XPath specification (under other references below) for valid expressions.

Step 2: Add the ARC

The following sample shows an XML document fragment that links the Note Rate Percent data to the Note Rate Percent in an XHTML view of the Note. It is important to stress that this sample does NOT contain all required elements or attributes for a SMART Doc. *Every data item referenced in the VIEW MUST have a corresponding element in the DATA section and must have an <ARC> element explicitly defining the mapping between the two.*

In the ARC element, the data is referenced in the XPath in the `DataLinkDescription` attribute. The `ViewLinkDescription` attribute makes use of the id() function in XPath. The id() function matches an element that has an attribute of type ID that contains the value specified. The first <ARC> element's `ViewLinkDescription` attribute specifies an element that has an attribute of type `ID` that has a value of `MORTGAGE_TERMS-NoteRatePercent`.

```
<MAP TargetIDREF="FNMA_Sample_View_3200">
 <ARC DataLinkDescription =
 "//LOAN/_APPLICATION/MORTGAGE_TERMS/@NoteRatePercent"
 ViewLinkDescription = "id(MORTGAGE_TERMS-NoteRatePercent)"/>
</MAP>
```

The XPath expression into the Data section will normally include only the XPath for the Data DTD. The View link XPath description will use an XPath ID function that will match the element with a matching ID.

It should be noted that the optional <CONVERT> element may be used to control the presentation of the data in the <VIEW> by specifying a mask to convert the raw data value into its visual representation. This is discussed in the next chapter, 3.2.

ARCs should NOT be created for data items that do not appear in the View.

Step 3: Add the ID to the View

Add the unique identifier for the data point to the <span> or <div> tags in the VIEW:

```
  <SMART_DOCUMENT>
     <HEADER> ... </HEADER>
  <DATA _ID="FNMA_Sample_Data_3200">
    <MAIN>
      <LOAN MISMOVersionIdentifier="2.3">
        <_APPLICATION>
 […]
          <MORTGAGE_TERMS NoteRatePercent="8.625/>
[…]
       </_APPLICATION>
       </MAIN>
    <MAP TargetIDREF="FNMA_Sample_View_3200">
             <ARC DataLinkDescription =
"//LOAN/_APPLICATION/MORTGAGE_TERMS/@NoteRatePercent"
 ViewLinkDescription =
"id(MORTGAGE_TERMS-NoteRatePercent)"/>
      […]
    </MAP>
</DATA>
    <VIEW MimeType="text/html" tagged="true">
      <html>
   […]
        < span class="dataEntered"
  id="MORTGAGE_TERMS-NoteRatePercent">8.625%</span>
  […]
    </html>
    </VIEW>
  </SMART_DOCUMENT>
```

Step 4: Validate the Mapping

The SMART Doc DTD does not provide a mechanism to validate the map between the DATA and VIEW sections. Additional software will need to be written to validate that the links are correct and that the values are correct.

Checklist

- 9 Create valid XPath expressions for all data points referenced by the VIEW.
- 9 Create unique identifiers for each data point used within the VIEW.
- 9 Add a MAP section, with ARC elements for every data point found within the view.
- 9 Add <span> or <div> tags that reference the unique identifiers within the VIEW.
- 9 Validate the mapping.

XML Structures Used

This section provides a listing of the XML and XHTML structures used, in the following format:
<MAP>
<ARC>
<span>
<div>

Known Issues

None

Other References

See Chapter 2.2: Populating a SMART Doc; Chapter 3.2: Data Conversions; Chapter 3.3:Using Multiple Conversions; and Chapter 3.4: Operators; for other considerations when mapping the data and view sections.

See Section 5 for information on the types of allowable views and requirements in the view section.

See Section 7 for specific information regarding requirements for the data section and customizing data and data DTDs.

See Chapter 5.4: eNote Language in the View for detail on the language needed for the eNote.

See Chapter 10: References for references to other documents.

Chapter 3.2: Data Conversions

This section describes how to convert a single data field in the data section to a single data field in the XHTML View with different formatting.

Version

2.0

Revision History

Version	Date	Change
2.0	02/06/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

MISMO® SMART Doc® Specification, Version 1.02

Overview

In certain situations, the representation of a single data field in the data section may differ from what is presented in the view. This chapter describes how to convert the data to a different representation in the view section. The conversions are only applicable to SMART Docs that are Category One. This chapter uses the format conversions that are present in the uniform instrument number 3200, the multi-state Note.

Pre Conditions

The state of the SMART Doc:
Document State: Any
Document Categories: 1

Post Conditions

The state of the SMART Doc:
Document State: Any
Document Categories: 1

Scenario

A document preparation company is preparing a Multistate, Uniform Instrument 3200 eNote SMART Doc. The 3200 eNote has several data fields that are represented differently in the View section than in the Data section. This chapter describes how to express the conversions.

Technical Guidance

This chapter assumes that the proper <ARC> elements have already been defined as described in Chapter 3.1. Conversion allows for data in the view section to be displayed in a variety of different formats, and conversion allows for masks to show the differences between the data section and the view.

This chapter covers a single conversion for a single data point with a single view representation. <ARC> elements can only contain zero or one <CONVERSION> elements. It is not possible to specify more that one conversion in a single ARC element. As specified in the DTD, it is NOT possible to do the following:

```
<ARC>
<CONVERSION/>
<CONVERSION/>
<ARC/>
```

If there are multiple acceptable conversions for a single data field, you must use the <OPERATOR> element. Operators group the conversions. Using operator with conversions is covered in Chapter, 3.4: Operator Types. However it is possible to include multiple ARCs in the <CONVERSION> element, for other situations. This is described in Chapter 3.3: Using Multiple Conversions.

The 3200 uniform instrument has the following Data fields and representative views:

DATA FIELD	DATA	VIEW
LOAN/_APPLICATION/MORTGAGE_TERMS/ @LenderLoanIdentifier		
LOAN/_CLOSING_DOCUMENTS/EXECUTION/ @_Date	***2001-08-14***	***August 14, 2001***
LOAN/_CLOSING_DOCUMENTS/EXECUTION/ @_City	Louisburg	Louisburg
LOAN/_CLOSING_DOCUMENTS/EXECUTION/ @_State	NC	NC
OAN/_APPLICATION/PROPERTY/@_StreetAddress	748 N. Main Street	748 N. Main Street
LOAN/_APPLICATION/PROPERTY/@_City,	Louisburg	Louisburg
LOAN/_APPLICATION/PROPERTY/@_State,	NC	NC
LOAN/_APPLICATION/PROPERTY/@_PostalCode,	27549	27549
LOAN/_APPLICATION/MORTGAGE_TERMS/ @OriginalLoanAmount,	***96500.00***	***$96,500.00***
LOAN/_CLOSING_DOCUMENTS/LENDER/ @_UnparsedName,	Columbia National Incorporated	Columbia National Incorporated
LOAN/_APPLICATION/MORTGAGE_TERMS/ @NoteRatePercent,	***8.625***	***8.625%***
LOAN/_APPLICATION/LOAN_PRODUCT_DATA/ MORTGAGETERMS/@PaymentRemittanceDay	***1***	***1st***
LOAN/_APPLICATION/LOAN_PRODUCT_DATA/ LOAN_FEATURES/@ScheduledFirstPaymentDate,	***2001-10-01***	***October 01, 2001***
LOAN/_APPLICATION/LOAN_PRODUCT_DATA/ LOAN_FEATURES/@LoanMaturityDate,	***2031-09-01***	***September 1, 2031***
LOAN/_APPLICATION/LOAN_PRODUCT_DATA/ LOAN_FEATURES/NOTE_PAY_TO/ @_StreetAddress,	P.O. Box 3050	P.O. Box 3050
LOAN/_APPLICATION/LOAN_PRODUCT_DATA /LOAN_FEATURES/NOTE_PAY_TO/@_City,	Columbia	Columbia
LOAN/_APPLICATION/LOAN_PRODUCT_DATA/ LOAN_FEATURES/NOTE_PAY_TO/@_State,	MD	MD
LOAN/_APPLICATION/LOAN_PRODUCT_DATA/ LOAN_FEATURES/NOTE_PAY_TO/@_PostalCode,	21045-6050	21045-6050
LOAN/_APPLICATION/LOAN_PRODUCT_DATA/ LOAN_FEATURES/ @OriginalPrincipalAndInterestPaymentAmount,	***763.02***	***$763.02***
LOAN/_APPLICATION/LOAN_PRODUCT_DATA/ LOAN_FEATURES/LATE_CHARGE/ @_GracePeriod,	***15***	***fifteen***
LOAN/_APPLICATION/LOAN_PRODUCT_DATA/ LOAN_FEATURES/LATE_CHARGE/@_Rate,	***4.000***	***4.000%***

LOAN/_APPLICATION/BORROWER/@_FirstName,	Richard	Richard
LOAN/_APPLICATION/ ORROWER/@_MiddleName,	R.	R.
LOAN/_APPLICATION/BORROWER/@_LastName	Bradley	Bradley
LOAN/_APPLICATION/MERS/MERS_MINNumber	1234512345123 45123	Not displayed in the view

There are five types of conversions in the table above:

Dates: from MMDDYYYY to Month DD, YYY
Amounts: 9999.99 to $9,999.99
Percentages: 9.999 to 9.999%
Numbers to words: 15 to fifteen
Numbers to ordinals: 1 to 1st

The steps involved in creating the conversion types are defined below

Situation 1: Date Conversions

The following is an example that requires a conversion: the formatting of a date in a different presentation than what is in the data section.

The <DATA> section contains:

```
<LOAN_FEATURES ScheduledFirstPaymentDate="2001-10-01" />
```

The View section contains the following for the first scheduled payment:

```
<span class="dataEntered" id="LOAN_FEATURES-
ScheduledFirstPaymentDate">
October 01, 2001</span>.
```

Note the use of a clear and consistent naming convention used for the view ID "LOAN_FEATURES-ScheduledFirstPaymentDate". The naming convention used in the view ID should provide an indication of the matching data element.

In order to convert the data, an ARC must first exist that links the data with the view:

```
<ARC DataLinkDescription =
 "//LOAN/_APPLICATION/LOAN_PRODUCT_DATA/LOAN_FEATURES/
               @ScheduledFirstPaymentDate"
             ViewLinkDescription=
             "id(LOAN_FEATURES-ScheduledFirstPaymentDate)"/>
```

In order to perform the conversion, we create a <CONVERSION> element that is a child of the ARC element:

```
<CONVERSION _Type=" "_MaskDescription=" " />
```

The <CONVERSION> element has two attributes: the conversion type, _Type and the mask for the conversion, _FillMask. This example shows a _FillMask conversion. Specifically, the _FillMask conversion is used to both create the presentation of the data used within the View as well as verify the data in the View matches the data in the Presentation. The _MaskDescription attribute contains the format (as defined in the specification) for the presentation. There are several date formats possible but we need to convert the date to a month in words, followed by the day and a four digit year. The mask needed is "MMMM dd, yyyy". Consult the SMART Doc specification for other types of date conversions. The resulting conversion is:

```
<CONVERSION _MaskDescription="MMMM dd, yyyy"
_Type="FillMask"/>
```

And this is linked to the appropriate ARC elements that have date conversions:

```
<ARC
DataLinkDescription="//LOAN/_CLOSING_DOCUMENTS/EXECUTION
/@_Date" ViewLinkDescription="id(EXECUTION-_Date)">
  <CONVERSION _MaskDescription="MMMM dd, yyyy"
_Type="FillMask"/>
</ARC>

<ARC
DataLinkDescription="//LOAN/_APPLICATION/LOAN_PRODUCT_DA
TA/LOAN_FEATURES/@LoanMaturityDate"
ViewLinkDescription="id(LOAN_FEATURES-
LoanMaturityDate)">
  <CONVERSION _MaskDescription="MMMM dd, yyyy"
_Type="FillMask"/>
</ARC>

 <ARC
DataLinkDescription="//LOAN/_APPLICATION/LOAN_PRODUCT_DA
TA/LOAN_FEATURES/@ScheduledFirstPaymentDate"
ViewLinkDescription="id(LOAN_FEATURES-
ScheduledFirstPaymentDate)">
  <CONVERSION _MaskDescription="MMMM dd, yyyy"
_Type="FillMask"/>
</ARC>
```

The `_FillMask` conversion is just one of many conversion types that can be used. The other type of conversion is the ConvertType as shown in the example below:

```
<ARC
DataLinkDescription=/LOAN/MORTGAGE_TERMS/@LoanAmount
ViewLinkDescription="id(MORTGAGE_TERMS-LoanAmount)">
      <CONVERSION _Type="ConvertType"
DataLinkDescription=http://www.w3.org/2001/XMLSchema#dec
imal
ViewLinkDescription="http://www.w3.org/2001/XMLSchema#st
ring" />
</ARC>
```

This `ConvertType` uses XML Schema Data Types in the link descriptions to convert a decimal like (125000.00) to the text "One Hundred Twenty Five Thousand". Consult the specification and the W3C schema datatypes recommendation (in other references below) for other types of schema data type conversions.

Situation 2: Converting amounts with a dollar sign

The following is an example that requires a conversion: the addition of the dollar sign in the view section with a comma-separated amount.

The <DATA> section contains:

```
<MORTGAGE_TERMS  OriginalLoanAmount="96500.00" />
```

The View section contains the following for the loan amount:

```
<span class = "dataEntered" id = "MORTGAGE_TERMS-
OriginalLoanAmount">$96,500.00</span>
```

In order to add a dollar sign to the view, an ARC must first exist that links the data with the view:

```
<ARC DataLinkDescription=
"//LOAN/_APPLICATION/MORTGAGE_TERMS/@OriginalLoanAmount"
ViewLinkDescription=
"id(MORTGAGE_TERMS-OriginalLoanAmount)"/>
```

In order to perform the conversion, we create a <CONVERSION> element that is a child of the ARC element:

```
<CONVERSION _MaskDescription="$,##0.00"
_Type="FillMask"/>
```

For a description of the `<CONVERSION>` element's two attributes _Type and `_FillMask,` see Step 1. This example shows a `_MaskDescription` attribute that contains the format (as defined in the specification) for the presentation. Specifically, this mask adds a dollar sign "$" to the amount and specifies that the amount contains commas and two decimal points. Consult the specification for other types of formats for amounts. The resulting conversion is linked to the appropriate ARC element:

```
<ARC DataLinkDescription=
"//LOAN/_APPLICATION/MORTGAGE_TERMS/@OriginalLoanAmount"
ViewLinkDescription=
"id(MORTGAGE_TERMS-OriginalLoanAmount)">
<CONVERSION _MaskDescription="$,##0.00"
_Type="FillMask"/>
</ARC>
```

Situation 3: Converting with a percent sign

The following is an example that requires a conversion: the addition of a percent sign in the view section.

The <DATA> section contains:

```
<MORTGAGE_TERMS NoteRatePercent="8.625" />
```

The View section contains the following for the note interest rate:

```
<span class = "dataEntered" id = "MORTGAGE_TERMS-
NoteRatePercent">8.625%</span>
```

In order to convert by adding a percent sign in the view, an ARC must first exist that links the data with the view:

```
<ARC DataLinkDescription
="//LOAN/_APPLICATION/MORTGAGE_TERMS/@NoteRatePercent"
ViewLinkDescription =
"id(MORTGAGE_TERMS-NoteRatePercent)"/>
```

In order to perform the conversion, we create a <CONVERSION> element that is a child of the ARC element:

```
<CONVERSION _MaskDescription="#0.000%"
_Type="FillMask"/>
```

For a description of the `<CONVERSION>` element's two attributes _Type and `_FillMask,` see Step 1. This example shows a `_MaskDescription` attribute that contains the format (as defined in the specification) for the presentation. Specifically, this mask adds a percent sign "%" to the amount and
specifies that the percent contains three decimal points. Consult the specification for other types of formats for percentages. The resulting conversion is linked to the appropriate ARC element:

```
<ARC
DataLinkDescription="//LOAN/_APPLICATION/MORTGAGE_TERMS/
@NoteRatePercent"
ViewLinkDescription="id(MORTGAGE_TERMS-
NoteRatePercent)">
          <CONVERSION _MaskDescription="#0.000%"
_Type="FillMask"/>
</ARC>
```

Situation 4: Representing numbers with words

The following is an example that requires a conversion: presentation of a numeric string as words.

The <DATA> section contains:

```
<LATE_CHARGE _GracePeriod="15" />
```

The View section contains the following for the late payment grace period:

```
<span class="dataEntered" id="LATE_CHARGE-
_GracePeriod">fifteen</span>
```

In order to convert from a numeral to words in the view, an ARC must first exist that links the data with the view:

```
<ARC
```

```
DataLinkDescription="//LOAN/_APPLICATION/LOAN_PRODUCT_DA
TA/LOAN_FEATURES/LATE_CHARGE/@_GracePeriod"
ViewLinkDescription="id(LATE_CHARGE-_GracePeriod)"/>
```

In order to perform the conversion, we create a <CONVERSION> element that is a child of the ARC element:

```
<CONVERSION _MaskDescription="words()"
_Type="FillMask"/>
```

For a description of the `<CONVERSION>` element's two attributes _Type and `_FillMask`, see Step 1. This example shows a `_MaskDescription` attribute that contains the format (as defined in the specification) for the presentation. Specifically, this mask includes the words() function indicating that the numeric amount is to be converted to words. Consult the specification for other types of formats for numbers. The resulting conversion is linked to the appropriate ARC element:

```
<ARC DataLinkDescription=
//LOAN/_APPLICATION/LOAN_PRODUCT_DATA/LOAN_FEATURES/LATE
_CHARGE/@_GracePeriod" ViewLinkDescription=
id(LATE_CHARGE-_GracePeriod)">
  <CONVERSION _MaskDescription="words()"
_Type="FillMask"/>
</ARC>
```

Situation 5: Converting numbers to ordinals

The following is an example that requires a conversion: presentation of a numeric string as an ordinal.

The <DATA> section contains:

```
<MORTGAGE_TERMS PaymentRemittanceDay="1" />
```

The View section contains the following for the late payment grace period:

```
<span class = "dataEntered" id = "MORTGAGE_TERMS-
PaymentRemittanceDay">1st</span>
```

In order to represent the numeric value as an ordinal in the view, an ARC must first exist that links the data with the view:

```
<ARC
DataLinkDescription="//LOAN/_APPLICATION/MORTGAGE_TERMS/
@PaymentRemittanceDay"
ViewLinkDescription="id(MORTGAGE_TERMS-
PaymentRemittanceDay)"/>
```

In order to perform the conversion, we create a <CONVERSION> element that is a child of the ARC element:

```
<CONVERSION _MaskDescription="ord" _Type="FillMask"/>
```

For a description of the `<CONVERSION>` element's two attributes _Type and `_FillMask`, see Step 1. This example shows a `_MaskDescription` attribute that contains the format (as defined in the specification) for the presentation. Specifically, this mask includes the ordinal function indicating that the numeric amount is to an ordinal. Consult the specification for other types of formats for numbers. The resulting conversion is linked to the appropriate ARC element:

```
<ARC
DataLinkDescription="//LOAN/_APPLICATION/MORTGAGE_TERMS/
@PaymentRemittanceDay"
ViewLinkDescription="id(MORTGAGE_TERMS-
PaymentRemittanceDay)">
          <CONVERSION _MaskDescription="ord"
_Type="FillMask"/>
</ARC>
```

Checklist

- Review all the required data fields in the view and check those that have a presentation different form the data section
- Create `<CONVERSION>` elements for those items in the View section that require formatting
- Consult the SMART Doc specification for the required conversion type and mask
- Validate the conversions.

XML Structures Used

This section provides a listing of the XML structures used, in the following format:

`<ARC>`
`<CONVERSION>`

Known Issues

There are no requirements on the format of the data in the data section. For instance, your application may choose to represent the scheduled pay-off date in the data section as:

```
<LOAN_FEATURES ScheduledFirstPaymentDate="2001-10-01"/>
```

And another implementaiton may choose the following:

```
<LOAN_FEATURES ScheduledFirstPaymentDate="10/01/2001"/>
```

XML DTD validation does not check for consistency within the <ARC> elements, nor will validation check that the conversions have been formatted correctly in the view. You must write additional software to check that the <ARC> elements correctly:

Specify a valid XPath in the data section
Include an identifier in the view
Contain a data element for every data field referenced in the view
Have converted the data in the view

Other References

See Chapter 10: References for references to other documents.

See the other chapters in this section, Section 3, for other considerations when handling conversions.

See section 5 for information on the types of allowable views and requirements in the view section.

See Section 7 for specific information regarding requirements for the data section.

Chapter 3.3: Using Multiple ARCs and Conversions

Using Multiple Conversions: This section covers one to many and many to one linking between the data section and the view section.

Version

2.0

Revision History

Version	Date	Change
2.0	02/07/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

MISMO® SMART Doc® Specification, Version 1.02

Overview

In certain situations, the representation of the data in the data section may differ from what is presented in the view. This chapter describes:

How to convert the data from multiple data items to a single representation in the view section. For instance, the data section can display discrete data points that are combined in the view. In the example we use in this chapter, the data that describes how long a borrower has lived at the present address – 2 years and 6 months – is mapped from the "2" and the "6" in the data section and combined using a "fill mask" to a view that depicts "Two years and six months."

How to map single data fields from the data section to multiple places in the view. For example, the lender's name may appear multiple times in the view but only exists once in the data section.

How to choose from an enumerated list and convert the enumerated list value to an alternative representation in the view. Enumerated lists are in the MISMO standard naming conventions, which concatenates word without spaces. This conversion converts the data section list item to readable text in the view .

The conversions are only applicable to SMART Docs that are of Category One.

Pre Conditions

The state of the SMART Doc:
Document State: Any
Document Categories: 1

Post Conditions

The state of the SMART Doc:
Document State: Any
Document Categories: 1

Business Context

The originating and receiving parties have a need to validate consistency in the data and view sections of each SMART Doc transferred. In some circumstances the display of the data in the view section differs from the representation in the view. This chapter describes how to implement data conversions that are one to many and many to one in the data and view.

Technical Guidance

Conversions that involve a simple one-to-one relationship are described in Chapter 3.2. For information on implementing operators and having multiple possible values for a single view field, see the next Chapter 3.4: Operators.

This guidance describes the use of operators in three different situations. Situation One describes many to one mappings. Situation 2 implements a one to many mapping. Situation 3 describes the conversion of enumerated list values. There may be other situations described in future versions of this implementation guide.

Situation 1: Mapping several input fields to a single output presentation (many to one)

It is possible to include multiple `<ARC>` elements in the `<CONVERSION>` element, for other situations such as multiple possible values which is discussed it the next chapter, 3.4 Operator Types: how to use operators in conversions. Some conversions may map several input data fields to a single output presentation field. These conversions are handled by allowing the `<CONVERSION>` element to contain a set of `<ARC>` elements involved in the conversion.

Here is an example where we map two fields in the XML Data section to a single formatted string. The method for formatting is the "`_FillMask`" type of conversion. The `<CONVERSION>` element contains the `<ARC>` elements used to populate the mask. Each individual `<ARC>` contains a separate `<CONVERSION>` element that shows what part of the mask the specific field populates.

The data sections contains:

```
<BORROWER>
   <_RESIDENCE BorrowerResidencyDurationYears="2"
               BorrowerResidencyDurationMonths="6"/>

</BORROWER>
```

And the view has the following presentation:

```
<span class = "dataEntered" id = "_RESIDENCE-Duration">two
years and six months </span>
```

We begin the conversion with a `<CONVERSION>` element. It contains a set of `<ARC>` elements to create the presentation string "two years and six months" with each `<ARC>` element containing a `<CONVERSION>` element that denotes the part of the mask that the top level `<CONVERSION>` element uses. The {} indicates the part of the mask that each <ARC> element refers to.

```
<CONVERSION _Type= "FillMask" _MaskDescription ="words()
‘years and’ words() ‘months’">
  <ARC DataLinkDescription = "//_RESIDENCE/
         @BorrowerResidencyDurationYears"
       ViewLinkDescription =
        "id(BorrowerResidencyDurationYears)">
      <CONVERSION _Type= "FillMask" _MaskDescription
="words() ‘years and’ {words()} ‘months’"/>
</ARC>
  <ARC DataLinkDescription = "//_RESIDENCE/
         @BorrowerResidencyDurationMonths"
       ViewLinkDescription =
        "id(BorrowerResidencyDurationMonths)">
      <CONVERSION _Type= "FillMask" _MaskDescription
="{words()} ‘years and’ words() ‘months’"/>
</ARC>
</CONVERSION>
```

Situation 2: Map the same data field to multiple view fields (one to many)

This example maps a single Lender name to two Lender Names referenced in the View. The first step is to create two <ARC> elements, one for each reference in the view. In the `ViewLinkDescription` attribute you must specify unique names for each view reference.

```
<ARC DataLinkDescription="/LENDER/@_Name"
ViewLinkDescription="id(LENDER-_Name01)" />

<ARC DataLinkDescription="/LENDER/@_Name"
ViewLinkDescription="id(LENDER-_Name02)" />
```

Note that this example does not contain any conversions. It is possible to have different conversions associated with each <ARC> element.

Situation 3: Selection and conversions of enumerated list values

Many data points have enumerated lists. Often you may wish to map the list items to strings that are different from the list item's value. As an example consider the `MortgageType` attribute of the `<MORTGAGE_TERMS>` element:

```
<MORTGAGE_TERMS  MortgageType=" FarmersHomeAdministration" />
```

The `MortgageType` attribute is an enumerated list of values:
Conventional
FarmersHomeAdministration
FHA
HELOC
VA and

other

The string “FarmersHomeAdministration” would not be acceptable in the view, so an `<ARC>` must be constructed when the `MortgageType` attribute value is “FarmersHomeAdministration” to convert it to readable version of “Farmers Home Administration”. XPath allows for selection of an attribute with a specific value. Any string may be used to construct the view conversion. For example:

```
<ARC
DataLinkDescription="/LOAN/MORTGAGE_TERMS/[@MortgageType=’Fa
rmersHomeAdministration']"
 ViewLinkDescription="id(MORTGAGE_TERMS-
FarmersHomeAdministration ">
 <CONVERSION _Type="FillMask" _MaskDescription="Farmers Home
Administration”/>
</ARC>
```

NOTE: The use of literal strings in `_MaskDescription` is restricted to the situation described here.

Checklist

- 9 Review all the required data fields in the view and check those that have a presentation different form the data section
- 9 Create <CONVERSION> elements for those items in the View section that require formatting
- 9 Consult the SMART Doc specification for the required conversion type and mask
- 9 Validate the conversions.

XML Structures Used

This section provides a listing of the XML structures used, in the following format:

<ARC>
<CONVERSION>

Known Issues

There are no requirements on the format of the data in the data section. For instance, your application may choose to represent the scheduled pay-off date in the data section as :

```
<LOAN_FEATURES ScheduledFirstPaymentDate="2001-10-01"/>
```

And another implementation may choose the following:
```
<LOAN_FEATURES ScheduledFirstPaymentDate="10/01/2001"/>
```

XML DTD validation does not check for consistency within the <ARC> elements, nor will validation check that the conversions have been formatted correctly in the view. You must write additional software to check that the <ARC> elements correctly:
Specify a valid XPath in the data section
Include an identifier in the view
Contain a data element for every data field referenced in the view
Have converted the data in the view

If you make use of strings in `_MaskDescription,` your validation software must check that the XPath expression references a single value in an enumerated list:

```
DataLinkDescription="/LOAN/MORTGAGE_TERMS/[@MortgageType=’Fa
rmersHomeAdministration']"
```

Other References

See the other chapters in this section, Section 3, for other considerations when handling lining the data and view section and handling conversions.

See section 5 for information on the types of allowable views and requirements in the view section.

See Section 7 for specific information regarding requirements for the data section.

See Chapter 10: References for references to other documetss

Chapter 3.4: Operator Types

Operator Types: This section covers how to use operators in conversions.

Version 2.0

Revision History

Version	Date	Change
2.0	02/06/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

MISMO SMART Doc® Specification, Version 1.02

Overview

The purpose of mapping data from the XML data to the XHTML view is to provide the information necessary to validate that the contents of the view match the included XML data. In certain situations, the representation of the data in the data section may differ from what is presented in the view.

This chapter describes how to convert the data to a different representation in the view section and allow for many possible values for the conversion in the view. For instance, it may be acceptable for a field in the view to be blank or contain a value. This is accomplished by using an operator in the conversion that states that the view field may be blank OR it may contain a value.

The conversions are only applicable to SMART Docs that are Category One.

Pre Conditions

The state of the SMART Doc:
Document State: Any
Document Categories: 1

Post Conditions

The state of the SMART Doc:
Document State: Any
Document Categories: 1

Business Context

Business data can often be presented for viewing in numerous formats. For example, a date could be represented as 07/24/03, or as 2003-07-24 or as July 24, 2003 and either representation is acceptable. The desired representation may vary, depending on the actual data value.

Sometimes a field may have a value and in other cases it is acceptable to leave the field in the view blank. In these cases a conditional operator is employed to instruct the displaying system on how to determine which formats are valid and to indicate that multiple presentations are acceptable.

Scenario

The originating and receiving parties have a need to validate consistency in the data and view sections of each SMART Doc transferred. In some circumstances the display of the data in the view section differs from the representation in the view. This chapter describes how to implement data conversions that may have multiple and different values in the view section.

Technical Guidance

Operator types provide a mechanism to create decisions in data validation. The OR operator type can be used if you want to ensure the SMART Doc can be valid with the data missing. The <OPERATOR> element is used to combine several <ARC> elements in a Boolean fashion. The <OPERATOR> element has one attribute, _Type. This attribute indicates which Boolean operator ("AND" or "OR") is to be used. The <OPERATOR> element may contain a <CONVERSION> element for a mask of the combined elements.

This guidance describes the use of operators in three different situations. Situation One describes how to allow for a view field to contain a value or to be left blank. Situation 2 implements two different acceptable values for a numeric data field. Situation 3 describes the use of the AND operator. Please refer to the SMART Doc Specification for additional examples.

Situation One: View may be blank or have a value

The Interest Rate Percent data is linked to the view and either it is a number with three decimals and a % sign OR it is empty; i.e., it is "blank line". This is a case of one data field mapping to one view field, but it has two possible values.

Two <ARC> elements are constructed, each with a conversion. One for the view with a percent sign value and the other to indicate that the field may be left blank. Both <ARC> elements are wrapped in a parent <OPERATOR> element with the _Type attribute set to "OR":

```
<OPERATOR _Type="OR">
     <ARC
DataLinkDescription="//MORTGAGE_TERMS/@NoteRatePercent
"     ViewLinkDescription="id(MORTGAGE_TERMS-
NoteRatePercent)" >
          <CONVERSION _MaskDescription="#0.000%"
_Type="FillMask"/>
     </ARC>
     <ARC
DataLinkDescription="//MORTGAGE_TERMS/@NoteRatePercent
"   ViewLinkDescription="id(MORTGAGE_TERMS-
NoteRatePercent)" >
           <CONVERSION
_MaskDescription="NullEqUnderscore" _Type="FillMask"/>
     </ARC>
</OPERATOR>
```

Situation 2: Numeric or Text values acceptable

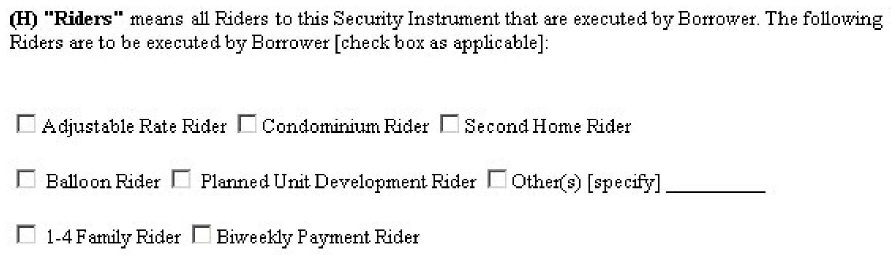
This example shows how the OR operator can be used to validate either a numeric value or a text value. Note that the implementation can use either of these masks when populating the view.

Each <ARC> element with a conversion is constructed. One for the view with a numeric value and one to indicate that the field has a textual representation of the number. Both <ARC> elements are wrapped in a parent <OPERATOR> element with the _Type attribute set to "OR":

```
<OPERATOR _Type="OR">
    <ARC
DataLinkDescription="//LOAN/_APPLICATION/LOAN_PRODUCT_
DATA/LOAN_FEATURES/LATE_CHARGE/@_GracePeriod"
ViewLinkDescription="id(LATE_CHARGE-_GracePeriod)">
    <CONVERSION _MaskDescription="words()"
_Type="FillMask"/>
    </ARC>
     <ARC
DataLinkDescription="//LOAN/_APPLICATION/LOAN_PRODUCT_
DATA/LOAN_FEATURES/LATE_CHARGE/@_GracePeriod"
ViewLinkDescription="id(LATE_CHARGE-_GracePeriod)">
<CONVERSION _Type="FillMask"
_MaskDescription="upper(words())"/>
     </ARC>
</OPERATOR>
```

Situation 3: Use of the AND Operator for two related view fields

This example of the operator AND will be used when the contents of multiple data fields must be in a particular state. For instance if a list of values is provided, and one of the choices is "other" and "other is selected", then the description field for "Other" must contain a value:


(H) "Riders" means all Riders to this Security Instrument that are executed by Borrower. The following Riders are to be executed by Borrower [check box as applicable]:

☐ Adjustable Rate Rider ☐ Condominium Rider ☐ Second Home Rider

☐ Balloon Rider ☐ Planned Unit Development Rider ☐ Other(s) [specify] ________

☐ 1-4 Family Rider ☐ Biweekly Payment Rider

The usage of this example is meant to validate that the Other checkbox has been checked and a value has been entered into the Other Rider Description.

Each <ARC> element with a conversion is constructed. One for the checkbox and one for the description of “Other”. Both <ARC> elements are wrapped in a parent <OPERATOR> element with the _Type attribute set to “AND”:

```
<OPERATOR _Type="AND">
     <ARC
DataLinkDescription="//LOAN/_CLOSING_DOCUMENTS/
RECORDABLE_DOCUMENT/RIDERS/@OtherRiderIndicator"
ViewLinkDescription="id(RIDERS-OtherRiderIndicator)">
          <CONVERSION _Type="FillMask"
_MaskDescription="bool(X)" />
     </ARC>
     <ARC
DataLinkDescription="//LOAN/_CLOSING_DOCUMENTS/
RECORDABLE_DOCUMENT/RIDERS/@OtherRiderDescription"
ViewLinkDescription="id(RIDERS-OtherRiderDescription)"
/>
</OPERATOR>
```

Other conversions will exist for the checkboxes with other values.

Please refer to the SMART Doc Specification for additional examples.

Checklist

- 9 Each <ARC> element with a conversion is constructed.
- 9 <ARC> elements are wrapped in a parent <OPERATOR> element
- 9 Select AND or OR depending on the mapping in the `_Type`:

XML Structures Used

This section provides a listing of the XML structures used, in the following format:

<ARC>
<CONVERSION>
<OPERATOR>

Known Issues

There are no requirements on the format of the data in the data section. For instance, your application may choose to represent the scheduled pay-off date in the data section as :

```
<LOAN_FEATURES ScheduledFirstPaymentDate="100101"/>
```

And another implementation may choose the following:

```
<LOAN_FEATURES ScheduledFirstPaymentDate="10/01/01"/>
```

If there are multiple possible values, your application will need to check for each acceptable conversion. XML DTD validation does not check for consistency within the <ARC> elements, nor will validation check that the conversions have been formatted correctly in the view. You must write additional software to check that the <ARC> elements correctly:
-- Specify a valid XPath in the data section
-- Include an identifier in the view
-- Contain a data element for every data field referenced in the view --
Have converted the data in the view

Other References

See the other chapters in this section, Section 3, for other considerations when linking the data and view section and handling conversions.

See section 5 for information on the types of allowable views and requirements in the view section.

See Section 7 for specific information regarding requirements for the data section.

See Chapter 10: References for references to other documents.

Chapter 4.1: Electronic Borrower Signatures

This chapter describes how to create signed documents by applying electronic text and image signatures for Borrowers.

Version 2.0

Revision History

Version	Date	Change
2.0	04/20/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc® Specification V 1.02
eMortgage Packaging Specification V 2.3

Electronic Signatures in Global and National Commerce Act ("ESIGN"), Pub. L. No. 106-229, 114 Stat. 464 (2000) (codified at 15 U.S.C. § 7001 *et seq*.).
https://www.fdic.gov/regulations/compliance/manual/10/x-3.1.pdf
Uniform Electronic Transactions Act ("UETA") (1999). The UETA is a model act drafted, approved, and recommended for enactment in all the states by The National Conference of Commissioners on Uniform State Laws (NCCUSL).
https://law.lis.virginia.gov/vacodepopularnames/uniform-electronic-transactions-act/

Overview

Electronic signatures come in many forms and can be implemented in multiple ways. It is very important to design the technology, as well as the process, so as to preserve the legal and technical integrity of the signatures, especially those used for borrowers in electronic mortgage documents.

One form of electronic signature – a digital signature – requires a different set of technologies to implement. To simplify the guidance around creating electronic signatures, we have reserved a separate chapter, Chapter 4.2, specifically for digital signatures. This chapter describes electronic signatures (except digital signatures) and how to apply them. Chapter 4.3 describes how apply tamper evident digital signatures to SMART Docs. Chapter 4.4 describes how to obtain and use a Digital Certificate. For information on documents in the signed state, see Chapter 2.4.

Pre Conditions

Document State: Signable
Document Categories: 1 or 2 only

Post Conditions

Document State: Signed

Business Context

In the paper world, it is easy for anyone to apply a distinctive mark to acknowledge and/or be bound to the terms of a paper document. In most cases, such marks can be attributed to a person or entity, and tampering can be detected to a very limited extent through forensic methods.

Electronic signature laws – ESIGN and UETA – have permitted the use of electronic signatures created by any sound, symbol or process applied with the intention to be bound by the signature. However, like ink signatures, electronic signatures most commonly represent the signer's name. Lenders and investors may have specific requirements for the types of signatures they accept for mortgage loans. However, there seems to be fairly broad agreement that sound- and video-type signatures will not be accepted on mortgage loan documents at this time.

Common Electronic Signatures

Electronic signatures can be implemented through a variety of technologies, which provide means for the borrower(s) to apply a distinctive mark to the document that represents their acknowledgement of, or agreement to, the contents of the electronic document. The list below provides examples of electronic signature technologies used within the SMART Doc Specification:

Comparison Table:

Technology	Description
"Typed" signatures, described in this chapter as *Text Signatures*	Signer types his/ner name or other distinguishing information (email address, PIN/password) and clicks a button to acknowledge act of signing. In other implementations, the signer's name may be already captured by the application and the signer simply clicking an "I sign" type button could apply the typed-name signature and the date of signing.
Holographic or digitized signature described in this chapter as *Image Signatures*	A hand-written signature is captured in real-time either through a signature pad or through scanning and embedded into document as a bitmap image, in variety of standard image formats (JPEG, GIF, PDF, etc).
Object Signatures	It is possible to implement other types of electronic, non-digital signatures in SMART Docs. Valid examples include biometrics or specialized signing applets. Relying parties must individually decide whether or not to accept these specialized types of signatures for particular business situations. This chapter does not cover object signatures; however, the SMART Doc Specification allows for the inclusion of biometric or specialized signing applets.
Digital Signatures (see Chapter 4.2 – Digital Borrower Signatures)	Uses cryptographic technologies to generate a unique numeric value from the signed document.

Scenario

A single borrower is at the closing table ready to electronically sign the eNote for the mortgage loan. The eNote is displayed on the computer screen in the Signable state. The borrower in this scenario will be applying an electronic signature. Each type of electronic signature will be described here for the same borrower.

Technical Guidance

There are two types of electronic signatures covered in this implementation guide: text and image. Implementations of object electronic signatures, such as those captured from a biometric device are not covered in this chapter.

Text or Typed Signature

A *Typed* signature consists of a typed representation of the signer's identity, usually his/her name, which represents the signer's acknowledgement and/or agreement to the document being presented to him/her.

There are multiple ways in which a typed signature may be implemented. Some examples include :

- Signers may type their name or other distinguishing information (such as an email address or a PIN/password) and click a button to acknowledge the act of signing
- Or the borrower's name may be already captured by the application and the borrower simply clicks an "I Sign" type button. The application applies the typed name signature directly.

Electronic Text Signature Implementation Process

The SMART Doc is in the "Signable" state. For information on the requirements for signable documents, see Chapter 2.3: "Making a SMART Doc Signable."

Step One: Change the Document State

Once the signatures are captured, the document must change to "Signed" state.

```
 <HEADER _ID="FNMA_Sample_Header_3200">
  <DOCUMENT_INFORMATION _StateType="Signed" /> </HEADER>
```

Step Two: Replace the Signature Line with the Text Signature

The `<SIGNATURE_LINE>` elements MUST be replaced with the actual signatures contained in `<SIGNATURE_TEXT>` elements, as shown on the example below:

```
<VIEW>
[. . .]
  <SIGNATURE_AREA _ID="Borrower1SignatureArea">
    <p class="right">
      <SIGNATURE_ABOVE_LINE/>
        <i>
            <SIGNATURE_TEXT _ID="Borrower1SignatureLine"> Electronically signed by
Richard R. Bradley on 2/1/2002 16:12:51 PST
            </SIGNATURE_TEXT>
        </i> (Seal) </p>
    <p class="right">
      <SIGNATURE_BELOW_LINE>
        <span class="dataEntered" id="BORROWER-_FirstName">
          Richard
        </span>
         <span class="dataEntered" id="BORROWER-_MiddleName">
          R.
        </span>
          <span class="dataEntered" id="BORROWER-_LastName">
          Bradley
        </span>
         - Borrower
      </SIGNATURE_BELOW_LINE>
    </p>
  </SIGNATURE_AREA>
[. . .]
</VIEW>
```

The `<SIGNATURE_TEXT>` element is required when applying a text signature.

The name of the Borrower is not required in the `<SIGNATURE_BELOW_LINE>`. However, this is common practice and the Borrower's name has been included in this example.

Image Signatures

A graphic signature is also known as a "holographic" or bitmap signature. It constitutes a graphical representation of the signer's handwritten signature and is evidence of the signer's acknowledgement and/or agreement to the document presented. Graphic signatures must be captured to standard image formats (such as JPEG. PNG or GIF) in files referenced from within the `<VIEW>` section.

The resolution of the image signature should be sufficiently detailed to be clearly legible, but not so large as to cause problems for display or storage of the document

Electronic Image Signature Implementation Process

The SMART Doc is in the "Signable" state. For information on the requirements for Signable documents, see Chapter 2.3: "Making a SMART Document Signable."

Step One: Change the Document State

Once the signatures are captured, the document must change to "Signed" state.

```
<HEADER _ID="FNMA_Sample_Header_3200">
  <DOCUMENT_INFORMATION _StateType="Signed" />
</HEADER>
```

Step Two: Replace the Signature Line with the Image Signature

The `<SIGNATURE_LINE>` elements must be replaced with the actual signatures contained in `<SIGNATURE_IMAGE>` elements.

```
<SIGNATURE_IMAGE _ID="Borrower1SignatureLine" _MIMEType="image/jpeg"
_EncodingTypeDescription="None">
    <img align="right"
          alt="Signature file is missing - Invalid Document"
          src="RichardRBradley.jpg"/>
```

```
</SIGNATURE_IMAGE>
```

Where:

- `_EncodingTypeDescription`: - The type of encoding used for the image file. Since the graphic signature is always contained in an external file, encoding must be set to "None".

- `_MIMEType` - The MIME Type of the signature file. For example, the MimeType will be set to "image/jpeg" for JPEG files and to "image/gif" for GIF files.

- `<img>` - The XHTML element for referencing images. The `src` attribute indicates the relative path to the external file containing the signature. The `Alt` attribute (Optional) contains text that is rendered in the event the graphic signature file is corrupted or not available. It is recommended to place in this attribute the following text: "Signature file is missing - Invalid Document".

Once the signatures are captured the `<SIGNATURE_LINE>` elements must be replaced with `<SIGNATURE_IMAGE>` elements containing the reference(s) to the actual signature file(s), as shown on the example below:

```
<VIEW>
[. . .]
  <SIGNATURE_AREA  _ID="Borrower1SignatureArea">
    <p class="right">
      <SIGNATURE_ABOVE_LINE/>
      <SIGNATURE_IMAGE  _ID="Borrower1SignatureLine" _MIMEType="image/jpeg"
_EncodingTypeDescription="None">
    <img align="right"
           alt="Signature file is missing - Invalid Document"
src="RichardRBradley.jpg"/>
</SIGNATURE_IMAGE>
(Seal) </p>
    <p class="right">
      <SIGNATURE_BELOW_LINE>
        <span class="dataEntered" id="BORROWER-_FirstName">
          Richard
        </span>
         <span class="dataEntered" id="BORROWER-_MiddleName">
          R.
        </span>
          <span class="dataEntered" id="BORROWER-_LastName">
          Bradley
</span>
         - Borrower
      </SIGNATURE_BELOW_LINE>
    </p>
  </SIGNATURE_AREA>
[. . .]
</VIEW>
```

With image signatures, the signature is external to the SMART Doc. The image signature file MUST be included in the SMART Doc package. For information on creating packages with image files, see Chapter 8.2, "Packages with Image Signatures." Image signatures have additional requirements on the tamperseal. The files must be referenced from the tamper-evident digital signature. For information on tamper-sealing SMART Docs see Chapter 4.3.

Power of Attorney (POA) executed eNotes

If the eNote is executed using a power of attorney, the following requirements must be met:

- The data section must provide the _POWER_OF_ATTORNEY element.
- The POA document must be packaged and delivered to MERS as described in section 8.1 Packages.

Sample XML:

The data section must provide the _POWER_OF_ATTORNEY element

```
<BORROWER BorrowerID="Brwr01" NonPersonEntityIndicator="N" _FirstName="XXXX"
_LastName="xxxx" _MiddleName="" _NameSuffix=""/>
    <BORROWER BorrowerID="Brwr02" NonPersonEntityIndicator="N" _FirstName="xxxx"
_LastName="xxxx" _MiddleName="" _NameSuffix="">
     <_POWER_OF_ATTORNEY _SigningCapacityTextDescription="as attorney in fact"
_UnparsedName="XXXXX"/>
    </BORROWER>
```

There must be an ARC element linking the _UnparsedName in POWER_OF_ATTORNEY to the view field.

```
<OPERATOR _Type="OR">
   <ARC
DataLinkDescription="//LOAN/_APPLICATION/BORROWER[2]/_POWER_OF_ATTORNEY/@_Signi
ngCapacityTextDescription" ViewLinkDescription="id(BORROWER2_POWER_OF_ATTORNEY-
_SigningCapacityTextDescription)"/>
   <ARC
DataLinkDescription="//LOAN/_APPLICATION/BORROWER[2]/_POWER_OF_ATTORNEY/@_Signi
ngCapacityTextDescription" ViewLinkDescription="id(BORROWER2_POWER_OF_ATTORNEY-
_SigningCapacityTextDescription)">
    <CONVERSION _MaskDescription="NullEqUnderscore" _Type="FillMask"/>
   </ARC>
  </OPERATOR>
  <OPERATOR _Type="OR">
   <ARC
DataLinkDescription="//LOAN/_APPLICATION/BORROWER[2]/_POWER_OF_ATTORNEY/@_Unp
arsedName" ViewLinkDescription="id(BORROWER2_POWER_OF_ATTORNEY-
_UnparsedName)"/>
```

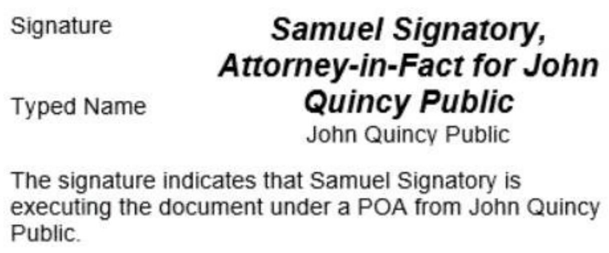
```
    <ARC
DataLinkDescription="//LOAN/_APPLICATION/BORROWER[2]/_POWER_OF_ATTORNEY/@_Unp
arsedName" ViewLinkDescription="id(BORROWER2_POWER_OF_ATTORNEY-
_UnparsedName)">
      <CONVERSION _MaskDescription="NullEqUnderscore" _Type="FillMask"/>
    </ARC>
   </OPERATOR>

<SIGNATURE_BELOW_LINE>
          ~~<span class="dataEntered" id="BORROWER2-_FirstName">xxxx</span>~~
          ~~<span class="dataEntered" id="BORROWER2-_LastName">xxxx</span>~~
          <span class="dataEntered" id="BORROWER2_POWER_OF_ATTORNEY-
_UnparsedName">xxxxx,</span>
          <span class="dataEntered" id="BORROWER2_POWER_OF_ATTORNEY-
_SigningCapacityTextDescription">Attorney-in-Fact for</span>
          <span class="dataEntered" id="BORROWER2-_FirstName">xxxx</span>
          <span class="dataEntered" id="BORROWER2-_LastName">xxxx</span>

          <span>    (Seal)</span>
          <span> - Borrower</span>
         </SIGNATURE_BELOW_LINE>
```

Sample View:

WITNESS THE HAND(S) AND SEAL(S) OF THE UNDERSIGNED.



Signature **_Samuel Signatory, Attorney-in-Fact for John Quincy Public_**

Typed Name John Quincy Public

The signature indicates that Samuel Signatory is executing the document under a POA from John Quincy Public.

Checklist 9 Header element `<DOCUMENT_INFORMATION>` changed to signed state
`<DOCUMENT_INFORMATION _StateType="Signed">`

9 Replacement of `<SIGNATURE_LINE>` with `<SIGNATURE_TEXT>` or
`<SIGNATURE_IMAGE>`

9 Text indicating signature “Electronically signed by …” or `<img>` tag referenced

9 Unique ID created for the signature
`<SIGNATURE_TEXT _ID="Borrower1SignatureLine">`

9 Unique ID referenced by the correct signer in the header
`<SIGNATURE_MODEL>`

9 `<SIGNER SignatureIDREF="Borrower1SignatureLine">`

XML Structures Used

Known Issues

Allowing for electronic signatures within the XHTML requires changes to the XHTML transitional DTDs.

Other References

See Chapter 10: References.

Chapter 4.2: Digital Borrower Signatures

This chapter describes how to create signed documents by applying digital signatures for borrowers.

Version

2.0

Revision History

Version	Date	Change
2.0	02/27/2019	Corrections and clarifications
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc® Specification V 1.02
XML Digital Signature Recommendation of the W3C.
(http://www.w3.org/TR/xmldsig-core/).

Overview

Digital signatures represent a special type of electronic signature that due to their complexity merit a special discussion. Digital signatures, when properly implemented, may provide a high degree of attribution or signer authentication, message or document integrity (tamper evidence) and may provide nonrepudiability (meaning a signer cannot later deny having signed the document).

A digital signature is used in SMART Docs to provide a tamper-evident wrap. This chapter addresses implementing tamperseal digital signature.

Pre Conditions

Document State: Signable
Document Categories: 1 – 5

Post Conditions

Document State: Signed

Business Context

In the paper world, it is easy for anyone to apply a distinctive mark to acknowledge and/or be bound to the terms of a paper document. Such marks, in most cases can be attributed to a person or entity, and tampering can be detected to a very limited extent through forensic methods.

Borrower's signatures (or those of other parties like sellers or witnesses) are most commonly created as "text" or as "image" signatures. However, a borrower signature could also be created through the use of a digital signature. This chapter describes how a digital signature should be applied for the purpose of creating a borrower's signature using a W3C XML digital signature.

Digital signatures may be used by relying parties to identify the entity(ies) or person(s) that signed the document. While an application may not necessarily rely on the authenticity of digital signatures for borrowers and other participants, these signatures must meet certain minimum structural elements in order to be validated for integrity purposes. Additionally, digital signatures have minimum structural requirements to ensure compatibility with other signature elements in the SMART Doc specification.

Comparison Table:

Technology	Description
"Typed" signatures, described as *Text Signatures* (see Chapter 4.1 – Electronic Borrower Signatures)	Signer types their name or other distinguishing information (email address, PIN/password) and clicks a button to acknowledge act of signing. In other implementations, the borrower's name may be already captured by the application and the borrower simply clicking an "I sign" type button could apply the typed-name signature and the date of signing.
Holographic or digitized signature described as *Image Signatures* (see Chapter 4.1 – Electronic Borrower Signatures)	A hand-written signature is captured either through a signature pad or through scanning and embedded into document as a bitmap image, in variety of standard image formats (JPEG, GIF, PDF, etc).
Object Signatures	It is possible to implement other types of electronic, non-digital signatures in SMART Docs. Valid examples include biometrics or specialized signing applets. Relying parties must individually decide whether or not to accept these specialized types of signatures for particular business situations. This chapter does not cover object signatures; however, the SMART Doc Specification allows for the inclusion of biometric or specialized signing applets.
*Digital Signature*s (described in this chapter)	Uses cryptographic technologies to generate a unique numeric value from the signed document.

Scenario

Our borrowers are at the closing table ready to use a digital signature to sign their promissory note for their mortgage loan. The eNote is displayed on the computer screen in the signable state. The borrowers in this scenario will each be applying a digital signature. The borrowers are going to use a smart card to sign their names.

Technical Guidance

As with any operation, digital signatures can be implemented through a variety of technical means and standards. Generally, Authenticated Digital Signatures for SMART Docs presented to relying parties should be implemented according to the following specifications:

Technology

All SMART Doc digital signatures MUST be created using public key technology by using X.509 v3 (X.509) digital certificates. Private keys associated with X.509 digital certificates must be securely generated and stored by the certificate holder.

Acceptable Certification Authorities

For Authenticated Digital Signatures Relying parties will need to decide which certification authorities and certificate policies to trust for specific applications. REFSMO/SISAC has implemented an accreditation program for secure identity providers, to assist relying parties in making these decisions. Certain applications may have specific requirements, such as specific certificate policy Identifiers, assurance levels or hardware storage of private keys.

Basic Requirements

The basic requirements for digital signatures for signer roles other than "Tampersealer" include:

- The X.509 digital certificate used to create the digital signature must be included in Base64 encoded form in the `<X509Certificate>` tag in the corresponding signature element.
- Both RSA and DSA signature algorithms are supported.
- The following W3C XML Signature parameters MUST be included in each digital signature representation: `<X509IssuerSerial>, <X509IssuerName>, <X509SerialNumber>, <X509SubjectName>`.
- Other elements and structures permitted by the XML Dsig standards MAY be included in the document.

Digital Signature Implementation Process

Step One: Change the Document State

Once the signatures are computed, the document must change to "Signed" state.

```
<HEADER _ID="FNMA_Sample_Header_3200">
    <DOCUMENT_INFORMATION _StateType="Signed" />
</HEADER>
```

Step Two: Set the SIGNER attributes

Set the `SignatureType` attribute to "`DigitalSignature`", add the order in which the Borrower signed the document in the `_SignatureOrderNumber` attribute, and add the role of the Signer to the `RoleType` attribute in `<SIGNER>`:

```
<SIGNER SignatureType="DigitalSignature" _RoleType="Borrower"
_SignatureOrderNumber="1"/>
```

Additionally, set the `SignatureIDREF` to reference the unique `_ID` of the digital signature line in the <`VIEW`>, the `SectionIDREF`and `AreaIDREF` to reference the <`SIGNATURE_SECTION`> and <`SIGNATURE_AREA`> of the Borrower's signaturein the <`VIEW`>, and the `TargetsIDREF` attribute to reference the <`VIEW`> that the digital signature applies to.

```
<SIGNER AreaIDREF="Borrower1SignatureArea"
SectionIDREF="BorrowerSignatures"
SignatureIDREF="Borrower1SignatureLine" SignatureType="DigitalSignature"
TargetsIDREFS="FNMA_Sample_View_3200" _RoleType="Borrower"
_SignatureOrderNumber="1"/>
```

Step Three: Compute and add the Digital Signature elements

The digital signature is enclosed in the <`SIGNATURES`> element. The following is an example of digital signature, using DSA, referenced by the <`SIGNATURE_OBJECT`> element above:

```
<SIGNATURES>
   <Signature Id="Borrower1Signature"
xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
         <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
            <Reference URI="#Borrower1SignatureArea">
               <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <DigestValue>RKeWU6WWEWyKpl8spn97kBXt6y0=</DigestValue>
            </Reference>
      </SignedInfo>
      <SignatureValue> bTU6JQJse1eu5aeD ….S5FL/Q== </SignatureValue>
      <KeyInfo>
       <KeyValue>
        <DSAKeyValue>
         <P>/X9TgR11EilS30qcLuzk5/Y …. gMZndFIAcc=</P>
               <Q>l2BQjxUjC8yykrmCouuEC/BYHPU=</Q>
  <G>Q7zKTx…..6Ae1UlZAFMO/7PSSo=</G>
          <Y>cmJPQKtej…VfHq+HDtZ2HVHOPgs817wJ7kCj8D80a3TnfuyhY=
         </Y>
        </DSAKeyValue>
       </KeyValue>
       <X509Data>
        <X509IssuerSerial>
         <X509IssuerName>
CN=Richard\20R\20Bradley,OU=Unknown,O=Unknown,L=Unknown,ST=NC,C=
Unknown
         </X509IssuerName>
         <X509SerialNumber>1030562998</X509SerialNumber>
        </X509IssuerSerial>
```

```
          <X509SubjectName>
CN=Richard\20R\20Bradley,OU=Unknown,O=Unknown,L=Unknown,ST=NC,C=
Unknown
          </X509SubjectName>
            <X509Certificate> ….mZVXwptpyIDEpzuQhK4=    </X509Certificate>
         </X509Data>
        </KeyInfo>
    </Signature>
</SIGNATURES>
```

Note: This digital signature example has been modified for readability purposes and will not validate with XML digital signature tools.

Step Four: Define Signatures in the VIEW Section: "Signed"

A digital signature consists of the XML digital signature (which is not easily viewable) and a typed representation of the signer's identity, usually his/her name, which represents the signer's acknowledgement and/or agreement to the document being presented to him/her.

Once the borrowers have signed with their smart card, and the digital signatures are captured, the document must change to the "Signed" state, and the `<SIGNATURE_LINE>` elements must be replaced with a `<SIGNATURE_OBJECT>` element that includes a reference to the digital signature contained in `<Signature>` elements. This is shown on the example below:

```
<VIEW>
        [. . .]
        <SIGNATURE_AREA _ID="Borrower1SignatureArea">
            <p class="right">
                <SIGNATURE_ABOVE_LINE/>
                <SIGNATURE_OBJECT _EncodingTypeDescription="none"
                                    Href="#Borrower1Signature"
                                     _ID="Borrower1SignatureLine"
                                    _MIMEType="text/xml">
                          Digitally signed by Richard R. Bradley on 7/31/2002 18:07
(Seal)</SIGNATURE_OBJECT></p>
            <p class="right">
                <SIGNATURE_BELOW_LINE>
                              <span class="dataEntered"
id="BORROWER_FirstName">Richard</span>
                <span class="dataEntered"/>
                              <span class="dataEntered"
id="BORROWER_MiddleName">R.</span>
                <span class="dataEntered"/>
                <span class="dataEntered" id="BORROWER-
_LastName">Bradley</span>   - Borrower</SIGNATURE_BELOW_LINE>
            </p>
            <p class="right">
                <SIGNATURE_BELOW_LINE>
```

```
   <span class="dataEnteredRight" id="BorrowerName1">Richard R.
Bradley</span>
                </SIGNATURE_BELOW_LINE>
             </p>
             </SIGNATURE_AREA>
        [. . .]
</VIEW>
```

The syntax of the <`SIGNATURE_OBJECT`> element is as follows:

```
<SIGNATURE_OBJECT _EncodingTypeDescription="none"
                         Href="#Borrower1Signature"
                          _ID="Borrower1SignatureLine"
                         _MIMEType="text/xml">
                          Digitally signed by Richard R. Bradley on 7/31/2002 18:07
(Seal)</SIGNATURE_OBJECT>
```

Where:

- _EncodingTypeDescription: The type of encoding used. Since the digital signature is XML the encoding should be set to "None".
- _MIMEType: The MIME Type of the signature file. The MIME Type will be set to "text/xml"...
- Href: The Href attribute indicates the relative path to the digital signature. If an external file contains the signature, the Href attribute will contain the relative path to the file and the filename. If the digital signature is within the same document, an XPath expression is used.
- _ID: The _ID attribute is the unique identifier for digital signature placeholder in the <VIEW>.

Checklist

9 Header element `<DOCUMENT_INFORMATION>` changed to signed state
`<DOCUMENT_INFORMATION _StateType="Signed">`

9 Add the `<SIGNER>` element with the appropriate attributes

9 Replacement of `<SIGNATURE_LINE>` with `<SIGNATURE_OBJECT>`

9 Text indicating signature "Digitally signed by …"

9 Unique ID created for the signature
`<SIGNATURE_OBJECT _ID="Borrower1SignatureLine">`

9 Unique ID referenced by the correct signer
`<SIGNER SignatureIDREF="Borrower1SignatureLine">`

9 Addition of the Digital Signature `<Signature>` to the `<SIGNATURES>` section

9 Apply the "Signed" `<AUDIT_TRAIL>` entry

XML Structures Used

This section provides a listing of the XML structures used, in the following format:

`<DOCUMENT_INFORMATION>`

`<SIGNER>`
`<SIGNATURE_OBJECT>`
`<SIGNATURES>`

Known Issues

Although standards exist, XML digital signatures are relatively new. There are some allowances in the specification that result in slight differences in interpretation. These allowances may cause interoperability issues depending upon the specific library or toolkit used to create signatures.

Other References

See Chapter 10: References.

Chapter 4.3: Tamper Evident Signatures

This chapter describes how apply tamper evident digital signatures to SMART Docs®.

Version 2.0

Revision History

Version	Date	Change
2.0	02/27/2019	Updates and corrections
1.0	01/26/2004	Release to industry

:

Relevant Specifications

SMART Doc Specification V 1.02
XML Digital Signature Recommendation of the W3C.
(http://www.w3.org/TR/xmldsig-core/).

Overview

Digital signatures represent a special type of electronic signature that merit a special discussion, due to their complexity. Digital signatures, depending on their implementation, may provide one or more of the following qualities: attribution or signer authentication, message or document integrity (tamper evidence) and non-repudiability (meaning a signer cannot later deny having signed the document). However, parties who rely on digital signatures may choose to only rely on one or more of the previously mentioned qualities.

All Category 1 SMART Docs MUST include at least one digital signature. The minimum required digital signature MUST be used as a tamper-evident wrap after all other signatures have been affixed. The tamperseal MUST be created and applied immediately following the other electronic signatures in the document. The tamperseal MUST be applied at the earliest possible time, before any post-processing is performed on the document by an application or system. (Note: The exact requirements of this timeframe will be determined by agreements between business partners or by investor delivery requirements).

Pre Conditions

For the tamperseal signature: Document
State: Signed
Document Categories: 1 – 5

Post Conditions

Document State: Signed
Last signer: _RoleType = “Tampersealer”

Business Context

XML Digital signatures are digital signatures represented in XML. Digital signatures assure the following:

- Acceptance - Agreement of the signer with the content of the SMART Doc.
- Integrity - That the SMART Doc has not been changed in any way
- Authenticity - The SMART Doc actually came from a sender that is referenced in the signature
- Non Repudiation - The sender of the document cannot claim that they did not send it.

Digital signatures are created and verified using public-key cryptography, a branch of applied mathematics, and may be applied to any type of digital content.

Cryptography transforms information into disguised and unintelligible forms and back again. For digital signatures, two different keys are generally used: one for creating a digital signature, the "private key," and one for verifying a digital signature, the "public key." The private key transforms data into a seemingly unintelligible form and the public key returns the information to its original form. The private key uniquely identifies the signer and access to the private key is restricted.

The proper use of digital signatures is comprised of two separate but equal processes. The first step is performed by the signer, computing a message digest over the data to be signed, using a mathematical function called a *hash algorithm*, and then encrypting the message digest with the signer's private key.

The second step is performed by the recipient. A recipient must have access to the corresponding public key in order to verify that a digital signature is the signer's. The receiver verifies the integrity of the signature by comparing the hash value to the decrypted signature using the public key. Encryption protects the hashed value and decryption is only successful using the public key. The XML Digital signature is a method of associating keys with referenced data.

After all signatures of any type have been applied, a final digital signature is applied to the document. For SMART Docs, this final signature is called the tamperseal signature.

Digital signatures may be used for both authentication and document integrity. For tamperseal signatures, the XML digital signature is used to validate document integrity only. There is no requirement to verify the issuer of the certificate.

Digital signatures may be used for other types of signatures in the document. For information on using a digital signature for borrower signatures, see chapter 4.2, Digital Borrower Signatures.

Scenario

All borrowers have electronically signed the promissory note for their mortgage loan. The LOS or closing application must now apply a tamperseal digital signature.

Technical Guidance

As with any operation, digital signatures can be implemented through a variety of technical means, standards and tools. Generally, tamperseal digital signatures for SMART Docs presented to relying parties should be implemented according to the following specifications:

Technology

All SMARTDOC digital signatures MUST be created using public key technology by using X.509 v3 digital certificates. Private keys associated with X.509 digital certificates must be securely generated and stored by the certificate holder. The X.509 certificate data must include the Issuer of the identity credential (e.g., name in Issuer field in X.509 certificate), and the subscriber identity named in the identity credential (i.e. the subject name in the X.509 certificate). For Tamperseal signatures, the Relying Party would normally accept X.509 v3 certificates from any certification authority, since The Relying Party does not rely on the identity assertion that such certificates make. The other requirement is that the digital signature must include a date and timestamp as to when the signature was applied. These requirements are discussed in detail in the steps below.

For information about applying an XML digital Signature for other purposes (such as a means to authenticate and validate integrity while exchanging SMART Docs between trading partners) see chapter 4.4. Applying Digital Signatures for Authentication and Integrity Validation.

Acceptable Certification Authorities

For Authenticated Digital Signatures Relying parties will need to decide which certification authorities and certificate policies to trust for specific applications. REFSMO/SISAC has implemented an accreditation program for secure identity providers, to assist relying parties in making these decisions. Certain applications may have specific requirements, such as specific certificate policy Identifiers, assurance levels or hardware storage of private keys.

Implementation Steps

Step One: Obtain a certificate

The Public Key of the signing Certificate will be included in the digital signature. The Certificate must contain at least:

- Your organization's name and address
- Your own public key
- Name of the Certificate issuer
- Serial number of the Certificate

Step Two: Modify the <HEADER>

Set the `<SIGNER>` attributes within the `<SIGNATURE_MODEL>` Element.

Set the `SignatureType` attribute to “`DigitalSignature`”, add the order in which the Borrower signed the document in the `_SignatureOrderNumber` attribute, and add the role of the Signer to the `_RoleType` attribute in `<SIGNER>` element is set to "`Tampersealer”:`

```
<SIGNER SignatureType="DigitalSignature"
_RoleType="TamperSealer" _SignatureOrderNumber="2"/>
```

Additionally, set the `SignatureIDREF` to reference the unique `_ID` attribute of the digital signature, and the `TargetsIDREFS` attribute to list all areas of the SMART Doc that the digital signature applies to; i.e., the `TargetsIDREFS` attribute must include all intra-document references within the XML digital signature.

```
<SIGNER SignatureIDREF="TamperSealer01"
SignatureType="DigitalSignature"
TargetsIDREFS="FNMA_Sample_Header_3200
FNMA_Sample_Data_3200 FNMA_Sample_View_3200
SignedContent01 TamperSealer01"
_RoleType="TamperSealer" _SignatureOrderNumber="2"/>
```

Step Three: Timestamp Requirements

In order for the dates and times in SMART Docs to be consistent and usable, all times must be coordinated to a standard clock. The clock at Greenwich, England is used as the standard clock for international reference of time. The letter designator for this clock is Z. This time is sometimes referred to as Zulu Time because of its assigned letter. Times are written in military time or 24 hour format such as 1830Z. The official name is Coordinated Universal Time or UTC. Previously it had been known as Greenwich Mean Time or GMT but this has been replaced with UTC. See http://www.iso.org/iso/en/prodsservices/popstds/datesandtime.html for more information on UTC formats and examples.

The time value for any timestamp, whether it is in the audit trail or is the tamperseal timestamp must be universal time (that is to say, not a local time with a time zone offset).

The XML Digital Signature specification does not have a standard way to include the signature timestamp. The XML Digital Signature DTD that is distributed with the SMART Doc Framework DTDs has been modified to include a timestamp under the SignatureProperties element:

```
<!ELEMENT DateTimeStamp EMPTY>
<!ATTLIST DateTimeStamp DateTime CDATA #REQUIRED>
```

The timestamp will be included in the digital signature after the `<KeyInfo>` element:

```
      <Object>
        <SignatureProperties>
          <SignatureProperty Id="TamperSealer01_DTS"
Target="TamperSealer01">
            <DateTimeStamp DateTime="2003-09-
23T15:24:08Z"/>
          </SignatureProperty>
        </SignatureProperties>
      </Object>
```

Please note, that the timestamp must be signed along with the other data.

Step Four: Compute and Add the Digital Signature

The digital signature is enclosed in the `<SIGNATURES>` element. There are several toolkits that will create XML digital signatures. XML digital signatures should be created according to IETF RFC3275. Although this is a standard, it is relatively new, allowing for some slight differences in interpretation that may cause interoperability issues depending upon the specific library or toolkit used to create signatures. It is recommended to use the latest toolkits available and to pay special attention to XML parser settings (whitespace handling) and the referenced canonicalization method of the tool.

In constructing the digital signature, you MUST add the following:

1) a unique identifier for the signature

```
<Signature Id="TamperSealer01">
```

2) The X.509 certificate data must include the Issuer of the identity credential (e.g., name in Issuer field in X.509 certificate and a serial number unique to the issuer's domain)

```
<X509IssuerSerial>
<X509IssuerName>Name of the issuer</X509IssuerName>
<X509SerialNumber>unique to the issuer’s domain
</X509SerialNumber>
</X509IssuerSerial>
```

3) a subscriber identity named in the identity credential (i.e. the subject name in the X.509 certificate)

```
<X509SubjectName>CN=eSolutions, OU=eBusiness,
O=Fannie Mae, L=DC, ST=Washington,
C=US</X509SubjectName
```

4) References to the header, data and view sections and the signatures within the document

```
<Reference URI="#FNMA_Sample_Header_3200">
<Reference URI="#FNMA_Sample_Data_3200">
<Reference URI="#FNMA_Sample_View_3200">
```

```
<Reference URI="#SignedContent01">
<Reference URI="#TamperSealer01_DTS">
```

The following is an example of digital signature:

```
<SIGNATURES>
    <Signature Id="TamperSealer01">
      <SignedInfo>
        <CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsasha1"
/>
        <Reference URI="#FNMA_Sample_Header_3200">
          <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>NIRYOZXOANH8g+etOeWOnclzKpI=</DigestValu
e>
        </Reference>
        <Reference URI="#FNMA_Sample_Data_3200">
          <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>/q6ptzFzZ0iZgXASWzzbTlypLs0=</DigestValu
e>
        </Reference>
        <Reference URI="#FNMA_Sample_View_3200">
          <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>ZMpMn95bGeq3NWGOmIXg1lGfuzE=</DigestValu
e>
        </Reference>
        <Reference URI="#SignedContent01">
          <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>lPTgpWdP6yQAoXt1s+qP068ajpA=</DigestValu
e>
        </Reference>
      </SignedInfo>
      <SignatureValue>

gdzAI3DobOYsvjR/1jRn/oaMgWIZv5RpUeSmiR74zdUQA0MN9X+9V
PmzLIG8S1t7

6KOr17u2luZOu2Prg/5kQFQ1NJpCSTaz//FyFxO3WlHy8a+gqF6+d
/5b4dBo24Br
```

```
        /AQTn6TiTltOuzyOidPpVlQsyyRYSOMgkqpOj3mcaX4=
      </SignatureValue>
      <KeyInfo>
        <X509Data>
          <X509IssuerSerial>
<X509IssuerName>Name of the
issuer</X509IssuerName>
          <X509SerialNumber>unique to the issuer’s
domain
           </X509SerialNumber>
          </X509IssuerSerial>
          <X509SubjectName>
            CN=eSolutions, OU=eBusiness, O=Fannie
Mae, L=DC, ST=Washington, C=US
          </X509SubjectName>
          <X509Certificate>
MIIDGzCCAoSgAwIBAgIBADANBgkqhkiG9w0BAQQFADBtMQswCQYDV
QQGEwJVUzET


//2ZCdQA6iFXNVNzeeDtQKjklu3c06f7hK9jlu+Azw==
          </X509Certificate>
        </X509Data>
      </KeyInfo>
    </Signature></SIGNATURES>
```

Note: This digital signature example has been modified for readability purposes and will not validate with XML digital signature tools.

Checklist

9 Type attribute in <SIGNER> element is set to "DigitalSignature"

9 _RoleType attribute in <SIGNER> is set to "Tampersealer"

9 SignatureIDREF in <SIGNER> element references the unique _ID of the digital signature

9 TargetsIDREFS attribute in <SIGNER> element lists all intra-document references within the XML digital signature

9 There is a unique identifier for the signature <Signature Id="TamperSealer01">

9 The X.509 certificate data must include the name in Issuer field in X.509 certificate and a serial number unique to the issuer's domain <X509IssuerSerial>element

9 A subscriber identity is named in the identity credential (i.e. the <X509SubjectName> subject name in the X.509 certificate)

9 References to the header, data and view sections and the signatures within the document as well as the time stamp

9 The tamper evident digital signature contains the date and time of application in the timestamp

9 Add "Signed <AUDIT_TRAIL> entry.

XML Structures Used

<SIGNER>
<SIGNATURES>
<Signature>

Known Issues

Other References

See other chapters in Section 4 for information on other types of signatures in SMART Docs.

See Chapter 10: References for all other references.

Chapter 4.4: Applying Digital Signatures for Authentication and Integrity Validation.

This chapter describes how to obtain and use a Digital Certificate.

Version

2.0

Revision History

Version	Date	Change
2.0	02/28/2019	Corrections and clarifications
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc® Specification V 1.02

Overview

Digital signatures represent a special type of electronic signature that merit a special discussion, due to their complexity. Digital signatures, when properly implemented, may provide a high degree of attribution or signer authentication, message or document integrity (tamper evidence) and may provide non-repudiability (meaning a signer cannot later deny having signed the document).

All Class 1 SMART Docs MUST include at least one digital signature that is used to validate document integrity; that is, provide a means determining if the file has or has not been tampered with in any way. This special case of a digital signature applied to a SMART Doc is known as the “tamperseal” signature for short.

There will be cases when trading partners require the ability to authenticate the sender of the document in addition to its integrity.

The exact requirements of this type of digital signature will be determined by agreements between business partners, or by investor delivery requirements.

Pre Conditions

Document State: Any
Document Categories: 1 – 5

Post Conditions

Document State: Any
Document Categories: 1 – 5

Business Context

XML Digital signatures are digital signatures represented in XML. Digital signatures assure the following:

Acceptance - Agreement of the signer with the content of the SMART Doc.
Integrity - That the SMART Doc has not been changed in anyway
Authenticity - The SMART Doc actually came from sender that is referenced in the signature
Non Repudiation - The sender of the document cannot claim that they did not send it.

Digital signatures are created and verified using cryptography, a branch of applied mathematics, and may be applied to any type of digital content.

Cryptography transforms information into disguised and unintelligible forms and back again. For digital signatures, two different keys are generally used: one for creating a digital signature, the "private key," and one for verifying a digital signature, the "public key." The private key transforms data into a seemingly unintelligible form and the public key returns the information to its original form. The private key uniquely identifies the signer and access to the private key is restricted.

The proper use of digital signatures is comprised of two separate but equal processes. The first step is performed by the signer that encrypts the data to be signed by using the private key and a hash value for the signature area. The hash value is the result of a mathematical method of detecting change over a digital area. The second step is performed by the recipient.

A recipient must have access to the corresponding public key in order to verify that a digital signature is the signer's. The receiver verifies the integrity of the signature by comparing the hash value to the decrypted signature using the public key. Encryption protects the hashed value and decryption is only successful using the public key. The XML Digital signature is a method of associating keys with referenced data.

Digital signatures may be used for both authentication and document integrity. For tamperseal signatures, the XML digital signature is used to validate document integrity only. There is no requirement to parse the certificate in the digital signature or verify the issuer of the certificate. However, your implementation may require authentication of the sender of the SMART Doc. In this situation, you must authenticate the sender's certificate.

This chapter describes how to create an XML digital Signature that includes certificates, key names, and key agreement algorithms.

Technical Guidance

As with any operation, digital signatures can be implemented through a variety of technical means and standards.

Acceptable Certification Authorities

The use of specific certificate types and issuers will be jointly determined by trading partners for different types of transactions and is not part of this Specification.

Technology

All SMARTDOC digital signatures MUST be created using public key technology by using X.509 v3 digital certificates. Private keys associated with X.509 digital certificates must be securely generated and stored by the certificate holder. The X.509 certificate data must include the Issuer of the identity credential (e.g., name in Issuer field in X.509 certificate), and the subscriber identity named in the identity credential (i.e. the subject name in the X.509 certificate).

The `<KeyInfo>` element contains information about the certificate and the certificate holder. It is not required by the XML Digital Signature DTD. Every SMART Doc digital signature must include this element.

For information about applying an XML digital Signature as a tamperseal, see chapter 4.3. Tamperseal Signatures.

Step One: Obtain a Digital Certificate

In order to create a digital signature, you must have a digital certificate.

Step Two: Add the Digital Signature

The digital signature is enclosed in the `<SIGNATURES>` element. The following is an example of digital signature:

```
<SIGNATURES>
    <Signature Id="Digs01">
      <SignedInfo>
        <CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <Reference URI="#FNMA_Sample_Header_3200">
          <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>NIRYOZXOANH8g+etOeWOnclzKpI=</DigestValue>
        </Reference>
        <Reference URI="#FNMA_Sample_Data_3200">
          <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>/q6ptzFzZ0iZgXASWzzbTlypLs0=</DigestValue>
</Reference>
        <Reference URI="#FNMA_Sample_View_3200">
          <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>ZMpMn95bGeq3NWGOmIXg1lGfuzE=</DigestValue>
        </Reference>
      </SignedInfo>
      <SignatureValue>

gdzAI3DobOYsvjR/1jRn/oaMgWIZv5RpUeSmiR74zdUQA0MN9X+9VPmzL
IG8S1t7

6KOr17u2luZOu2Prg/5kQFQ1NJpCSTaz//FyFxO3WlHy8a+gqF6+d/5b4
dBo24Br
        /AQTn6TiTltOuzyOidPpVlQsyyRYSOMgkqpOj3mcaX4=
      </SignatureValue>
      <KeyInfo>
        <X509Data>
          <X509SubjectName>
            CN=eSolutions, OU=eBusiness, O=Fannie Mae,
L=DC, ST=Washington, C=US
          </X509SubjectName>
          <X509Certificate>
MIIDGzCCAoSgAwIBAgIBADANBgkqhkiG9w0BAQQFADBtMQswCQYDVQQGE
wJVUzET
            //2ZCdQA6iFXNVNzeeDtQKjklu3c06f7hK9jlu+Azw==
          </X509Certificate>
        </X509Data>
      </KeyInfo>
    </Signature>
</SIGNATURES>
```

Note: This digital signature example has been modified for readability purposes and will not validate with XML digital signature tools.

Step Three: Add the Certificate Information

The `<KeyInfo>` element indicates the key to be used to validate the signature. Identification mechanisms can include certificates, key names, and key agreement algorithms. The <KeyInfo> element is optional in the W3C specification. For SMART Docs that require authentication, this element is required.

The <X509Data> element under <KeyInfo> contains a host of information about the certificate used to sign the document.

The actual verification certificate is in <X509Certificate>.

When a Certificate Authority issues as certificate, the certificate is assigned a unique serial number. This serial number may not be unique across Certificate Authorities. The <X509IssuerSerial> element identifies the certificate issuer's name and its serial number.

Further information may be encoded in the <X509SubjectName> element. This element uniquely identifies a particular end-entity including name (the name of a person or a software-server based application), organization, and location.

```
<KeyInfo>
```

```
        <X509Data>
          <X509SubjectName>
            CN=eSolutions, OU=eBusiness, O=Fannie Mae,
L=DC, ST=Washington, C=US
          </X509SubjectName>
          <X509Certificate>
MIIDGzCCAoSgAwIBAgIBADANBgkqhkiG9w0BAQQFADBtMQswCQYDVQQGE
wJVUzET
            //2ZCdQA6iFXNVNzeeDtQKjklu3c06f7hK9jlu+Azw==
          </X509Certificate>
        </X509Data>
      </KeyInfo>
```

Checklist

9 Obtain a digital signature certificate either from a certificate authority or of your own creation

9 Create and add the digital signature to the SMART Doc

9 Ensure that the `<KeyInfo>` element exists and includes the certificate, key names, and key agreement algorithms

9

XML Structures Used

```
<SIGNATURES>
<Signature>
```

Known Issues

You will need to consult with your trading partners on the specific requirements for digital signing of SMART Docs for purposes other than the tamperseal.

Other References

See Section 4 for information on types of signatures in SMART Docs.

See Chapter 10: References for all other references.

Chapter 5.1: Implementing Tagged Views

This chapter describes how to create tagged views in a SMART Doc®.

Version

2.0

Revision History

Version	Date	Change
2.0	03/06/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification V 1.02

Overview

A SMART Doc VIEW is the visual representation of the document. In addition to the document narrative text, it may also present the data from the DATA section and any signatures on the document. The Specification allows for two kinds of views:

Tagged Views: The visual representation of the document is contained inside the SMART Doc, expressed as XHTML, i.e., it is a marked up view. At present the only supported tagged view format is XHTML.

Image Views: Any other representation of the document other than XHTML, such as PDF, TIFF, etc.

This chapter demonstrates how tagged views are created and specific requirements for tagged views. Please refer to Chapter 5.2 for a discussion of image views.

An eNote must have one and only one view, and it must be a tagged view.

Pre Conditions

Document State: Any
Document Categories: 1 and 2

Post Conditions

Document State: Any
Document Categories: 1 and 2

Business Context

In the paper world, there is usually no distinction made between the content of a document (form) and the data that is merged into the form (filled-out fields). As the document is filled out at various points in its life cycle, the data is merged in and becomes part of the visual document. The SMART Doc Specification models this process as discrete states that define how the document presentation – its VIEW – is manipulated as it flows through its life cycle.

With the information and its presentation, and the relationship between the two bound in a single tamper-evident file, the integrity of the electronic data can be guaranteed. In other words, this specification allows system validation to ensure that what the borrower sees and signs on the computer screen is the exact document that will be stored. It also ensures that the data displayed on the screen will be the exact data used for downstream processing of the loan.

In order to manipulate the VIEW, it is represented in a structured XHTML format, known as a Tagged View. If the document is a category 1 or 2 SMART Doc, the data in the DATA section is linked to its corresponding representation in the VIEW. This linkage provides a mechanism to verify the integrity of the data and the presentation of the view within a single file.

It is technically possible to have tagged views other than XHTML, such as SVG or Scalable Vector Graphics and tagged PDF. The inclusion of other types of tagged views is targeted for a future version of the SMART Doc specification.

Technical Guidance

There are three kinds of tagged views:

- Internal view: The XHTML view is contained inside the SMART Doc. Please note that it is possible that an internal tagged view may have external references, for instance when a company logo or an image file containing the borrower's signature is included in the SMART Doc.
- External view: The XHTML view is outside the SMART Doc and is referenced by the `Href` attribute of the VIEW element.
- Encoded view: This is internal and contains a text encoding of the view, such as a base64 encoded and embedded file, specified in the _EncodingTypeDescription attribute. An encoded view may be used to maintain the privacy and confidentiality of the VIEW information.

Tagged View Implementation Process

This section describes how to create an internal, tagged VIEW. We assume that the SMART Doc template has been created and that it is in the Unpopulated state.

Step One: Create the VIEW element

The VIEW element is required and must exist. In addition, the following attributes MUST be set:

- `_ID`: This is required and identifies the VIEW. It is recommended that a unique identifier is added to the VIEW so that it may be later referenced by the tamper-evident digital signature. For more information on the tamper evident-signature, please see Chapter 4.3, Tamper Evident Signatures.
- `_TaggedIndicator`: This is required and must be set to True for tagged views
- `_MIMETypeDescription`: This required attribute defines the MIME type of the VIEW for both external and internal views. The best current practice for using various Internet media types for serving various XHTML Family documents is as follows:
- 'application/xhtml+xml' SHOULD be used for XHTML Family documents, This is used for the eNote.
- the use of 'text/html' SHOULD be limited to HTML-compatible XHTML 1.0 documents.
- 'application/xml' and 'text/xml' MAY also be used, but whenever appropriate, 'application/xhtml+xml' SHOULD be used rather than those generic XML media types.
- `Href`: This optional attribute is only required for external views. If the tagged view is in a file outside of the SMART Doc, this attribute will reference the filename of the VIEW.

- `_EncodingTypeDescription`: This optional attribute is used when the view is a text encoding of an image view, for example a TIFF document expressed as a Base64 encoded VIEW. It is not relevant to tagged views.

Step Two: Create the XHTML content

Once the VIEW element is created, the document content needs to be converted to its XHTML representation. This is equivalent to authoring the document in HTML. However, care needs to be taken to make sure that the HTML is XML compliant as XHTML and that the XHTML corresponds to the SMART Doc Specification XML standards. See Chapter 5.3 for specific XHTML requirements on tagged views.

The author of the visual presentation has the responsibility of defining where signature lines and actual signatures for various roles appear. The document template MUST provide enough information for a down-stream signing platform to apply signatures to the document. The signature information may be applied at the unpopulated, populated or signable states of the SMART Doc.

However, once the signature lines are applied and signature sections and areas have been defined, the SMART Doc is in the Signable state. For further information about the Signable State, see Chapter 2.3, Making a SMART Doc Signable.

Validate that the XHTML view displays in a browser. Note: not all browsers display XHTML content in exactly the same way. You will need consult your trading partner's requirements for acceptable browser versions/platforms.

Step Three: Create the DATA section

If you are creating a Category 1 SMART Doc, you will need to create the DATA\MAIN section. Please refer to Chapter 3 for more information on creating the DATA section.

Step Four: Link the VIEW and DATA sections

If you are creating a Category 1 SMART Doc, the data elements or attributes in the data section MUST be linked to elements in the tagged XHTML view. Linking is maintained by three components:

1) Mortgage Data expressed in the data section:

2) Mapping of the data in the data section to a unique id within the View

3) Using the unique id specified in the mapping section in the XHTML tags

The data section in a tagged SMART Doc will contain a mapping between each field in the XML data section and the visual depiction of that data in the tagged XHTML view. The link is maintained in the XHTML view by using the `<span>` or `<div>` tags and the `<span>` or `<div>` tag's `id` attribute. The value of the id attribute matches the XML data tag that it is associated with by using the <MAP> element.

The `<MAP>` element maintains the relationship between the `<DATA>` element and a tagged view in the `<VIEW>` element. It maps each variable data field in the view to a corresponding XML element or attribute in the `<DATA>` element (under the `<MAIN>` element). The `<MAP>` element contains one or more `<ARC>` or `<CONVERT>` elements.

The `<ARC>` element uses XPath expressions for mortgage information in the data section and an id for the XHTML view (`<span>` or `<div>` elements).

Please refer to Section 3 for more information on linking the DATA to the VIEW section and handling conversions.

Step Five: Validate the VIEW section

To validate the VIEW section, the SMART Doc must pass XML validation.

Checklist

- 9 Create the VIEW element and add the `_ID` attribute, set the `_TaggedIndicator` attribute to `True,` add the `_MIMETypeDescription` that defines the MIME appropriate fro your application, and set the `Href`: for external views, if your tagged view is external.
- 9 Create the XHTML representation of the document
- 9 For Category 1 SMART Docs, create the DATA section, complete with mapping the data to the view and any conversions.
- 9 For Category 1 SMART Docs, link the DATA and VIEW sections
- 9 Validate the VIEW section

XML Structures Used

`<VIEW>`

Known Issues

Although it is technically possible, there is no requirement for an external XHTML view for eNotes at this time.

Other References

See 2.2, Populating a SMART Doc, and Section 3, Linking, Data Conversions and Operators for other considerations when mapping the data and view sections.

See Section 7 for specific information regarding requirements for the data section and customizing data and data DTDs.

See Chapter 8.3, National eNote Registry and SMART Docs for detail on the language needed for the eNote.

See Chapter 10: References for all other references.

Chapter 5.2: Implementing Image Views

This chapter describes how to create image, i.e. non-tagged, views in a SMART Doc®.

Version 2.0

Revision History

Version	Date	Change
2.0	03/07/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification V 1.02

Overview

A SMART Doc VIEW is the visual representation of the document. In addition to the document narrative text, it may also present the data from the DATA section and any signatures on the document. The Specification allows for two kinds of views:

1) Tagged Views: The visual representation of the document is contained inside the SMART Doc, expressed as XHTML, i.e., it is a marked-up view

2) Image Views: Any non-textual, binary, graphical representation of the document, such as PDF, TIFF, etc.

This chapter demonstrates how image views are created. Please refer to Chapter 5.1 for a discussion of tagged views.

Pre Conditions

Document State: Any
Document Categories: 3 and 4

Post Conditions

Document State: Any
Document Categories: 3 and 4

Business Context / Scenario

The SMART Doc Specification provides a structured way of defining the visual representation of a document using XHTML tagged views. However, legacy or formatting requirements may not always allow for a tagged view. In these cases, an image view is used to host the document inside a SMART Doc container. An example of an image view would be a TIFF or PDF document contained inside a SMART Doc. The SMART Doc may or may not contain a data section. With image views it is not possible to link the data in the view section with data in a data section. Consult you trading partner's requirements for SMART Docs that are acceptable as image views.

Technical Guidance

There are two kinds of non-tagged, image views:

- Encoded view: The image view is internal and contained inside the SMART Document as a text encoded stream.
- External view: The image view is outside the SMART Doc and is referenced by the `Href` attribute of the VIEW element.

Image View Implementation Process

This section describes how to create an image VIEW. We assume that a SMART Doc template has been created and that it is in the Unpopulated state.

Step One: Create the VIEW element

The `VIEW` element is required and must exist. In addition, the following attributes can be set:

- `_ID`: This is required and identifies the VIEW
- `_TaggedIndicator`: This is required and must be set to **False** for image views
- `_MIMETypeDescription`: This required attribute defines the MIME type of the VIEW for both external and internal views. Consult the IETF RFC 2046 for acceptable MIME types.
- `Href`: This optional attribute is only required for external views
- `_EncodingTypeDescription`: This optional attribute is used for internal, encoded image views, where the view is contained inside the SMART Doc as a text-encoded stream, for example a TIFF document encoded in Base64

Here is the VIEW element for an external PDF image view:

```
<VIEW _ID="FNMA_Sample_View_3200"
_MIMETypeDescription="application/pdf"
_TaggedIndicator="False" Href="FNMA_Sample_3200.pdf">
```

See the next section for an example of defining an internal, encoded view.

Step Two: Add the VIEW content, if needed

For external image views, there is no VIEW content and this step is not necessary. The actual visual representation of the document either exists outside the SMART

Document in the file referenced by the `Href` tag in the VIEW element or internal image views are stored inside the SMART Doc in a text encoded format. Therefore, the VIEW element needs to define additional attributes:

- `_MIMETypeDescription`: This required attribute defines the MIME type of the VIEW. Consult the IETF RFC 2046 for acceptable MIME types.
- `_EncodingTypeDescription`: Specifies the text encoding format, for example Base64

Here is a sample internal image view for a PDF VIEW encoded in Base64:

```
<VIEW _ID="FNMA_Sample_View_3200"
_MIMETypeDescription="application/pdf"
_TaggedIndicator="False" _EncodingTypeDescription="Base64">
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4NCjwhRE9DVF
lQRSBTTUFSVF9E
T0NVTUVOVCBTWVNURU0gIlNNQVJUX0RPQ1VNRU5UX1ZfMV8wLmR0ZCI+DQo8U0
1BUlRfRE9DVU1F
TlQgTUlTTU9WZXJzaW9uSWRlbnRpZmllcj0iMS4wIiBQb3B1bGF0aW5nU3lzdG
VtRG9jdW1lbnRJ […]
ooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigA
ooooAKKKKAP//Z
</VIEW>
```

Step Three: Validate the VIEW section

To validate the VIEW section, the SMART Doc must pass XML validation. In addition, some sort of validation of external references should also be performed. If the View is an internal encoded view, the encoded content should be decoded to its binary form and reviewed in the appropriate application. For instance, if the image view is a PDF file, the encoded view would need to be decoded and viewed in Adobe Acrobat or another suitable tool.

Checklist

- 9 Create the VIEW element
- 9 add the `_ID` attribute
- 9 set the `_TaggedIndicator` attribute to `False`
  - 9 add the `_MIMETypeDescription that` defines the MIME type appropriate for your application
- 9 For encoded views, set the `_EncodingTypeDescription`.
- 9 If your image view is external set the `Href` for external views.
- 9 Create the image representation of the document
- 9 For Category 3 SMART Docs, create the DATA section.
- 9 Validate the VIEW section

XML Structures Used

`<VIEW>`

Known Issues

Other References

See 2.2, Populating a SMART Doc, and Section 3, Linking, Data Conversions and Operators for other considerations when mapping the data and view sections.

See Section 7 for specific information regarding requirements for the data section and customizing data and data DTDs.

See Chapter 5.4, eNote Language in the View, for detail on the language needed for the eNote.

For other references, see Chapter 10: References.

Chapter 5.3 XHTML Requirements for Views

This chapter describes XHTML requirements for the View Section

Version 2.0

Revision History

Version	Date	Change
2.0	03/07/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification V 1.02

Overview

A SMART Doc® VIEW is the visual representation of the document. In addition to the document narrative text, it may also present the data from the DATA section and any signatures on the document.

For some SMART Docs, specifically Category 1 and 2, the View section MUST be represented in XHTML (eXtensible HyperText Markup Language).

This chapter outlines XHTML requirements for tagged Views. Please refer to Chapter 5.1 and Chapter 5.2 for a discussion of implementing views and Chapters 3.1, 3.2 and 3.3 for Data Mapping and Conversions.

Pre Conditions

Document State: Any
Document Categories: 1 and 2

Post Conditions

Document State: Any

Business Context, Scenario

In the paper world, there is usually no distinction made between the content of a document and the data that is merged into the fill-out fields on a form. As the document is filled out at various points in its life cycle, the data is merged in and becomes part of the visual document. The SMART Doc Specification models this process as discrete states that define how the document presentation – its VIEW – is manipulated as it flows through its life cycle. If the document is a Category 1 or 2 SMART it is represented in a structured, XHTML format, known as a Tagged View. This chapter outlines the requirements for XHTML for the view section in a SMART Doc.

Technical Guidance

XHTML

HTML is the set of codes in a document that allows the document to be displayed on the World Wide Web. HTML 5.2 is the current version. HTML documents describe the presentation of information on a computer screen (“a web page in a browser”).

XHTML is the next logical step to HTML. An XHTML document is an HTML document and an XML document. An XHTML document may be viewed as a web page and can be processed by XML software.

XHTML documents are XML conforming. They are readily viewed, edited, and validated with standard XML tools. XHTML documents can be written to operate as well or better in existing HTML browsers as well as in new, XHTML 1.0 conforming browsers.

The SMART Doc specification uses the XHTML 1.0 recommendation for tagged views.

XHTML differs from HTML in that it requires, among other things, that tags be properly nested, empty tags be closed, non-empty tags be followed by a closing tag, tag names be in lowercase, and attribute values be surrounded by quotation marks. It is a generally a straightforward process to convert existing HTML web pages to XHTML. Any views expressed in as a Category 1 or 2 SMART Doc MUST conform to XHTML syntax.

XHTML View Creation

Static View

All XHTML views must be “static” and not rendered dynamically at any point after the documents are presented to the borrower. That is, no post processing is allowed to create the XHTML view.

The use of the <script> tag is NOT allowed.

XSL is used to dynamically generate a view of an XML document. An XSL design relies on software to apply the stylesheet to the XML document and generate a view. It was determined that reliance on outside software to generate a view did not afford the highest possible level of document integrity. The XML SMART Doc MUST not use XSL to generate the VIEW section from the data section. XSL may be used to extract the View for rendering in a browser. See the section “XHTML View Display” below.

XHTML Formatting

There are no requirements on the format of the XHTML except for the following:

- Spacing, such as extra space, between words, paragraphs, sections, etc. is not significant (see exception below)
- Spacing, such as an extra space, between special characters such as section symbols, dollar signs, etc. and the data associated with it is not significant

- Indentations are not significant (see exception below)
- Formatting, in general, is not significant (see exception below)
- **EXCEPTION**: there are only a few circumstances where state statute specifically requires adherence to certain formatting. For example, a short paragraph, which is bold, centered, and uses block indent is a good indication of a specific requirement. All statute-required formatting must be followed.

The use of Cascading Style Sheets (CSS) is acceptable so long as the CSS style sheets are included within the XHTML.

Data Mapping

For Category 1 SMART Docs, the data elements and attributes in the data section MUST be linked to elements in the tagged XHTML view. Linking is maintained by three components:

1) Mortgage Data expressed in the data section
2) Mapping of the data in the data section to a unique id within the View
3) Using the unique id specified in the mapping section in the XHTML tags

Use of <span> and <div> IDs

Use of a <span> or <div> tag with an id attribute specified is reserved for a special purpose in SMART Docs: Span or div tags with an id attribute identify values in the view section that also exist in the data section. In the SMART Doc specification, these are defined as "variable data elements". All variable data elements that appear in the View must be linked by way of an <ARC> as specified in Chapters 3.1, 3.2, and 3.3 of this implementation guide. It is not recommended to create SMART Docs that contain <span> or <div> tags with id attributes that do not have corresponding ARC elements.

XHTML View Display

For display of the <VIEW> section, it is recommended that the XHTML contents be extracted to a separate file with an "html" or "htm" extension. There are two methods to achieve this:

A) Cut and paste from the <html> tag to the </html> into a new, separate file
B) Use the following stylesheet to automatically generate the XHTML:

```
<?xml      version="1.0"      encoding="UTF-8"?>
<xsl:stylesheet                    version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:output    method="html"       version="4.0"
encoding="UTF-
8"/>
```

```
   <xsl:template match="/">
      <xsl:copy-of select="//VIEW/*"/>
   </xsl:template>
   </xsl:stylesheet>
```

Note: There may be non XHTML compliant tags in the View. These tags are extensions to XHTML and are used for signature information. Some examples of these tags include <SIGNATURE_AREA>, <SIGNATURE_TEXT> and <SIGNATURE_TARGET>.

Checklist

- 9 XHTML is static. No scripting or XSL or other techniques are used to generate portions of or the entire presentation on the computer screen.
- 9 The XHTML in the view must be valid XHTML
- 9 For Category 1 documents, the data elements and attributes in the data section MUST be linked to elements in the tagged XHTML view
- 9 The <span> or <div> tags id attributes MUST have corresponding ARC elements; that is, there must not be an id attribute without a corresponding data element.
- 9 The XHTML must be viewable in a browser

XML Structures Used

This section provides a listing of the XML structures used, in the following format:

```
<VIEW>
<html>
```

Known Issues

Some XHTML tags are not valid per the W3C standard. See the modified XHTML DTD.
Not all browsers will display XHTML in exactly the same way. Consult your trading partner's requirements for supported browser versions/platforms.

Other References

See 2.2, Populating a SMART Doc, and Section 3, Linking, Data Conversions and Operators for other considerations when mapping the data and view sections.

See Section 7 for specific information regarding requirements for the data section and customizing data and data DTDs.

See Chapter 5.4, eNote Language in the View, for detail on the language needed for the eNote.

For other references, see Chapter 10: References.

Chapter 5.4: eNote Language in the View

This chapter describes the special eNote Language requirements for SMART Docs®

Version

2.0

Revision History

Version	Date	Change
2.0	02/15/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification V 1.02

Overview

The MERS® eRegistry is a compliance vehicle to satisfy certain requirements of the Uniform Electronic Transactions Act (UETA) and the federal Electronic Signatures in Global and National Commerce Act (E-SIGN) which provide an owner of an eNote (the Controller) with the status of a "Holder in Due Course." An eNote issued and transferred in compliance with Section 16 of UETA or Title II of E-SIGN is called a Transferable Record (TR). Specifically, Section 16 of UETA and Title II of E-SIGN require that the party in control of the Authoritative Copy (AC) of the TR at any given point in the life cycle of an eNote be readily identified. The MERS eRegistry is owned and operated by MERSCORP, Holdings, Inc., a Deleware Corporation.

This document describes the requirements for special language in the eNote that refers to the eRegistry that must be present in the `<VIEW>` section of a SMART Doc.

Pre Conditions

Document States: Unpopulated
Document Categories: All

Post Conditions

Document State: Unpopulated

Business Context

The eRegistry Concept

As described above, the MERS® eRegistry has been designed to establish ownership of eNotes in compliance with eSignature laws. In order for a SMART Doc eNote to become a Transferable Record under the law, the eNote must contain special language, which we are calling the eNote Clause, that attests to the borrower's intention to issue the document electronically and instructs any holder of the eNote to look to the registry named in the eNote for definitive information on the controller of that eNote.

Scenario

The borrower is sitting at the closing table and is ready to sign the eNote. eSignature laws require that the borrower not only sign to attest to the content of the eNote but also attest to the fact that they agree to issue an eNote. This clause exists in the view section of the SMART Doc, which in turn is displayed on the computer screen at the time of signing. The eNote language informs any entity that needs to know that the definitive source of information on its controller can be found in the named registry.

Technical Guidance

eNote Headder

The Uniform eNote has a new line below the word "Note" that serves as an additional reminder that the eNote is designed only for electronic signatures. The word "Note" should be replaced by:

Note
(For Electronic Signature)

eNote Footer

The footer for an eNote is similar to the document footer that appears on the standard mortgage and deed of trust notes, except for the identification of the document as an eNote. The footer for the Uniform Multistate Fixed Rate eNote is shown below:

MULTISTATE FIXED RATE **e**NOTE–Single Family–Fannie Mae/Freddie Mac Uniform Instrument   Form 3200**e**   5/05

The eNote Clause

The eNote Clause should be included in all eNotes as the last numbered section of the eNote and having its own number. For instance, the eNote
Clause should be numbered 11 for the Multistate Fixed Rate Note (Form 3200), where the last section in the paper note is Section 10. Likewise for other notes, the eNote Clause should be placed as the last section and be numbered accordingly.

The following language represents the eNote Clause for conforming Uniform Instrument Fannie Mae/Freddie Mac eNotes:

ISSUANCE OF TRANSFERABLE RECORD; IDENTIFICATION OF NOTE HOLDER; CONVERSION FROM ELECTRONIC NOTE TO PAPER-BASED NOTE
(A) I expressly state that I have signed this electronically created Note (the "Electronic Note") using an Electronic Signature. By doing this, I am indicating that I agree to the terms of this Electronic Note. I also agree that this Electronic Note may be Authenticated, Stored and Transmitted by Electronic Means (as defined in Section

11(F)), and will be valid for all legal purposes, as set forth in the Uniform Electronic Transactions Act, as enacted in the jurisdiction where the Property is located ("UETA"), the Electronic Signatures in Global and National Commerce Act ("E-SIGN"), or both, as applicable. In addition, I agree that this Electronic Note will be an effective, enforceable and valid Transferable Record (as defined in Section 11(F)) and may be created, authenticated, stored, transmitted and transferred in a manner consistent with and permitted by the Transferable Records sections of UETA or E-SIGN.

(B) Except as indicated in Sections 11 (D) and (E) below, the identity of the Note Holder and any person to whom this Electronic Note is later transferred will be recorded in a registry maintained by MERSCORP Holdings, Inc., a Delaware Corporation, or in another registry to which the records are later transferred (the "Note Holder Registry"). The authoritative copy of this Electronic Note will be the copy identified by the Note Holder after loan closing but prior to registration in the Note Holder Registry. If this Electronic Note has been registered in the Note Holder Registry, then the authoritative copy will be the copy identified by the Note Holder of record in the Note Holder Registry or the Loan Servicer (as defined in the Security Instrument) acting at the direction of the Note Holder, as the authoritative copy. The current identity of the Note Holder and the location of the authoritative copy, as reflected in the Note Holder Registry, will be available from the Note Holder or Loan Servicer, as applicable. The only copy of this Electronic Note that is the authoritative copy is the copy that is within the control of the person identified as the Note Holder in the Note Holder Registry (or that person's designee). No other copy of this Electronic Note may be the authoritative copy.

(C) If Section 11 (B) fails to identify a Note Holder Registry, the Note Holder (which includes any person to whom this Electronic Note is later transferred) will be established by, and identified in accordance with, the systems and processes of the electronic storage system on which this Electronic Note is stored.

(D) I expressly agree that the Note Holder and any person to whom this Electronic Note is later transferred shall have the right to convert this Electronic Note at any time into a paper-based Note (the "Paper-Based Note"). In the event this Electronic Note is converted into a Paper-Based Note, I further expressly agree that: (i) the Paper-Based Note will be an effective, enforceable and valid negotiable instrument governed by the applicable provisions of the Uniform Commercial Code in effect in the jurisdiction where the Property is located; (ii) my signing of this Electronic Note will be deemed issuance and delivery of the Paper-Based Note; (iii) I intend that the printing of the representation of my Electronic Signature upon the Paper-Based Note from the system in which the Electronic Note is stored will be my original signature on the Paper-Based Note and will serve to indicate my present intention to authenticate the Paper-Based Note; (iv) the Paper-Based Note will be a valid original writing for all legal purposes; and (v) upon conversion to a Paper-Based Note, my obligations in the Electronic Note shall automatically transfer to and be contained in the Paper-Based Note, and I intend to be bound by such obligations.

(E) Any conversion of this Electronic Note to a Paper-Based Note will be made using processes and methods that ensure that: (i) the information and signatures on the face of the Paper-Based Note are a complete and accurate reproduction of those reflected on the face of this Electronic Note (whether originally handwritten or manifested in other symbolic form); (ii) the Note Holder of this Electronic Note at the time of such

conversion has maintained control and possession of the Paper-Based Note; (iii) this Electronic Note can no longer be transferred to a new Note Holder; and (iv) the Note Holder Registry (as defined above), or any system or process identified in Section 11 (C) above, shows that this Electronic Note has been converted to a Paper-Based Note, and delivered to the then-current Note Holder.
(F) The following terms and phrases are defined as follows: (i) “Authenticated, Stored and Transmitted by Electronic Means” means that this Electronic Note will be identified as the Note that I signed, saved, and sent using electrical, digital, wireless, or similar technology; (ii) “Electronic Record” means a record created, generated, sent, communicated, received, or stored by electronic means; (iii) “Electronic Signature” means an electronic symbol or process attached to or logically associated with a record and executed or adopted by a person with the intent to sign a record; (iv) “Record” means information that is inscribed on a tangible medium or that is stored in an electronic or other medium and is retrievable in perceivable form; and (v) “Transferable Record” means an electronic record that: (a) would be a note under Article 3 of the Uniform Commercial Code if the electronic record were in writing and (b) I, as the issuer, have agreed is a Transferable Record.”

Loan Originator and Broker Names and NMLSR IDs

At the bottom of the note, following the borrower signatures, The Loan Originator and Broker (if exists) names and NMLSR Identifiers should be provided:

Loan Originator (Individual): **George Bailey** NMLSR ID Number: **123456**
Loan Originator (Company): **Bailey Building and Loan** NMLSR ID Number: **234567**
Loan Broker (Individual): **Benjamin Broker** NMLSR ID Number: **345678**
Loan Broker (Company): **XYZ Financial Services** NMLSR ID Number: **456789**

For the Data Section, an extended DTD containing the following must be used to support this data:

```
<!ELEMENT LOAN_ORIGINATOR EMPTY>
<!ATTLIST LOAN_ORIGINATOR
_NationwideMortgageLicensingSystemAssignedIdentifier CDATA #REQUIRED
_UnparsedName CDATA #REQUIRED NonPersonEntityIndicator (Y | N) #REQUIRED>
<!ELEMENT PARTY EMPTY>
<!ATTLIST PARTY _Type (ClosingAgent | Lender | LoanOfficer | MortgageBroker |
RealEstateAgentListing | RealEstateAgentSelling) #REQUIRED
NationwideMortgageLicensingSystemAssignedIdentifier CDATA #REQUIRED
_UnparsedName CDATA #REQUIRED NonPersonEntityIndicator (Y | N) #REQUIRED>
```

In the XML Data section, the data will appear as follows:

```
<CUSTOM>
  <LOAN_ORIGINATOR _UnparsedName="George Bailey"
_NationwideMortgageLicensingSystemAssignedIdentifier= "123456"
NonPersonEntityIndicator="N"/>
    <LOAN_ORIGINATOR _UnparsedName="Bailey Building and Loan"
_NationwideMortgageLicensingSystemAssignedIdentifier="234567"
NonPersonEntityIndicator="Y"/>
     <PARTY _UnparsedName="Benjamin Broker"
_NationwideMortgageLicensingSystemAssignedIdentifier="345678"
NonPersonEntityIndicator="N" _Type=”MortgageBroker”/>
     <PARTY _UnparsedName="XYZ Financial Services"
_NationwideMortgageLicensingSystemAssignedIdentifier="456789"
NonPersonEntityIndicator="Y" _Type=”MortgageBroker”/>
 </CUSTOM>
```

See the 1_02 SMARTDoc Format.xlsx and the sample eNotes for further information about this data.

Data that is not needed to support the eNote should not be included in the Data section or the View, especially Personally Identifiable Information such as Borrowers' Social Security Numbers and telephone numbers. See MERS requirements for instructions for providing additional data needed for linking in the MERS Repository.

Checklist

√ Check for the eNote language clause from your specific investor delivery guidelines

√ Ensure that the language has been added to the <VIEW> section as the last section.

XML Structures Used

<VIEW>

Known Issues

None.

Other References

See Chapter 10: References.

Chapter 6.1: SMART Doc® Categories

Version 2.0

Revision History

Version	Date	Change
2.0	03/07/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification v1.02

Overview

This chapter describes the characteristics of the various SMART Doc Categories. Within each Category's description are examples of probable uses for the various categories. Document examples of each of these categories can be found in the I-Guide Appendices.

Business Context

Moving to more fully electronic documents benefits the industry through the reduction of the time and resources required to create and handle documents, and by increasing the speed and efficiency of managing packages (by moving data in place of paper). The greatest value to the industry will come when the majority of new loans utilize SMART Docs, and when the majority of documents (or at least the most important documents), are created as Category One or Category Two (see below). However, there appears to be no universal business reason that would require every document to be one of these two categories. In addition, it is unlikely that all business partners will migrate to these two document types at the same pace, necessitating the existence of other alternatives that offer some SMART functionality.

Category Descriptions

Category One	HEADER, DATA, XHTML VIEW	This category is designed to fully exploit all capabilities of the specification. Example: The eNote is a prime candidate for deployment as a Category One SMART Doc. Category One Requirements: o HEADER is fully implemented. o DATA section is complete. o View is in XHTML format (tagged view). o ARCs are implemented for every data point in the VIEW. o When appropriate, the `<SIGNATURES>` section is in place and includes a final tamperseal digital signature, with references to all sections.
Category Two	HEADER, XHTML VIEW	This category describes a document that can fully exploit all the capabilities of the specification as defined for a Category One document, except that it has no data. This implementation eliminates the DATA section, including the MAP section. Since no data section exists in these documents, the data in a Category Two view cannot be validated or easily extracted. An example of the use of a Category Two document is one in which the document is essential to the mortgage package, like the mortgage application, Good Faith Estimate or Truth In Lending disclosure, and which would meet trading partner requirements. Category Two Requirements: o HEADER is fully implemented. o VIEW is in XHTML format (tagged view). o When appropriate, the SIGNATURE section is in place and includes a final tamperseal digital signature, with references to all sections.

	Category Three	HEADER, DATA, Image VIEW	This category is designed to describe documents that do not have a tagged VIEW in XHTML. The VIEW can be any image, most commonly a PDF or other form of image such as a JPEG or TIFF. Examples of the use of a Category Three document could be an appraisal; some forms of insurance; and other business partner documents. The data cannot be validated against the view representation. The data could be extracted and utilized by parties, and the view could also be utilized as a copy (electronic or paper) as is done today. Category Three Requirements: o HEADER is fully implemented. o DATA section includes all data that is represented in the VIEW. o Non-tagged VIEW. o MAP section is not required. o It is understood that the DATA and VIEW cannot be systematically compared. o Guarantees about the data consistency are made at the discretion of the trading partners. o When appropriate the SIGNATURE section is in place and includes a final tamperseal digital signature, with references to all internal sections and external files.

Category Four	HEADER, Image VIEW	This category describes documents that contain a non-tagged VIEW but do not contain a DATA section. These documents have limitations in terms of their "SMART" quality since they cannot pass data downstream. However, the fully implemented HEADER enables some exchange of information. An example of the use of a Category Four document is the case of a W2 in which a system can read what the document is ("W-2"), and other limited information like the audit trail. These documents have been referred to as "containerized documents." Another example could be the property survey, where the most important piece of the document is the image as it was produced by a Surveyor. Category Four Requirements: o HEADER is fully implemented. o Non-tagged VIEW. o No DATA section. o The SIGNATURES section, when appropriate, is in an external file. The signature file includes a final tamperseal digital signature, with references to all internal sections and external files.
Category Five	HEADER, DATA	This category is designed for documents that do not have a VIEW. These documents are used primarily to pass other data (extraneous to the loan documents) that will be needed to complete a process or transaction. A Category Five document could be used to pass extra data, such as ARM plan information for servicing setup. Category Five Requirements: o HEADER is fully implemented. o DATA section without a MAP section. o The SIGNATURES section, when appropriate, contains, or consists only of a final tamperseal digital signature, with references to all internal sections.

Other References

Example documents for each of these categories can be found in the distribution package for this Implementation Guide.

See Chapter 10: References for references to other documents.

Chapter 7.1: Setting the Document Type

This chapter describes the document type setting, and how to specify the document type in the header of the SMART Doc®.

Version

2.0

Revision History

Version	Date	Change
2.0	03/07/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification V 1.02

Overview

This chapter discusses the document type setting, which indicates what type of mortgage document the SMART Doc contains.

Pre Conditions

Document State: Any
Document Categories: Any

Post Conditions

Document State: Any
Document Categories: Any

Business Context

SMART Docs use the MISMO® Closing DTD for the data section. The MISMO Closing DTD contains data containers for a large number of eMortgage documents. There are no individual DTDs for each eMortgage document type. That is, a DTD that *only* contains the data specific to an eNote does not exist. With SMART Docs, it is not possible to refer to the DTD to determine its document type.

The use of the _Type attribute in the Header element allows a SMART Doc to indicate which document type it contains, such as Note, Addendum, Security Instrument, etc. During initial processing of a set of SMART Docs, back-end systems can read the _Type attribute for any sorting or pre-processing tasks that are required.

Scenario

A closing agent has sent a set of eMortgage documents to you. Your application does not archive the closing instructions included in the eMortgage package. For each document in the eMortgage package, your application must check type of document in the header and process each document type appropriately.

Technical Guidance

The required `<HEADER>` element contains information (metadata) about the document, such as its document type, its state and access rights. The `<HEADER>` element contains the `<DOCUMENT_INFORMATION>` element, which has a `_Type` attribute. The `_Type` attribute should be assigned one of the following possible enumerated list values:

```
Note Addendum
SecurityInstrument
Assignment
Rider
Disclosure
TIL
Itemization
RightToCancel
HUD-1
RESPA
ClosingInstruction
PaymentLetter
BorrowerAffidavit
Insurance
Lender
Investor
LoanModification
DeedOfTrust
Other
```

If the "`Other`" is chosen as the document type you must provide a document type in the attribute `_TypeOtherDescription.` The correct setting for an eNote is:

```
<DOCUMENT_INFORMATION _Type="Note"/>

<DOCUMENT_INFORMATION
_TypeOtherDescription="PowerOfAttorney"/>
```

Checklist

- 9 Check specific investor requirements about document types
- 9 Set the _Type attribute in the <HEADER> to the correct document type

XML Structures Used	`<DOCUMENT_INFORMATION>`
Known Issues	There are several document types that are similar; for instance, the "`DeedOfTrust`", "Mortgage" and the "`SecurityInstrument`" document types may be considered to be equivalent in some systems. Consult individual investor requirements.
Other References	See Chapter 7.5 for specific information regarding requirements for the data section. For other references, see Chapter 10: References.

Chapter 7.2: Adding CUSTOM Data

This chapter provides instruction on how to add custom data to a SMART Doc®.

Version 2.0

Revision History

Version	Date	Change
2.0	03/07/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification V1.02

Overview

There is often a business need to pass data within a SMART Doc that is not specifically defined in the MISMO® standard and/or not specifically defined in a given MISMO DTD. In order for SMART Docs to contain additional data that is not defined as a part of SMART Doc framework DTD or the SMART Doc Data DTD, an extended and customized DTD must be created. There are three methods by which the SMART Doc may be extended. This section describes how to create and add custom information tailored to specific processes or application requirements. If you are interested in using a DTD that is not an extension of the MISMO Data DTDs see then next chapter, Chapter 7.2, Using other Data DTDs.

Pre Conditions

Document State: All
Document Categories: All

Post Conditions

Document State: N/A

Business Context

The SMART Doc specification defines a standard for the representation of mortgage documents in an electronic format. However, it is possible that during implementation a situation may arise for which extra data is required that is not included within the specification.

The following is a list of the types of documents that can be constructed using the SMART Doc building blocks:

LEGAL DOCUMENTS
Promissory Notes/--Applicable Addendums and Riders
Security Instruments/--Applicable Riders
Assignments

FEDERAL TRUTH IN LENDING DOCUMENTS
TIL
Itemization of Amount Financed
Right to Cancel

RESPA DOCUMENTS
Notice of Assignment, Sale, or Transfer of Servicing Rights
RESPA Servicing Disclosure
Initial Escrow Account Statement
HUD-1

HOEPA / PMI DOCUMENTS
PMI Disclosures--Fixed, ARM, Amortization Schedule

ECOA DOCUMENTS
Fair Lending Notice
Right to Copy of Appraisal

BROKER / LENDER / INVESTOR SPECIFIC DOCUMENTS
Closing Instructions
Payment Letter
Affidavit of Occupancy
Signature/Name Affidavit
E/O Compliance Agreement
Borrower's Certification and Authorization
Hazard Insurance Authorization, Requirements and Disclosure
Tax Information and Collection
Notice of Flood Hazard Area
Flood Insurance Notification/Authorization
Request for Copy of Tax Form (4506)
Request for Tax ID & Certification (W9)

If there is a need to include data within the SMART Doc that is not defined within the specification, the DTD may be customized to include newly defined, proprietary or application specific data. The SMART Doc specification allows for three areas to be customized. The manner in which the SMART Doc is customized is dependent on implementation specific requirements.

1) If meta information or information about the SMART Doc as a whole is to be included, then the <HEADER> element should be customized. For instance, if there is requirement to include the browser version that was used to present the SMART Doc to the signer, this information may be included within the header.

2) If there is additional data to be included or extended from the SMART Doc Data set (the MISMO Closing DTD or the PRIA DTD), the method for extending the data should follow the MISMO recommendation and use the Generic Structured Extension, as specified in the Engineering Guidelines.

3) If specific information for the SMART Doc is to be included, the Data section is customized. For instance, an application may require the addition of the database tables that were used as input data to the SMART Doc. In this use case, the DATA section should be customized. Or, an implementation may need to add in conversions that are not included in the SMART Doc specification. In this case, the `<CUSTOM>` section may include the conversions so that the format in the VIEW is preserved.

Implementation requirements will dictate which method of customization is needed and how it should be done.

Technical Guidance

Adding CUSTOM data to the Header

The <META_DATA> element is functionally equivalent to the HTML <meta> tag. It is used to add name/value pairs of information to describe the document. A developer may add customized metadata to a document using this repeatable element. The <META_DATA> element contains the following attributes:

- The _Name attribute is a descriptive name of a characteristic describing the document such as “Browser Version used by Signers”.
- The _Value attribute carries the value associated with the _Name attribute such as “Internet Explorer 5.5 SP2”.

Step One: Add the <META_DATA> element to the header

Add the <META_DATA> elements that are required by the implementation with the attributes _Name and _Value. Note that the <META_DATA> element must come after <DOCUMENTATION_INFORMATION> and before the <SIGNATURE_MODEL> element.

```
<HEADER _ID="FNMA_Sample_Header_3200">
        <DOCUMENT_INFORMATION>
        […]
        </DOCUMENT_INFORMATION>
        <META_DATA _Name="" _Value=""></META_DATA>
        <SIGNATURE_MODEL>
        […]
        </SIGNATURE_MODEL>

</HEADER>
```

Step Two: Add the values to _Name and _Value

```
<HEADER _ID="FNMA_Sample_Header_3200">
        <DOCUMENT_INFORMATION>
        […]
        </DOCUMENT_INFORMATION>
        <META_DATA _Name="Browser Version used by Signer 1" _Value="Internet
        Explorer 5.5 SP2"></META_DATA>
        <META_DATA _Name="Browser Version used by Signer 2"
        _Value="Netscape 7.1"></META_DATA>
        <SIGNATURE_MODEL>
        […]
        </SIGNATURE_MODEL>
</HEADER>
```

Extending Standard Data within the SMART Doc

In order for SMART Docs to include additional data that is not defined as a part of the standard SMART Doc Data DTD, you MUST employ the MISMO Architecture Working Group’s endorsed extension mechanism. This means you must edit and add your custom data to the MISMO standard DTD.

The MISMO extension mechanism provides a way to pass additional arbitrary data in a structured fashion that facilitates and encourages reuse of the extended data.

Extensions are composed of an EXTENSION container that holds one or more EXTENSION_SECTION containers. Each EXTENSION_SECTION container should be associated with a specific organization and a particular extension section type. The organization, type, and version are identified by XML attributes in the EXTENSION_SECTION container. If each EXTENSION_SECTION container is identified consistently, it will allow other applications to "recognize" common extensions.

The EXTENSION_SECTION container also allows the declaration of a default namespace. As of this writing, MISMO has not issued a definitive statement on the use of namespaces.

The actual data is contained in the EXTENSION_SECTION_DATA element.

If we wanted to extend _RESIDENCE container to include a parsed street address, we would perform the following steps:

Step 1: Add the EXTENSION definition to your customized DTD:

In this scenario, we would extend the SMART_DCOUMENT_Closing_V_2_3.dtd by adding the MISMO approved extension container:

```
<!-- ======================================== -->
<!--   Container: EXTENSION                   -->
<!--========================================
    Description: Common container to hold other
    Extension Section   containers
  ========================================= -->

<!ELEMENT EXTENSION (EXTENSION_SECTION*) > <!ATTLIST
EXTENSION ExtensionID ID  #IMPLIED>

<!--========================================= -->
<!--   Container: EXTENSION_SECTION                -->
<!--========================================
    Description: Contain organization specific
    extended data elements
============================================== -->

<!ELEMENT EXTENSION_SECTION
(CONTACT_DETAIL*,EXTENSION_SECTION_DATA?) >
<!ATTLIST EXTENSION_SECTION ExtensionSectionID ID  #IMPLIED>
<!ATTLIST EXTENSION_SECTION
         ExtensionSectionOrganizationName CDATA  #IMPLIED>
<!ATTLIST EXTENSION_SECTION
         ExtensionSectionTypeDescription CDATA  #IMPLIED>
<!ATTLIST EXTENSION_SECTION
         ExtensionSectionVersion CDATA  #IMPLIED>
<!ATTLIST EXTENSION_SECTION xmlns CDATA  #IMPLIED>
```

```
<!-- ========================================= -->
<!--  Container: EXTENSION_SECTION_DATA             -->
<!-- ========================================= -->
<!ELEMENT EXTENSION_SECTION_DATA ANY >
<!ATTLIST EXTENSION_SECTION_DATA ExtensionSectionDataID ID
#IMPLIED>
```

Note that the EXTENTION_SECTION may contain any element defined in the DTD:

```
<!ELEMENT EXTENSION_SECTION_DATA ANY >
<!ELEMENT CONTACT_DETAIL (CONTACT_POINT*)>
<!ATTLIST CONTACT_DETAIL _Name CDATA #IMPLIED>
<!ELEMENT CONTACT_POINT EMPTY>
<!ATTLIST CONTACT_POINT _RoleType (Home |
Mobile |
Work) #IMPLIED
<!ATTLIST CONTACT_POINT _Type (Email |
Fax |
Other |
Phone) #IMPLIED
<!ATTLIST CONTACT_POINT _OtherTypeDescription CDATA #IMPLIED
<!ATTLIST CONTACT_POINT _Value CDATA #IMPLIED
<!ATTLIST CONTACT_POINT _PreferenceIndicator (Y | N) #IMPLIED>
```

Step 2: Extend the _RESIDENCE container to include EXTENSION

Change
```
<!ELEMENT _RESIDENCE EMPTY>
```
To

```
<!ELEMENT _RESIDENCE (EXTENSION?)>
```

Step 3: Add the customized data container to the DTD with the 3 letter prefix appropriate for your company or organization:

```
<ELEMENT ABC_PARSED_STREET_ADDRESS EMPTY>
<!ATTLIST ABC_PARSED_STREET_ADDRESS
      _StreetName CDATA #IMPLIED
      _DirectionPrefix CDATA #IMPLIED
      _DirectionSuffix CDATA #IMPLIED
      _StreetSuffix CDATA #IMPLIED
      _StreetType CDATA #IMPLIED
      _HouseNumber CDATA #IMPLIED
      _ApartmentOrUnit CDATA #IMPLIED
      _RuralRoute CDATA #IMPLIED
```

```
>
```

Step 5: Create the XML:

Step 5a: Create the Standard XML Container The _RESIDENCE container still has the original data:

```
<_RESIDENCE _StreetAddress="123 B Main St" _City="Boston"
_State="MA" _PostalCode="02100" _Country="Text"
BorrowerResidencyBasisType="Own"
BorrowerResidencyDurationYears="4"
BorrowerResidencyType="Current">
```

Step 5b: Add information about the extension

The EXTENSION_SECTION container is now allowed inside the <_RESIDENCE container. Information about the extension is contained at the top level:

```
<EXTENSION_SECTION ExtensionSectionID="ExtSect00001"
ExtensionSectionOrganizationName="ABC"
ExtensionSectionTypeDescription="Residence Parsed Street
Address" ExtensionSectionVersion="1.0">
```

Step 5c: Add the contact information

Information regarding a contact about the extension may be provided. It is not required:

```
<CONTACT_DETAIL _Name="Rachael Sokolowski">
 <CONTACT_POINT _PreferenceIndicator="Y" _RoleType="Work"
_Type="Phone" _Value="5555551223"/>
 <CONTACT_POINT _PreferenceIndicator="N" _RoleType="Work"
_Type="Email" _Value="rsokolowski@magnoliatech.com"/>
</CONTACT_DETAIL>
```

Step 5d: Add the customized data:

```
<EXTENSION_SECTION_DATA>
    <ABC_PARSED_STREET_ADDRESS _HouseNumber="123"
_ApartmentOrUnit="B" _StreetName="Main" _StreetSuffix="St" />
   </EXTENSION_SECTION_DATA>
```

The above example shows a simple extension that adds a parsed street address to the _RESIDENCE container. There is no limitation on the complexity of structure that may be added provided that one adds the definitions of the extension ELEMENTS to the DTD file. The ANY content model allows for any content to be included; however, the elements must be defined in the DTD for validation.

The following extension will be needed to support adding Broker and Originator names and NMLS IDs to the eNote:

```
<!ELEMENT LOAN_ORIGINATOR EMPTY>
<!ATTLIST LOAN_ORIGINATOR
```

```
_NationwideMortgageLicensingSystemAssignedIdentifier CDATA #REQUIRED
_UnparsedName CDATA #REQUIRED NonPersonEntityIndicator (Y | N) #REQUIRED>
<!ELEMENT PARTY EMPTY>
<!ATTLIST PARTY _Type (ClosingAgent | Lender | LoanOfficer | MortgageBroker |
RealEstateAgentListing | RealEstateAgentSelling) #REQUIRED
NationwideMortgageLicensingSystemAssignedIdentifier CDATA #REQUIRED
_UnparsedName CDATA #REQUIRED NonPersonEntityIndicator (Y | N) #REQUIRED>
```

MISMO Extension Method For Extending Enumerated Lists

The MISMO Engineering Guidelines do not explicitly state a method for extending enumerated attribute lists. However, there is an approved method for extending attributes that contain enumerated lists without an “Other” and Other Type Description attribute.
MISMO will accept extensions of attribute lists as long as the extended list values contain the three-character company identifier. For instance, to extend LoanDocumentType within the Loan Features container the following is acceptable:

```
<!ATTLIST LOAN_FEATURES
       LoanDocumentationType (Alternative | FullDocumentation |
NoDepositVerification |
NoDepositVerificationEmploymentVerificationOrIncomeVerificatio
n | NoDocumentation |
NoEmploymentVerificationOrIncomeVerification | Reduced |
StreamlineRefinance | ABC_NewAttribute) #IMPLIED
>
```

Adding CUSTOM data for the SMART Doc

The SMART Doc has the capability to define custom data elements that are outside the SMART Doc DTDs. The majority of the time, this will be left empty. The CUSTOM section is only for extensions to the SMART Doc Framework and not for extensions to data. If you need a data container that does not already exist, you must extend the SMART Doc according to the MISMO Engineering Guidelines described above. Some implementations may require including a stylesheet or using a method to convert data in the view that is not represented in the specification. In these situations, you may use the CUSTOM section. Please note that if you use the `<CUSTOM>` element, anyone using your SMART Doc will have to support it.

The `<CUSTOM>` element provides a free-form container that allows developers and implementers to add/extend the SMART Doc DTDs with information specific to particular implementations or applications. It is intended that the tags contained within the `<CUSTOM>` element are not validated and can be used for any implementation-specific purpose. The `<CUSTOM>` element may contain text, a CDATA section or implementation specific tags.

There is a mechanism in the DTD to allow the author of a document instance to extend the definition of the <CUSTOM> element by extending the document type definition. Defining an entity of “null” content with in the content model of the `<CUSTOM>` element accomplishes this. This “null content” may be replaced later in the document instance:

```
      <!ENTITY % CUSTOM.ANY "">
```

And using the null content in the content model of `<CUSTOM>:`

```
      <!ELEMENT CUSTOM (#PCDATA %CUSTOM.ANY;)*>
```

The following XML document fragment shows a sample implementation of the <CUSTOM> element:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SMART_DOCUMENT SYSTEM "SMART_DOCUMENT_V_1_0.dtd" [

      <!ENTITY % CUSTOM.ANY "|
ImplementationSpecificConversion">
      <!ELEMENT ImplementationSpecificConversion (#PCDATA)>

]>

<SMART_DOCUMENT MISMOVersionIdentifier="1.0"
PopulatingSystemDocumentIdentifier="FNMA_Sample_3200">
      […]
      <DATA _ID="FNMA_Sample_Data_3200">
            <MAIN>
          […]
           </MAIN>
      <CUSTOM>
            <ImplementationSpecificConversion>
            ZZZZZZZ-ZZ
            </ImplementationSpecificConversion>
            </CUSTOM>
      </DATA>
```

Note: SMART Docs with the <CUSTOM> DTD extension will not render correctly in some HTML browsers. HTML browsers are not capable of understanding the DTD extensions. XHTML views that are extracted from the XML file and saved as an HTML file will display correctly.

Additionally, if any text is included in the <CUSTOM> tag, the text will render in the browser.

Alternatively, the DTD may also be extended for the specific tags allowed in the CUSTOM section.

Checklist

- Determine the type of customization required: header, data or SMART Doc
- For the HEADER, add as many <META> tags as is required
- For ANY data customization, use the MISMO Architecture Working Group's endorsed extension mechanism and the <EXTENSION> container.
- For SMART Doc custom elements and attributes that are implementation specific, use the <CUSTOM> container

XML Structures Used

<HEADER> <META>
<DATA> (<MAIN>, <CUSTOM>)
<EXTENSION>
Whether a specific XML element is used or not depends on the type of customization

Known Issues

With the exception of customizations within the header, any extensions must to be included in the edited versions of the standard DTDs. Your trading partners may or may not choose to process these DTD extensions.

Other References

MISMO Engineering Guidelines

See Chapter 7.3 for methods to employ when you wish to use a non-MISMO Data DTD. See Chapter 7.4 for specific information regarding requirements for the data sections of eNotes.

For other references, see Chapter 10: References.

Chapter 7.3: Using Other Data DTDs

This chapter provides instruction on how to use DTDs other than the SMART Doc® Data DTDs

Version 2.0

Revision History

Version	Date	Change
2.0	02/15/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification V1.02

Overview

There may be a business need to use data within a SMART Doc that is not specifically defined in the MISMO standard and/or not specifically defined in a given MISMO DTD. This section describes how to include DTDs for the DATA section tailored to specific processes or application requirements.

Pre Conditions

Document State: All
Document Categories: 1, 3, 5

Post Conditions

Document State: All

Document Categories: 1, 3, 5

Business Context

The SMART Doc specification defines a standard for the representation of mortgage documents in an electronic format. It is possible to use the SMART Doc Framework for any type of electronic document. For instance, it is possible to use the SMART Doc framework for any type of loan application, not just a mortgage application.

The DTDs:

- Define the structure of the document
- Determine the names of the tags and the tag attributes
- Do not define values for the data fields (with the exception of standardized enumerated lists and fixed values, such as the MISMO version identifier)
- Do not contain the narrative text found within the VIEW

The SMART Doc Specification and DTDs can be thought of as a set of building blocks that can be used to construct electronic documents. There are four main blocks:

1) Information about the document as a whole or the header <HEADER>
2) Tags for data found within the narrative document in a convenient and easily accessible section <DATA>
3) The view or computer screen display of the original document <VIEW>
4) Electronic and tamper seal signatures

The blocks define the names of the tagged fields within the SMART Doc and the structure. Implementers of the SMART Doc Specification use these blocks to construct individual eMortgages and to populate the data fields and the narrative text within the view appropriate for that specific document. An implementation of an electronic Note would have a different set of data fields and view than an electronic URLA.

The Specification has been designed to allow for any type of document to be SMART. The current specification uses the data fields defined by the MISMO Closing workgroup and PRIA (for recorded documents). If there is a need to use the SMART Doc Framework with a newly defined, proprietary or application specific DTD, the SMART Doc Data DTDs must be edited. The new Data DTD must be referenced from the SMART Doc Data DTD. This chapter explains how to use a non-MISMO data DTD.

Specific implementation requirements will dictate when a non-MISMO data DTD should be used.

Technical Guidance

The Structure of the DTDs

Framework DTDs

The SMART Doc framework DTDs define the structure of SMART Documents and include other DTDs. The <HEADER>, the <AUDIT_TRAIL> and elements used for electronic signatures are defined by the framework DTD. The framework includes the XHTML DTDs for the VIEW section and the W3C XML Digital Signature DTD for digital signatures.

Data DTDs

The SMART Doc DATA DTD defines the structure and the elements that may appear under the <MAIN> element. The SMART_DOCUMENT_Data_V_1_0 is the top level Data DTD that includes all referenced data DTDs. It is referenced by the SMART Doc DTD. The following DTDs are referenced by the SMART DOCUMENT DATA DTD:

- **SMART_DOCUMENT_V_1_02_Closing_V_2_3:** this DTD is a modified version of the MISMO Closing Workgroup 2.3 DTD. The DTD has been modified to exclude containers that are shared with the PRIA DTD (see below)

- **SMART_DOCUMENT_PRIA_V_1_2_RC_2_0** The PRIA DTD defines data for recorded documents and is the work of the Property Records Industry Association (PRIA). The DTD has been modified to exclude containers that are shared with the MISMO Closing DTD.
- **SMART_DOCUMENT_Data_Common_V_1_02** This DTD contains all the elements that are shared by the MISMO Closing DTD and PRIA. The definitions in some cases have been merged.
- **MISMO_embedded_file.dtd,** the latest definition of the EMBEDDED_FILE element as published by the MISMO Architecture Workgroup, December 2002.

Use of Modular DTDs

When DTDs are simple, design constraints usually don't warrant modularizing the DTDs into separate files and areas of specification. But when the DTDs are complex or when components of the DTD are delivered from different sources and used by different DTDs, breaking the DTD into fragments or modules is a good solution. Since the data definitions for SMART Doc come from multiple sources and the sources have commonly defined elements, modularization of the DTDs has been performed. In order for modularization to work, each DTD may include other DTDs, but a single DTD may only be included once. The data DTDs for SMART Docs include multiple DTDs, which in turn include other DTDs.

Using the modularization approach allows for the SMART Docs to include any data DTDs.

Step 1: Create a Non-MISMO data DTD

For purpose of this use case, let's define the following DTD in the file other_data.dtd:

```
<!ELEMENT OTHER_DATA EMPTY>
```

Step 2: Modify the SMART Doc Data DTD to exclude the MISMO DTDs

XML DTDs are allowed to have conditional sections. With conditional statements in the DTD, it is possible to include or ignore a section from the DTD. The keywords INCLUDE and IGNORE are used for this. If the keyword INCLUDE is used, the conditional section is executed. If the keyword IGNORE is used, the conditional section is ***not*** executed. In the following, a conditional section has been constructed to include or exclude the SMART Doc DATA DTD, depending on the value of `include.emortgage_data_dtds`

The current SMART Doc DTD currently reads:

```
<!ENTITY % include.emortgage_data_dtds "INCLUDE">

<!ENTITY % exclude.smartdoc_data "IGNORE">
```

and executes the following include sequence:

```
<![ %include.emortgage_data_dtds; [
<!-- Set this entity to exclude duplicate definitions
-->
<!ENTITY % standalone-or-mixed-use "IGNORE"> <!--
Include common elements  DTD ... this DTD contains
shared definition between      -->
<!-- closing and PRIA release candidate DTDs
-->
<!ENTITY % sd-common-include SYSTEM
"SMART_DOCUMENT_Data_Common_V_1_0.dtd">
%sd-common-include;

<!-- include MISMO Closing Data DTD -->
<!ENTITY % sd-closing-include SYSTEM
"SMART_DOCUMENT_V_1_0_Closing_V_2_3.dtd">
%sd-closing-include;

<!ENTITY % embed-file-include SYSTEM
"mismo_embedded_file.dtd">
%embed-file-include;

<!-- include PRIA DTD -->
<!ENTITY % rc-PRIA-include SYSTEM
"SMART_DOCUMENT_PRIA_V_1_2_RC_2_0.dtd">
%rc-PRIA-include;

<!-- Note: MAIN MUST only contain elements included by
the  SMART DOCUMENT DATA DTD                  -->
<!ELEMENT MAIN (LOAN)>

]]>
```

This must be changed to

<!ENTITY % include.emortgage_data_dtds “IGNORE">

<!ENTITY % exclude.smartdoc_data " INCLUDE">

The implication of this change is that the following lines are now executed:

```
<![ %exclude.smartdoc_data; [

<!-- If the SMART_DOCUMENT DATA  DTD is NOT included
then the EMBEDDED_FILE DTD       -->
<!-- MUST be included  for the Package to work. There
is no data in the data section   -->
<!ENTITY % embed-file-include SYSTEM
"mismo_embedded_file.dtd">
%embed-file-include;
<!-- Note: MAIN is empty
-->
<!ELEMENT MAIN EMPTY>

]]>
```

Step 3: Include the Other Data DTD

Reference the other data DTD in the modified SMART Doc Data DTD:

Add the reference to:

```
<!ENTITY % embed-file-include SYSTEM
"mismo_embedded_file.dtd">
%embed-file-include;
```

Our sample DTD is called “other_data.dtd” and the revised references now look thus:

```
<!ENTITY % other-data-file-include SYSTEM
"other_data.dtd">
% other-data-file-include;

<!ENTITY % embed-file-include SYSTEM
"mismo_embedded_file.dtd">
%embed-file-include;
```

Step 4: Include the Other Data Containers

Reference the other data under the <MAIN> section:

```
<!ELEMENT MAIN (OTHER_DATA)>
```

Checklist

9 Develop your data DTD to be used by the SMART Doc Framework

9 Exclude the references to the MISMO SMART Doc DATA DTDs.

9 Include a reference to your data DTD

9 Change the <MAIN> definition to include your data elements

XML Structures Used	<DATA> (<MAIN>)
Known Issues	Your trading partners may or may not choose to process non-MISMO Data DTDs.
Other References	MISMO Engineering Guidelines Property Records Industry Association (PRIA) eRecording XML Standards, http://www.pria.us See Chapter 7.3 for methods to employ when you wish to use a non-MISMO Data DTD. See Chapter 7.4 for specific information regarding requirements for the data sections of eNotes. For other references, see Chapter 10: References

Chapter 8.1 Packages

This chapter describes how to create eMortgage Packages

Version 2.0

Revision History

Version	Date	Change
2.0	02/15/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc® Specification, V1.02 eMortgage ePackage Specification, V2.4

Overview

The SMART Doc Specification is used to create single, discrete documents, such as an eNote for a specific borrower relating to a specific loan. In certain implementations, such as an eNote with an image signature, there may be more than one file associated with a single SMART Doc.

A collection of files related to a single SMART Doc must be bound together in an eMortgage Package. eMortgage Packages can contain one or more SMART Docs and their associated files. The eMortgage Package has a structure that is suitable for transmission as well as archival purposes.

This chapter describes how to create eMortgage Packages in a variety of scenarios.

Pre Conditions

Image signed SMART Doc
The ability to create a Binary Document and to create a Base64-encoded version

Post Conditions

An eMortgage Package

Scenario

After the eNote and other closing documents are signed, the eNote may be delivered to another eVault. In our example, the eNote, the document and its signature images, are bundled and delivered in a Package.

Business Context

Businesses need the ability to bundle multiple documents, such as an entire loan package, as a logical unit. The bundling capability for SMART Docs is known as an eMortgage Package. An eMortgage Package can contain SMART Documents of any level and their related files. Trading partners can electronically send and receive multiple related documents as a single package.

The eMortgage package provides a flexible yet simple mechanism to collect a set of SMART Docs, a SMART Doc and its related files and/or encoded embedded files for exchange between two trading partners. This chapter describes how to create eMortgage packages.

The Packaging Specification was designed to fulfill the following requirements:

- Provide a single file that contains a collection of SMART Docs and other related files.
- Provide for tamper-evident sealing of the container file.
- Provide for unique naming of the contained files.
- Provide the ability to derive a “manifest” of the contained documents and files.
- Provide support for the containment of related files (for example, images) and for information about the type of contained file.

There are some aspects of bundling documents that the eMortgage Package **does not address**: compression, messages about the type of package, routing and transmission protocols, business logic and rules, and processing instructions. These aspects are outside of the scope of the eMortgage Package specification and are also outside the scope of this implementation guide.

Technical Context

eMortgage Package Types

The <EMORTGAGE_PACKAGE> is a flexible container that may contain a SMART Doc in XML, an embedded and encoded file and/or another eMortgage package.

There are two ways to package a SMART Doc: as XML or as Encoded Data. If the files to be packaged are all SMART Docs, then they should be packaged as XML. If the file set includes non-SMART Docs as well, then all files should be packaged as encoded data. However, the choice of native XML or encoded XML is implementation specific.

Encoded data does provide a mechanism to obscure the data in eMortgage packages. It should be noted that encoding is not the same as encryption. The purpose of encoding the file is to allow it to pass through an XML processor that can't handle the data directly. The purpose of encryption is to prevent unauthorized persons from viewing or using the information. It is possible for a message to use both encoding and encryption. Encryption is outside the scope of this chapter. Consult your trading partner’s requirements on encoding and encrypting eMortgage Packages.

Supplemental Document Delivery

Overview

When transmitting documents between eVaults using MERS eDelivery, if the document is not a SMART Doc with the associated document identifier contained internally, then the document identifiers which are part of the eMortgage Package structure must be present to identify the document contained in the package. The following are the requirements for packaging non-SMART Docs for MERS eDelivery:

High Level Requirements

- Each document must be contained within a separate MIMSO v2.4 eMortgage Package.
- An eNote must be either (a) included in a separate eMortgage Package in the same eDelivery, OR (b) delivered prior to the supporting document
- The embedded document must be contained within the <EMORTGAGE_DOCUMENT> element
- Package must identify the MERS MIN.
- The DOCUMENT_INFORMATION must be present and include:
    - EMBEDDED_FILE_ID (max 50 characters).
    - _Type using the enumerated list within the v2.4 specification.
    - If _Type "Other" is selected, the associated description must be supplied via _TypeOtherDescription. It is recommended that the enumerations from DocumentType the latest version of the MISMO standard be used for this purpose.
    - Embedded file "_EncodingType" must be "Base64" or "GzipBase64".
    - "MIMEType" must be one of the following: "application/pdf", "image/jpeg", "image/tiff", or "image/png".

Scenario 1: Packaging a SMART Doc as XML:

Step One: Create the <EMORTGAGE_PACKAGE> element

In most cases an eMortgage package will be exchanged between 2 or more parties. An <EMORTGAGE_PACKAGE> may be part of an enveloping transaction to provide a wrapper for the transfer of the package. For information about including the eMortgage package as part of a MISMO envelope, see Chapter 8.2, Enveloping. The <EMORTGAGE_PACKAGE> element is the top-level element of the package, whether it part of a MISMO request/response envelope or it stands alone.

```
<EMORTGAGE_PACKAGE  MISMOVersionIdentifier = "1.01">
[…]
</EMORTGAGE_PACKAGE>
```

Step Two: Set the _ID of the Package

This value should be set as a unique identifier for the entire package.

```
<EMORTGAGE_PACKAGE _ID="QWIE123092" MISMOVersionIdentifier = “1.01”>
```

Step Three: Set the PackageIdentifier (optional)

Provided the SMART Doc has not been tamper-sealed, you can set the PackageIdentifier in the SMART Doc to the above _ID. This will associate individual SMART Docs with a single eMortgage Package. In order to use this feature of the SMART Doc specification, you must know in advance the eMortgage Package identifier.

```
<SMART_DOCUMENT _ID="SMART_DOCUMENT"
PackageIdentifier=" QWIE123092">
```

Step Four: Insert the SMART_DOCUMENT element The SMART Doc element is added directly under the <EMORTGAGE_PACKAGE> element:

```
<EMORTGAGE_PACKAGE _ID="QWIE123092"
MISMOVersionIdentifier = “1.01”>
<SMART_DOCUMENT _ID="SMART_DOCUMENT"
MISMOVersionIdentifier = “1.01” PackageIdentifier="
QWIE123092">
           <HEADER>
                [...]
           </HEADER>
           <DATA>
                [...]
           </DATA>
           <VIEW _ID="Wxyz" _TaggedIndicator="True"
_MIMETypeDescription="">
                [...]
                </VIEW>
           <AUDIT_TRAIL>
                [...]
           </AUDIT_TRAIL>
      </SMART_DOCUMENT>
</EMORTGAGE_PACKAGE>
```

Step Five: Add a digital signature to provide a tamper evident wrap (optional)

The *Signature* element has been reused from the W3C XML-Signature Syntax and Processing Recommendation. The inclusion of the Signature element allows for the package to be digitally signed. The recommendation for XML Digital Signatures may be found at http://www.w3.org/TR/xmldsig-core/. See chapter 4.3 of this guide for information on creating a tamper evident signature.

Scenario 2: Packaging a SMART Doc with a Signature Image File:

If the SMART Doc includes image-based signatures you must follow steps 1 through 4 above to include the SMART Doc as XML and then perform the following steps:

Step One: Encode the Image Signatures files

Encoding is required for any binary files inserted into XML documents. There are certain characters in binary files that can be misinterpreted by XML processing software. An example is the "<" character. Encoding the binary data into ASCII characters allows the file to be treated by XML processing software as text. On the receiving end, the file is decoded back into the original file.

Use Base64 encoding for files to be embedded into the package. Base64 is an alphanumeric encoding format and is the preferred encoding method for attachments in email messages. The following is an excerpt from a Base64 encoded PDF file:

PFNNQVJUX0RPQ1VNRU5UIFBvcHVsYXRpbmdTeXN0ZW1Eb2N1bWVudElkZ
W50aWZpZXI9Inh4
[…]
eCI+DQoJCTxIRUFERVI+DQoJCQk8RE9DVU1FTlRfSU5GT1JNQVRJT04gX1R5
cGU9Ik5vdGUi

Step Two: Insert the EMBEDDED_FILE element

The <EMBEDDED_FILE> element contains several attributes that describe the encoded contents.

- `Set the _ID`. This uniquely identifies the EMBEDDED File
- `MIMEType`: This required attribute defines the MIME type of the embedded file. Consult the IETF RFC 2046 for acceptable MIME types.
- `_EncodingType`: Set _EncodingType="Base64". Base64 is currently the only supported encoding format.
- _Name contains the name of the file that was encoded. The file type extension of the name, such as ".xml" should be included.
- _Type contains the extension of the file that was encoded.
- Use _Description to provide a textual description of the contents of the file

```
      <EMBEDDED_FILE _ID="EMBEDDED_FILE"
MIMEType="application/jpg" _EncodingType="Base64"
```

```
_Name="sample_SMART_DOCUMENT" _Extension="jpg"
_Description="Borrower's signature in jpg">
```

Step Three: Insert the encoded image into the <DOCUMENT> element. The <DOCUMENT> element is included under the <EMBEDDED_FILE> element. Place the encoded representation of the image file into a <DOCUMENT> element:

```
<DOCUMENT>PFNNQVJUX0RPQ1VNRU5UIFBvcHVsYXRpbmdTeXN0ZW1Eb
2N1bWVudElkZW50aWZpZXI9Inh4 […]
X1RSQUlMPg0KCTwvU01BUlRfRE9DVU1FTlQ+</DOCUMENT>
```

You may choose to wrap the encoded file in a CDATA section:

```
<DOCUMENT>
       <![CDATA[...File Data Goes Here...]]>
</DOCUMENT>
```

Step Four: Add a digital signature to tamperseal (optional)
Add a tamper evident signature as described in Scenario 1, Step 5.

Scenario 3: Packages within Packages

Packages are recursive that is, packages may include other packages. In order to maintain relationships between documents, it is recommended to make use of the hierarchy that is part of the ePackaging specification. The hierarchy maintains the relationship of <EMORTGAGE_PACKAGE> files. For instance, an eNote SMART Doc with an image-based signature would comprise one package. The eNote package is included in a main package. This main package contains two packages: the eNote package and an Addendum to the eNote package.

Step One: Create the <EMORTGAGE_PACKAGE> element
Create the <EMORTGAGE_PACKAGE>. This package will contain the other packages.

```
<EMORTGAGE_PACKAGE MISMOVersionIdentifier = “1.01”>
</EMORTGAGE_PACKAGE>
```

Step Two: Set the _ID of the Package
This value should be set as a unique identifier for the entire package.

```
<EMORTGAGE_PACKAGE _ID="SMARTDOC_PACKAGE3"
MISMOVersionIdentifier = “1.01”>
```

Step Three: Add the packages
Follow the steps in both Scenario 1 and 2 above. Add the packages.

The following is an example of a main ePackage that contains two packages: one with a SMART Doc in XML that references an image in a jpg file and one that does not:

```
<EMORTGAGE_PACKAGE _ID="SMARTDOC_PACKAGE3"
MISMOVersionIdentifier = “1.01”>
   <EMORTGAGE_PACKAGE _ID="SMARTDOC_PACKAGE1"
MISMOVersionIdentifier = “1.01”>
       <SMART_DOCUMENT _ID="SMART_DOCUMENT" _Type=”Note”
PopulatingSystemDocumentIdentifier="xxx">
            <HEADER>
                  <DOCUMENT_INFORMATION _Type="Note"
_StateType="Signed"
NegotiableInstrumentIndicator="True"
MustBeRecordedIndicator="True"/>
[…]
            </HEADER>
            <DATA>
                  <MAIN>
[…]
                  </MAIN>
            </DATA>
            <VIEW _ID="Wxyz" _TaggedIndicator="True"
_MIMETypeDescription="">
[…]

            </VIEW>
            <AUDIT_TRAIL>
                  <AUDIT_ENTRY _PerformedByName=""
_ActionType="Unpopulated" _DateTime=""/>
[…]
            </AUDIT_TRAIL>
      </SMART_DOCUMENT>
      <EMBEDDED_FILE _ID="EMBEDDED_FILE"
MIMEType="application/jpg" _EncodingType="Base64"
_Name="sample_SMART_DOCUMENT" _Extension="jpg"
_Description="Borrower's signature in jpg">
      <DOCUMENT>PFNNQVJUX0RPQ1VNRU5UIFBvcHVsYXRpbmdTeXN0
ZW1Eb2N1bWVudElkZW50aWZpZXI9Inh4 […]
X1RSQUlMPg0KCTwvU01BUlRfRE9DVU1FTlQ+</DOCUMENT>
  </EMBEDDED_FILE>
   </EMORTGAGE_PACKAGE>
<EMORTGAGE_PACKAGE _ID="SMARTDOC_PACKAGE2"
MISMOVersionIdentifier = “1.01”>
       <SMART_DOCUMENT _ID="SMART_DOCUMENT" _Type=”Note”
PopulatingSystemDocumentIdentifier="xxx">
            <HEADER>
```

```
                 <DOCUMENT_INFORMATION _Type="Note"
_StateType="Signed"
NegotiableInstrumentIndicator="True"
MustBeRecordedIndicator="True"/>
[…]
           </HEADER>
           <DATA>
                 <MAIN>
[…]
                 </MAIN>
           </DATA>
           <VIEW _ID="Wxyz" _TaggedIndicator="True"
_MIMETypeDescription="">
[…]

           </VIEW>
           <AUDIT_TRAIL>
                 <AUDIT_ENTRY _PerformedByName=""
_ActionType="Unpopulated" _DateTime=""/>
[…]
           </AUDIT_TRAIL>
      </SMART_DOCUMENT>
   </EMORTGAGE_PACKAGE>
</EMORTGAGE_PACKAGE>
```

Step Four: Create KEY elements (optional)

The <KEY> element provides a series of name value pairs. These may be used for any purpose. In this example the <KEY> element is used to provide a high level description of the package contents:

```
<EMORTGAGE_PACKAGE _ID="SMARTDOC_PACKAGE3"
MISMOVersionIdentifier = “1.01”>
   <KEY Name="SMARTDOC_PACKAGE1"  _Value="Note"/>
   <KEY Name="SMARTDOC_PACKAGE2"  _Value="Addendum">
   <EMORTGAGE_PACKAGE _ID="SMARTDOC_PACKAGE1"
MISMOVersionIdentifier = “1.01”>
[…]
   </EMORTGAGE_PACKAGE>
   <EMORTGAGE_PACKAGE _ID="SMARTDOC_PACKAGE2"
MISMOVersionIdentifier = “1.01”>
[…]
   </EMORTGAGE_PACKAGE>
</EMORTGAGE_PACKAGE>
```

Step Five: Add a digital signature to tamperseal (optional)

Add a tamper evident signature as described in Scenario 1, Step 5.

Scenario 4: Packaging a SMART Doc as Encoded Data

Step One: Encode embedded file(s)

Encoding is required for any binary files inserted into XML documents. However, you may also choose to encode the XML SMART Doc. There are several reasons you may wish to do this. One is the use of unique identifiers in SMART Docs. Identifiers may not be unique across SMART Docs within the same package. In this situation you will want to encode the XML SMART Docs to avoid conflicts. Note: encoding is not the same as encryption.

Step Two: Encode the SMART Doc

Use GzipBase64 encoding for SMART Docs to be embedded into the package. GzipBase64 is an alphanumeric encoding format:

PFNNQVJUX0RPQ1VNRU5UIFBvcHVsYXRpbmdTeXN0ZW1Eb2N1bWVudElkZ
W50aWZpZXI9Inh4
[…]
eCI+DQoJCTxIRUFERVI+DQoJCQk8RE9DVU1FTlRfSU5GT1JNQVRJT04gX1R5
cGU9Ik5vdGUi

Step Three: Insert the EMBEDDED_FILE element

The <EMBEDDED_FILE> element contains several attributes that describe the encoded contents.

- `Set the _ID`. This uniquely identifies the EMBEDDED File
- `MIMEType`: This required attribute defines the MIME type of the embedded file. Consult the IETF RFC 2046 for acceptable MIME types.
- `_EncodingType`: Set the flags for _EncodingType="GzipBase64". GzipBase64 is currently the only supported encoding format.
- _Name contains the name of the file that was encoded
- _Type contains the extension of the file that was encoded.
- Use _Description to provide a textual description of the contents of the file

```
      <EMBEDDED_FILE _ID="EMBEDDED_FILE"
MIMEType="text/xml" _EncodingType="GzipBase64"
_Name="encoded_SMART_DOCUMENT" _Extension="xml"
_Description="Encoded SMART Doc">
```

Step Four: Insert the encoded document into the <DOCUMENT> element

The <DOCUMENT> element is included under the <EMBEDDED_FILE> element.

Place the encoded representation of the image file into a <DOCUMENT> element:

```
<DOCUMENT>PFNNQVJUX0RPQ1VNRU5UIFBvcHVsYXRpbmdTeXN0ZW1Eb
2N1bWVudElkZW50aWZpZXI9Inh4 […]
X1RSQUlMPg0KCTwvU01BUlRfRE9DVU1FTlQ+</DOCUMENT>
```

You may choose to wrap the encoded file in a CDATA section:

```
<DOCUMENT>
       <![CDATA[...File Data Goes Here...]]>
</DOCUMENT>
```

Step Five: Add a digital signature to tamperseal (optional)
Add a tamper evident signature as described in Scenario 1, Step 5.

Scenario 5: Extracting a Manifest

A manifest is a "packaging list" of contents. Using XSL, it is possible to derive a manifest from the eMortgage Package. The following XSL is one potential method of deriving a manifest:

```
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet
version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" version="1.0" omit-
xmldeclaration="no" encoding="UTF-8"/>
 <xsl:template match="EMORTGAGE_PACKAGE">
           <xsl:element name="MANIFEST">
                <xsl:for-each
select="//SMART_DOCUMENT|//EMBEDDED_FILE|EMORTGAGE_PACK
AGE">
                     <xsl:element name="File">
                          <xsl:attribute
name="PackageSequenceNumber"><xsl:value-of
select="position()"/></xsl:attribute>
                      <xsl:attribute
name="PackageItem"><xsl:value-of
select="name()"/></xsl:attribute>
                     </xsl:element>
                </xsl:for-each>
           </xsl:element>
      </xsl:template>
</xsl:stylesheet>
```

The following is a sample eMortgage Package with the above XSL stylesheet referenced:

```
<?xml-stylesheet type="text/xsl" href="manifest.xsl"?>

<EMORTGAGE_PACKAGE _ID="SMARTDOC_PACKAGE">
      <SMART_DOCUMENT _ID="SMART_DOCUMENT"
PopulatingSystemDocumentIdentifier="xxx">
            <HEADER>
                  <DOCUMENT_INFORMATION _Type="Note"
_StateType="Signed"
NegotiableInstrumentIndicator="True"
MustBeRecordedIndicator="True"></DOCUMENT_INFORMATION>
            </HEADER>
            <DATA>
                  <MAIN>
                  </MAIN>
            </DATA>
            <VIEW _ID="Wxyz" _TaggedIndicator="True"
_MIMETypeDescription=""></VIEW>
            <AUDIT_TRAIL>
                  <AUDIT_ENTRY _PerformedByName=""
_ActionType="Unpopulated" _DateTime=""></AUDIT_ENTRY>
            </AUDIT_TRAIL>
      </SMART_DOCUMENT>
      <EMBEDDED_FILE _ID="EMBEDDED_FILE"
MIMEType="application/xhtml+xml"
_EncodingType="GzipBase64"
_Name="sample_SMART_DOCUMENT" _Extension="xml"
_Description="Borrower's signature in jpg">

 <DOCUMENT>PFNNQVJUX0RPQ1VNRU5UIFBvcHVsYXRpbmdTeXN0
ZW1Eb2N1bWVudElkZW50aWZpZXI9Inh4
eCI+DQoJCTxIRUFERVI+DQoJCQk8RE9DVU1FTlRfSU5GT1JNQVRJT04
gX1R5cGU9Ik5vdGUi
IF9TdGF0ZVR5cGU9IlVucG9wdWxhdGVkIiBOZWdvdGlhYmxlSW5zdHJ
1bWVudEluZGljYXRv
cj0iVHJ1ZSIgTXVzdEJlUmVjb3JkZWRJbmRpY2F0b3I9IlRydWUiPjw
vRE9DVU1FTlRfSU5G
T1JNQVRJT04+DQoJCTwvSEVBREVSPg0KCQk8VklFVyBfSUQ9Inh5eiI
gX1RhZ2dlZEluZGlj
YXRvcj0iVHJ1ZSIgX01JTUVUeXBlRGVzY3JpcHRpb249IiI+PC9WSUV
XPg0KCQk8QVVESVRf
VFJBSUw+DQoJCQk8QVVESVRfRU5UUlkgX1BlcmZvcm1lZEJ5TmFtZT0
iIiBfQWN0aW9uVHlw
ZT0iVW5wb3B1bGF0ZWQiIF9EYXRlVGltZT0iIj48L0FVRElUX0VOVFJ
ZPg0KCQk8L0FVRElU
X1RSQUlMPg0KCTwvU01BUlRfRE9DVU1FTlQ+</DOCUMENT>
      </EMBEDDED_FILE>
</EMORTGAGE_PACKAGE>
```

Using an XSL processor, the following manifest is generated:

```
<MANIFEST>
      <File PackageSequenceNumber="1"
PackageItem="SMART_DOCUMENT"/>
      <File PackageSequenceNumber="2"
PackageItem="EMBEDDED_FILE"/>
</MANIFEST>
```

XML Structures Used

`<EMORTGAGE_PACKAGE>`
`<SMART_DOCUMENT>`

`<EMBEDDED_FILE>`

`<DOCUEMENT>`

`<KEY>`

Known Issues

GzipBase64 encoding results in a representation approximately 37% larger than the original file.

Other References

The eMortgage Package specification reuses the MISMO embedded file as defined by the MISMO Architecture workgroup.

For other references see Chapter 10: References.

Chapter 8.2: Enveloping

This chapter describes how to create a MISMO® envelope for an eMortgage Package.

Version

2.0

Revision History

Version	Date	Change
2.0	02/15/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc® Specification V 1.02

eMortgage Packaging Specification V1.01

MISMO Enveloping Specification Version 2.3

Overview

This chapter provides instruction on how to place an eMortgage package within a MISMO request and response envelope.

Pre Conditions

eMortgage Package

Post Conditions

MISMO request and/or response envelope.

Business Context

In most cases an eMortgage package will be exchanged between 2 or more parties. An eMortgage Package may be part of an enveloping transaction to provide a wrapper for the transfer of the package. An enveloping transaction typically has two types defined:

- Transaction Request: Envelope used in requesting services or products, such as requesting borrower signatures or requesting data population of unpopulated SMART Docs.
- Transaction Response: Envelope used by vendors delivering services or products such as a signing platform or a document preparation company that populates an unpopulated SMART Doc template.

MISMO has developed a set of Transaction Envelope DTDs that can be used to wrap data, such as an eMortgage Package. The MISMO Envelope DTDs contain basic information common to most transactions – elements that identify the requesting party, receiving party, responding party and other reference data that is commonly exchanged between business partners. The use of the MISMO Transaction Envelope is optional. Some business partners may prefer to use other methodologies such as SOAP (Simple Object Access Protocol) to wrap eMortgage packages. This chapter describes how to use the MISMO envelope with eMortgage Packages.

This chapter of the implementation guide covers both the Transaction Request

Envelope for requesting services for an eMortgage package and the Transaction Response envelope used to deliver an eMortgage package.

A lender sends an eMortgage package in a MISMO request envelope to a closing

Scenario A lender sends an eMortgage package in a MISMO request envelope to a closing agent for signatures as a MISMO request. The closing agent responds by returning the signed eMortgage package in a MISMO response envelope.

Technical Guidance

Requests

Step 1: Modify the request envelope to accept eMortgage Packages.
In order to utilize the MISMO enveloping DTDs with an eMortgage package, the DTDs must be modified to include the eMortgage package. For requests, the <REQUEST> element holds the <REQUEST_DATA> which is generically defined as:

```
<!ELEMENT REQUEST_DATA ANY>
```

To create an eMortgage package request, this should be changed to:

```
<!ELEMENT REQUEST_DATA (EMORTGAGE_PACKAGE)*>
```

and the eMortgage package DTD should be included by the request envelope DTD:

```
<!ENTITY % em-include SYSTEM
"RC3_eMortgage_Package.dtd" >
%em-include;
```

This change allows for the inclusion of eMortgage packages within the request data.

Step 2: Create the Request Envelope
The top-level structure of the Request Envelope is <REQUEST_GROUP> and it contains information necessary for the request.

```
<REQUEST_GROUP MISMOVersionID="2.3">
 […]
</REQUEST_GROUP>
```

Step 3: Create the Requesting Party

The Requesting Party is the customer of the service. The <REQUESTING_PARTY> identifies the name and address of the requesting party.

```
<REQUEST_GROUP MISMOVersionID="2.3">
 <REQUESTING_PARTY _Name="XYZ Closing Agent"
_StreetAddress="21650 Greene Street" _City="Boston"
_State="MA" _PostalCode="02456"/>
[…]
</REQUEST_GROUP>
```

Step 4: Create the Receiving Party
The Receiving party is the provider of a service. The <RECEIVING_PARTY> identifies the name and address of the party that will receive the request.

```
<REQUEST_GROUP MISMOVersionID="2.3">
 <REQUESTING_PARTY _Name="XYZ Closing Agent"
_StreetAddress="21650 Greene Street" _City="Boston"
_State="MA" _PostalCode="02456"/>
 <RECEIVING_PARTY _Name="ABC SigningServices"
_StreetAddress="7200 Main Street" _City="Atlanta"
_State="GA" _PostalCode="30010"/>
[…]
</REQUEST_GROUP>
```

Step 5: Create the Request
The request will contain the eMortgage Package,

```
<REQUEST_GROUP MISMOVersionID="2.3">
 <REQUESTING_PARTY _Name="XYZ Closing Agent"
_StreetAddress="21650 Greene Street" _City="Boston"
_State="MA" _PostalCode="02456"/>
 <RECEIVING_PARTY _Name="ABC SigningServices"
_StreetAddress="7200 Main Street" _City="Atlanta"
_State="GA" _PostalCode="30010"/>
 <REQUEST RequestDatetime="2002-01-08T17:19:12"
InternalAccountIdentifier="ABC-0732">
  <KEY _Name="XYZ Transaction ID"
_Value="702430023"/>
  <KEY _Name="XYZ Portfolio ID" _Value="XYZ20020030"/>
  <REQUEST_DATA>
   <EMORTGAGE_PACKAGE _ID="ZZZ">
               […]
              </EMORTGAGE_PACKAGE>
  </REQUEST_DATA>
 </REQUEST>
</REQUEST_GROUP>
```

Responses

In responses, the Requesting Party becomes the Respond To Party and the Receiving Party becomes the Responding Party. The RESPONDING_PARTY and RESPOND_TO_PARTY are similar in structure to RECEIVING_PARTY and REQUESTING_PARTY

Step 1: Modify the response envelope to accept eMortgages
In order to utilize the MISMO enveloping DTDs with an eMortgage package, the DTDs must be modified to include the eMortgage package.

For responses, the <RESPONSE> element holds the <RESPONSE_DATA> which is generically defined as:

```
<!ELEMENT RESPONSE_DATA ANY>
```

To create an eMortgage package request, this should be changed to:

```
<!ELEMENT RESPONSE_DATA (EMORTGAGE_PACKAGE)*>
```

and the eMortgage package DTD should be included by the response envelope DTD:

```
<!ENTITY % em-include SYSTEM
"RC3_eMortgage_Package.dtd" >
%em-include;
```

This change allows for the inclusion of eMortgage packages within the response data.

Step 2: Create the Response Envelope

The top-level structure of the Response Envelope is <RESPONSE_GROUP> and it contains information necessary for the request.

```
<RESPONSE_GROUP MISMOVersionID="2.3">
 […]
</RESPONSE_GROUP>
```

Step 3: Create the Respond To Party

```
<RESPONSE_GROUP MISMOVersionID="2.3">
 <RESPOND_TO_PARTY _Name="XYZ Closing Agent"
_StreetAddress="21650 Greene Street" _City="Boston"
_State="MA" _PostalCode="02456"/>
[…]
</RESPONSE_GROUP >
```

Step 4: Create the Responding Party

```
<RESPONSE_GROUP MISMOVersionID="2.3">
 <RESPOND_TO_PARTY _Name="XYZ Closing Agent"
_StreetAddress="21650 Greene Street" _City="Boston"
_State="MA" _PostalCode="02456"/>
 <RESPONDING_PARTY _Name="ABC SigningServices"
_StreetAddress="7200 Main Street" _City="Atlanta"
_State="GA" _PostalCode="30010"/>
[…]
```

```
</RESPONSE_GROUP>
```

Step 5: Create the Response

```
<RESPONSE_GROUP MISMOVersionID="2.3">
 <RESPOND_TO_PARTY _Name="XYZ Closing Agent"
_StreetAddress="21650 Greene Street" _City="Wellsley"
_State="MA" _PostalCode="02456"/>
 <RESPONDING_PARTY _Name="ABC SigningServices"
_StreetAddress="7200 Main Street" _City="Atlanta"
_State="GA" _PostalCode="30010"/>
 <RESPONSE RequestDatetime="2002-01-08T17:19:12"
InternalAccountIdentifier="ABC-0732">
  <KEY _Name="XYZ Transaction ID"
_Value="702430023"/>
  <KEY _Name="XYZ Portfolio ID" _Value="XYZ20020030"/>
  <RESPONSE_DATA>
   <EMORTGAGE_PACKAGE _ID="ZZZ">
               […]
              </EMORTGAGE_PACKAGE>
  </RESPONSE_DATA>
 </RESPONSE>
</RESPONSE_GROUP>
```

Checklist

9 Modify the request and response DTDs to include the eMortgage Package DTDs

9 Create the request

9 Create the response

9 Consult your trading partner’s requirements for transmission of the envelopes

XML Structures Used

```
<EMORTGAGE_PACKAGE>
<REQUEST_DATA>
<REQUEST>
<REQUEST_GROUP>
```

Known Issues

Other References

The eMortgage Package specification reuses the MISMO embedded file as defined by the MISMO Architecture workgroup.

For other references, see Chapter 10: References.

Chapter 8.3: MERS® eRegistry and SMART Docs

This chapter describes the MERS® eRegistry requirements for Category 1 SMART Docs®

Version

2.0

Revision History

Version	Date	Change
2.0	03/13/2019	General revisions; added MERS® eRegistry SMART Doc requirements.
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification v1.02

Overview

The MERS® eRegistry is a compliance vehicle to satisfy certain requirements of the Uniform Electronic Transactions Act (UETA) and the federal Electronic Signatures in Global and National Commerce Act (E-SIGN), which provide an owner of an eNote (the Controller) with the status of a "Holder in Due Course."

An eNote issued and transferred in compliance with Section 16 of UETA or Title II of E-SIGN is called a Transferable Record (TR). Specifically, Section 16 of UETA and Title II of E-SIGN require that the party in control of the Authoritative Copy (AC) of the TR at any given point in the life cycle of an eNote be readily identified. The MERS® eRegistry is the system of record that identifies the Controller of a registered eNote and the custodian (Location) of the Authoritative Copy of the eNote.

This document describes the MERS® eRegistry concept and the requirements necessary to register eNotes in the MERS® eRegistry that are Category 1 SMART Docs. The requirement to present an eNote at registration is investor specific.

Pre Conditions

Document States: Signed and Tampersealed

Document Categories: 1

Post Conditions

Document States: Signed and Tampersealed

Document Categories: 1

SMART Doc is ready to submit the MERS® eRegistry.

Business Context

The eRegistry

As described above, the MERS® eRegistry has been designed to establish ownership of eNotes in compliance with eSignature laws. The Controller of each eNote, a term coined to describe the owner who establishes ownership by means of control, is determined solely by the MERS eRegistry. The MERS® eRegistry is the tool by which possession of a paper note is supplanted by control of an electronic note's Authoritative Copy as a means of asserting "Holder in Due Course" status. The MERS® eRegistry lists the Controller of each eNote registered, the Location of the Authoritative Copy of the eNote, and any Servicing Agent designated to make discrete updates to the record by the Controller. The MERS® eRegistry is the definitive source of ownership information, regardless of any alternative claims by those who purport to "hold" an electronic copy of an eNote.

In order for a SMART Doc eNote to become a Transferable Record under the law within the MERS® eRegistry paradigm, the eNote must include in the VIEW element the eNote language defined in chapter 5.4, and the eNote must be registered in the MERS® eRegistry as soon as possible after the tamper-evident seal has been applied (individual lenders and investors may issue their own guidelines for a time limit on registering eNotes).

MERS® eRegistry Requirements

The MERS® eRegistry has several requirements for SMART Doc eNotes.

The eNote must:

- Be a version 1.02 Category 1 SMART Doc.
- Include a header that specifies a document `_Type` of "Note" and a `SMARTDocumentCategoryType` of "1".
- Contain a Mortgage Identification Number (MIN) to uniquely identify the eNote on the MERS® eRegistry. See the *MERS® eRegistry Procedures Manual* for details on the MIN.
- Contain a `DATA` element as specified in chapter 7.4.
- Contain a `VIEW` element with the eNote language defined in chapter 5.4.
- Contain a final tamper-evident seal applied with a digital signature (see chapter 4.3).

The XML *Registration Request* submitted to the MERS® eRegistry must:

- Include additional data as specified in the *XML DTD: Registration* chapter of the *MERS® eRegistry Programming Interface Guide.*
- Use version 2.4 of the `EMORTGAGE_PACKAGE` container to include the SMART Doc and any related files.
- Encode the eNote and any related image files in Base64 format.

The `EMORTGAGE_PACKAGE` container must occur once and not include any:

- `SMART_DOCUMENT` or `EMORTGAGE_DOCUMENT` container.
- File not referenced by the eNote.

Scenario

The borrower is at the closing table and ready to sign the eNote. eSignature laws require that the borrower not only sign to attest to the content of the eNote but also attest to the fact that they agree to issue an eNote.

The eNote language states that the identity of the Note Holder and any person to whom the eNote is later transferred will be recorded in a registry maintained by MERSCORP Holdings, Inc., a Deleware Corporation (i.e., the MERS® eRegistry) or in another registry to which the records are later transferred.

Once the borrower(s) have signed the eNote, the tamper-evident seal must be applied (see chapter 4.3). The eNote should be registered on the MERS® eRegistry as soon after the application of the tamper-evident seal as possible, and it must be completed within one business day.

Technical Guidance

There are several technical considerations when preparing Category 1 SMART Doc eNotes for registration on the MERS® eRegistry.

Step 1: Add the MIN Number to the Data Element

The SMART Docs must: contain a MIN in the DATA element as shown below:

```
<DATA _ID="FNMA_Sample_Data_3200">
  <MAIN>
    <LOAN MISMOVersionIdentifier="2.3">
      <_APPLICATION>
        <LOAN_PRODUCT_DATA>
          <LOAN_FEATURES
          ScheduledFirstPaymentDate="10/01/01"
          LoanMaturityDate="09/01/2031"
          OriginalPrincipalAndInterestPaymentAmount=
          "763.02"/>
            <LATE_CHARGE _GracePeriod="15"
            _Rate="4.000"/>
            <NOTE_PAY_TO _StreetAddress="P.O. Box
            3050" _City="Columbia" _State="MD"
             _PostalCode="21045-6050"/>
          </LOAN_FEATURES>
        </LOAN_PRODUCT_DATA>
        <MERS MERS_MINNumber="123451234512345123"/>
        <MORTGAGE_TERMS NoteRatePercent="8.625"
        PaymentRemittanceDay="1"
        OriginalLoanAmount="96500.00"
        LenderLoanIdentifier="04405355"/>
        <PROPERTY _StreetAddress="748 N. Main
        Street" _City="Louisburg" _State="NC"
        _PostalCode="27549"/>
        <BORROWER BorrowerID="B123456789"
        _FirstName ="Richard" _MiddleName="R."
        _LastName="Bradley"
        _HomeTelephoneNumber="123-456- 7890"/>
      </_APPLICATION>
      <_CLOSING_DOCUMENTS>
        <EXECUTION _Date="08142001"
        _City="Louisburg" _State="NC"/>
        <LENDER _UnparsedName="Columbia National
        Incorporated"/>
      </_CLOSING_DOCUMENTS>
    </LOAN>
  </MAIN>
[…]
```

```
</DATA>
```

Step 2: Tamperseal the document

Every eNote submitted to MERS® eRegistry must contain a final tamper-evident seal applied with a digital signature (see chapter 4.3 for details).

Step 3: Submit the SMART Doc for Registration

You must submit the SMART Doc eNote for registration as soon as possible after the document has been tampersealed. For specific information on the MERS® eRegistry, consult the MERS® eRegistry documentation on the Member website.

Checklist

ü Check that the `<DATA>` element contains a MIN in the `MERS_MINNumber` attribute of the `<MERS>` element.

ü Apply a tamper evident digital signature, after all parties have signed the document (see chapter 2.4).

ü Register the eNote on the MERS® eRegistry at the soonest possible time after the tamper evident digital signature has been applied.

XML Structures

`<DATA>`

`<MERS>`

`<SIGNATURES>`

`<Signature>`

Other References

See Chapter 4 for information on signatures in SMART Docs.

See Chapter 5 for information on the types of allowable `VIEWS` and requirements in the `VIEW` element.

See Chapter 10: References for all other references.

Chapter 9: The AUDIT TRAIL

This chapter provides instruction on how to maintain an AUDIT TRAIL for a SMART Doc®.

Version 2.0

Revision History

Version	Date	Change
2.0	02/15/2019	Updates and corrections
1.0	01/26/2004	Release to industry

Relevant Specifications

SMART Doc Specification V1.02

Overview

A single SMART Doc will progress through various states during its lifecycle and a log of these events is kept to record the state transitions and signers of the document. This section describes how to create and add entries to the audit trail.

Pre Conditions

Document State: All
Document Categories: All

Post Conditions

Audit Trail Entries

Business Context

The `<AUDIT_TRAIL>` stores a set of audit entries that records all the activities performed on the SMART Doc. The `<AUDIT_TRAIL>` is required and must at a minimum log the state transitions and signers of the SMART Doc. As an example, every time a signature is applied, a new entry is placed in the `<AUDIT_TRAIL>` section with the action set to “Signed:” indicating that the document was signed by the Borrower or Notary or Tampersealer.

There is no requirement to tamperseal the `<AUDIT_TRAIL>`. However, standard business practices for individual systems may require that the `<AUDIT_TRAIL>` be included in the tamperseal. Consult your trading partner’s requirements before implementing the audit trail and tamperseal signatures.

Technical Guidance

An <AUDIT_ENTRY> element is used to log events in the <AUDIT_TRAIL>. It is an empty element that contains attributes that describe the logged event. It is recommended that the audit trail be used for significant changes to the document, such as transitions between document states and signing events.

The _PerformedByName attribute describes who created the entry. The action performed is encoded in the _ActionType attribute. The _DateTime attribute records the date and time the action occurred. The following is an XML document fragment for the <AUDIT_TRAIL> element:

```
 <AUDIT_TRAIL>
   <AUDIT_ENTRY _DateTime="2002-07-30T20:30:50Z"
_PerformedByName="Document Prep Company"
_ActionType="Unpopulated"/>
   <AUDIT_ENTRY _DateTime="2002-07-30T20:45:18Z"
_PerformedByName="Document Prep Company"
_ActionType="Populated"/>
   <AUDIT_ENTRY _DateTime="2002-07-31T18:06:59Z"
_PerformedByName="Closing Company"
_ActionType="Signable"/>
   <AUDIT_ENTRY _DateTime="2002-07-31T18:07:32Z"
_PerformedByName="Borrower" _ActionType="Signed"/>
 </AUDIT_TRAIL>
```

Step 1: Convert the Date and Time to UTC

In order for the date and time values in SMART Docs to be consistent and usable, all dates must be represented in a standard format and all times must be coordinated to a standard clock. The clock at Greenwich, England is used as the standard clock for international reference of time. The letter designator for this clock is Z. This time is sometimes referred to as Zulu Time because of its assigned letter. Times are written in military time or 24 hour format such as 1830Z. The official name is Coordinated Universal Time or UTC. Previously it had been known as Greenwich Mean Time or GMT but this has been replaced with UTC. See https://www.iso.org/standard/70907.html for more information on UTC formats and examples.

The time value for any timestamp, whether it is in the audit trail or is the tamperseal timestamp, MUST be universal time (that is to say, not a local time with a time zone offset).

Step 2: Add the AUDIT TRAIL ENTRY

Every time a state transition occurs or a signature is applied, a new <AUDIT_ENTRY> is placed in the <AUDIT_TRAIL> section:

```
<AUDIT_TRAIL>
```

```
  <AUDIT_ENTRY"/>
</AUDIT_TRAIL>
```

Note: If the document is in the unpopulated state, you may need to create the initial `<AUDIT_TRAIL>` element.

Step 2a: Add the Date and Time to the Entry

Add the UTC date to the `_DateTime` attribute in `<AUDIT_ENTRY>`:

```
<AUDIT_ENTRY _DateTime="2002-07-31T18:07:32Z"/>
```

Step 2b: Add the name of who performed the event Add the entity or person responsible for the event in the `_PerformedByName` attribute of `<AUDIT_ENTRY>`:

```
<AUDIT_ENTRY _DateTime="2002-07-31T18:07:32Z"
_PerformedByName="Borrower" />
```

Step 2c: Add the Event

Add event in the `_ActionType` attribute of `<AUDIT_ENTRY>`:

```
<AUDIT_ENTRY _DateTime="2002-07-31T18:07:32Z"
_PerformedByName="Borrower" _ActionType="Signed"/>
```

Note: The `_ActionType` attribute is an enumerated list of the states of the document: "`Unpopulated`", "`Populated`", "`Signable`", "`Recordable`", "`Signed`", "`Exported`", and the following other values: "`Voided`", "`PaperedOut`", "`Validated`", "`Other`”.

Checklist

9 Convert the date and time of the entry in the AUDIT_TRAIL to UTC

9 Add the UTC date and time to the _DateTime attribute in <AUDIT_ENTRY>

9 Add the logger of the event to the _PerformedByName attribute in <AUDIT_ENTRY>

9 Add the event type in the _ActionType attribute in <AUDIT_ENTRY>

XML Structures Used

<AUDIT_TRAIL>
<AUDIT_ENTRY>

Known Issues

Other References

See Chapter 10: References.

Chapter 10: References

This chapter provides a list of references used through out the SMART Doc® I-Guide

Version

2.0

Revision History

Version	Date	Change
2.0	03/13/2019	Created this separate Chapter 10 for the References and removed the Other References links from the individual chapters of the Implementation Guide.
1.0	01/26/2004	Release to industry. References were embedded in each section.

References

ESIGN: Electronic Signatures in Global and National Commerce Act ("ESIGN"), Pub. L. No. 106-229, 114 Stat. 464 (2000) (codified at 15 U.S.C. § 7001 et seq.). https://www.fdic.gov/regulations/compliance/manual/10/x-3.1.pdf

UETA: Uniform Electronic Transactions Act ("UETA") (1999). The UETA is a model act drafted, approved, and recommended for enactment in all the states by The National Conference of Commissioners on Uniform State Laws (NCCUSL). https://law.lis.virginia.gov/vacodepopularnames/uniform-electronic-transactions-act/

Time: For time format in the audit trail, see ISO 8601, the International Standard for the representation of dates and times https://www.iso.org/iso-8601-date-and-time-format and http://www.w3.org/TR/NOTE-datetime. Date and time format specified by the W3C.

XML: The SMART Doc specification is based on the W3C XML 1.0 Recommendation. The base specifications are XML 1.0, W3C Recommendation November 2008 (http://www.w3.org/TR/xml/), and Namespaces, December 2009 (http://www.w3.org/TR/REC-xml-names/).

XHTML: One of the potential views in the SMART Doc specification is an XHTML document. XHTML 1.0 is the W3C's Recommendation for XHTML, following on from earlier work on HTML 4.01, HTML 4.0, HTML 3.2 and HTML 2.0. XHTML 1.0 is a reformulation of HTML 4.01 in XML, and it combines the strength of HTML 4 with the power of XML. The XHTML Specification is the W3C Recommendation August 2002 (http://www.w3.org/TR/xhtml1/).

Xpointer: XPointer, which is based on the XML Path Language (XPath), supports addressing into the internal structures of XML documents. It allows for traversals of a document tree and choice of parts based on various properties, such as element types, attribute values, character content, and relative position. The SMART Doc specification makes use of XPointer and XPath referencing elements in the data and view sections of the document. The specification is the XML Pointer Language (XPointer) Version 1.0 W3C Candidate Recommendation March 2003 (http://www.w3.org/TR/xptr-element/) XPath is a language for addressing parts of an XML document, designed to be used by XPointer. The specification is XML Path Language (XPath) Version 2.0 W3C Recommendation October 2016 (http://www.w3.org/TR/xpath20/).