MeF Submission Composition Guide

Modernized e-File System

TIPSS-2 # 1066
Document Version 1.6
Date: August 12, 2020
Document Number: BMF136

Publication 5446 (12-2020) Catalog Number 74895N Department of the Treasury **Internal Revenue Service** www.irs.gov

Table of Contents

List of Figures

List of Tables

Introduction

Identification

This document is identified as the **MeF Submission Composition Guide**.

Scope

This document provides guidance to the IRS trading partners (Software Developers, Originators, and Transmitters) on how to compose Submissions, and Transmission Files that are sent to the IRS for processing by the Modernized e-File System.

Purpose

The purpose of this document is to describe the structure of a Submission and a Transmission file supported by MeF (Modernized e-File System). These structures are defined at a high level in the **ISS MEF State and Trading Partners ICD**. This guide references the static structures defined therein, and walks through them step-by-step to help understand them and to facilitate the composition process.

It is intended to help the Software Developers, Originators, and Transmitters in composing these artifacts in the format required by MeF.

MeF provides Web Services that can be invoked by Client Applications (Transmitter Systems, and State Systems) to conduct business with the MeF System. Data is exchanged between the Client Applications and MeF using Messages. The structure of these Messages is documented in the *ISS MEF State and Trading Partners ICD* and is not addressed in this document.

Document Organization

This document is organized into the following sections:

- Section 1 defines the purpose and scope of this document, as well as the parent, applicable, and reference documents.
- Section 2 provides guidance on transmission channel selection
- Section 3 describes the structure of a transmission file
- Section 4 provides guidance on composing a transmission file
- Section 5 provides guidance on composing a submission
- Section 6 describes how documents are attached to a submission
- Section 7 describes general philosophy on the definition of XML Schemas
- Section 8 describes how to validate a return against XML Schemas

Related Documentation

This section identifies the documents that are related to the contents of this document.

Parent Documents

The parent documents establish the criteria and technical basis for the existence of this document. The parent documents are:

- Functional Requirements
- *ISS MEF State and Trading Partners ICD*

Applicable Documents

Applicable documents are those documents whose content are considered to form a part of this document. The specified parts of the applicable documents carry the same weight as if they were stated within the body of this document. The applicable documents are the:

- None.

Reference Documents

Reference documents are those documents that, although not a part of this document, serve to amplify or clarify its contents, or dictate work policy or procedures. The specific reference documents are:

- *MeF Glossary*

Transmission Channel Selection

The MeF System allows you to transmit tax returns and extensions to the IRS through two channels: MeF Internet Filing Application (IFA) and MeF Application-to-Application (A2A). Table 2-1 highlights the different features of the two channels.

Table 2-1: Feature Comparison of Two Transmission Channels

FEATURES \ CHANNELS	IFA Channel	A2A Channel
Transmission Protocol	HTTP File Upload	Web Services
IRS vs. State Submission	IRS and State	IRS and State
Transmission File vs. Message	Transmission File	Message
MIME vs. MTOM Attachment	MIME Only	MIME or MTOM

The following sections provide general guidance on channel selection.

IFA Channel

If the transmitter chooses to use the IFA channel to communicate with MeF, they may send federal and state submissions to MeF. The transmission file must conform to the ZIP archive format using MIME. Acknowledgements must be retrieved from the same channel that was used to submit the return.

A2A Channel

If the transmitter chooses to use the A2A channel to communicate with MeF, they may send federal and/or state submissions to MeF. Acknowledgements must be retrieved from the same channel that was used to submit the return.

A transmitter needs to have web service client software installed on a client machine. The software shall send web services requests to the A2A web services end point, and submission data are attached to the request messages. The software shall also handle the web services responses. Such web services client software are developed by third-party software developers, not the IRS MeF (Modernized e-File) system.

If the web service client software is based on the Microsoft .NET technology, then MTOM attachment may be used. If the web service client software is based on J2EE or other open source technologies then either MIME or MTOM attachment may be used. Both MIME and MTOM attachments are accepted in the A2A channel.

Transmission File Structure

When using the IFA channel, a transmitter sends a transmission file to the MeF System. When using the A2A channel, a client application sends a message to the MeF System.

The structure of a transmission file and the structure of the web services message used to transmit submissions are similar: both use SOAP with attachments. The only differences between them are:

1. A web services message contains additional SOAP header elements for web services addressing and security;
2. A web services message may contain MIME attachments.

This section describes the structure of a transmission file.

Note: This document provides guidance to the transmitters and EROs in composing the transmission file and the submissions, respectively. For guidance on composing web service messages, please see the *ISS MEF State and Trading Partners ICD.*

Transmission File for IFA Channel

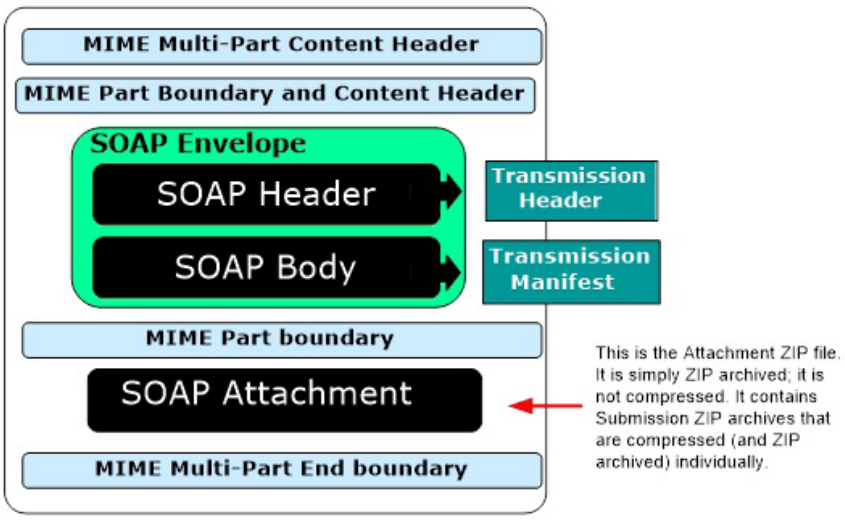
The transmission file for IFA channel is a MIME (Multipurpose Internet Mail Extensions) multi-part document that contains two parts and conforms to "SOAP 1.1 with attachments" standard. The first part of the multi-part document is the SOAP envelope and the second part is a SOAP attachment. The SOAP envelope contains transmission-level information, and the SOAP attachment contains one or more submissions. MIME boundaries separate the parts in the multi-part document.

The SOAP envelope consists of a SOAP header and a SOAP body. The SOAP header, also referred to as the *transmission header* in the MeF System, contains information about the transmitter and the transmission. The SOAP body, also referred to as the *transmission manifest*, contains a list of all submissions in the transmission file.

The SOAP attachment in the transmission file is a ZIP file. This ZIP file, also referred to as Attachment ZIP file, is not compressed; it is simply ZIP archived (i.e., compression is turned off when this file is created). This Attachment ZIP file contains one or more submissions that are themselves ZIP Archive files. These ZIP files, also referred to as Submission ZIP files, are compressed.

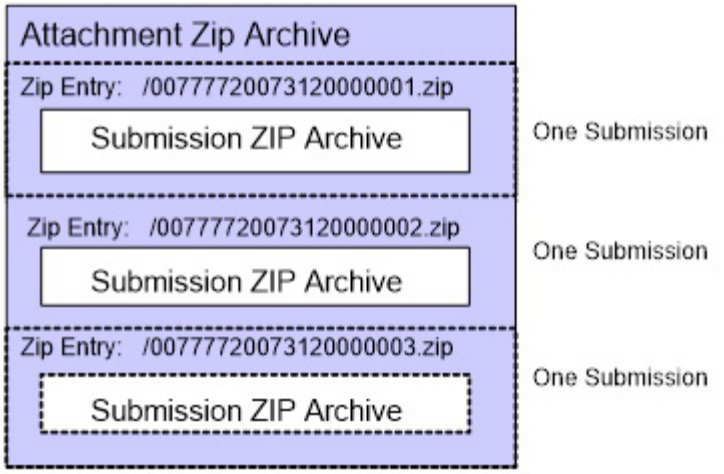
The structure of the transmission file is depicted in Figure 3–1 below, followed by the structure of the SOAP Attachment shown in Figure 3–2. Figure 3–3 shows the transmission file structure using MIME header fields.

Figure 3–1: Graphic Representation of a Transmission File Structure



The graphic below shows three submissions that have been grouped together in the Attachment ZIP Archive. Attachment may contain one or more submissions. Note that the names of the Submission ZIP Archives must be <SubmissionID.zip> and the SubmissionID must match the SubmissionID provided in the manifest.xml file that describes the submission. The year in the SubmissionID must be the processing year. If the transmitter chooses to use the A2A or IFA channel to communicate with MeF, an attachment may contain up to one hundred federal and/or state returns.

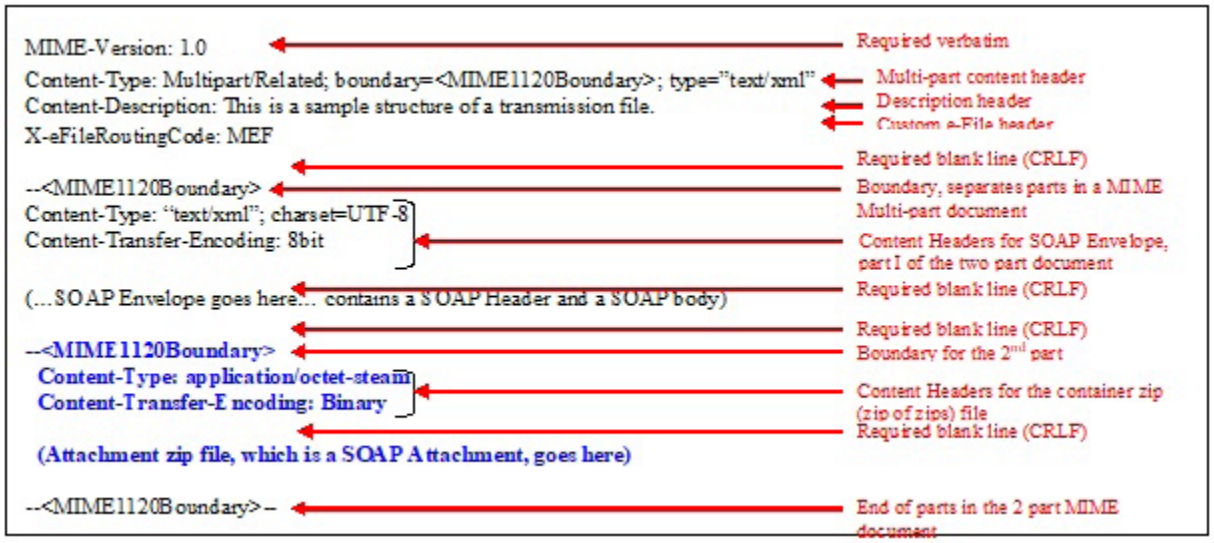
Figure 3–2: Graphic Representation of the Attachment ZIP Archive



The structure of the transmission file using MIME header fields is shown below, followed by explanations of the header fields and whether or not they are required. The text in the right column explains the purpose of the header fields.

Note: The values in brackets (e.g., "`<MIME1120Boundary>`") need to be replaced by the application composing the file.

Figure 3–3: Sample Transmission File Submitted to IFA



Header Fields Used in the Transmission File

The *MIME content headers*, or simply the content headers, are used to describe the contents of MIME parts. The *multi-part content header* (`Content-Type="Multipart/Related"`) specifies a boundary value that separates MIME parts in a MIME multi-part structure. Other header fields provide additional information about the MIME part. There is one multi-part content header for each MIME multi-part structure (the transmission file in our case).

Since the transmission file is a multi-part structure, there is one multi-part content header for the transmission file. The value in the *boundary* parameter (`"—MIMEBoundary"` in the transmission file structure above) of this content header separates (the two) parts in the transmission file. The first part is the SOAP envelope, and the second part is a SOAP attachment, a ZIP file that contains submissions).

MIME-Version Header Field

The MIME-Version header field is *required* at the top level in the transmission file. It is not required for each body part of a multi-part document. It indicates that the transmission file conforms to MIME version 1.0 standard. It must be included with the following verbatim text:

```
MIME-Version: 1.0
```

Content-Type Header Field

The Content-Type header field is *required* for each MIME part (SOAP envelope and the submissions ZIP file). The Content-Type header field describes the nature of the data in the MIME part by giving media type and subtype identifiers.

The value of the Content-Type header field for a multi-part header must be set to `"Multipart/Related"` and a value must be provided for both the *boundary* parameter and the type parameter. It must appear as follows:

```
Content-Type: Multipart/Related; boundary=<MIME1120Boundary>; type="text/xml"
```

The *boundary* parameter separates parts in the multi-part document: it separates the SOAP envelope and the submissions ZIP file. The value for the *boundary* parameter (`"<MIMEBoundary>"` above) is to be created by the application composing the transmission file. The *type* parameter indicates the content type of the multipart/related object. It must have the value `"text/xml"` for the SOAP envelope part as per SOAP 1.1 specification.

Note: The value of a parameter does not include the quotes that are used when describing the value. That is, the quotation marks in a quoted-string are not a part of the value of the parameter. So, the following two forms are equivalent:

```
Content-Type: text/xml; charset=UTF-8

Content-Type: "text/xml"; charset=UTF-8
```

The value of the Content-Type header field is set to either `"text/xml"` when describing an XML part, or to `"application/octet-steam"` when describing a ZIP archive part in a multi-part structure.

When the Content-Type header field is included in the *body part header* of the SOAP envelope, its value must be set to `"text/xml"` as shown below:

```
Content-Type: "text/xml"; charset=UTF-8
```

When the Content-Type header field is included in the *body part header* of an attachment ZIP, its value must be set to `"application/octet-steam"` as shown below:

```
Content-Type: "application/octet-steam"
```

The *charset* parameter must be set to the value "UTF-8" when the value of the header is "text/xml" This parameter is not required when the value of the header is "application/octet-steam".

The names of the parameters and the type and subtype values are not case sensitive, however, the parameter values are. For example, "text/xml" and "Text/xml" are equivalent. However, the value for the *boundary* parameter (in the multi-part content header) is case-sensitive. Hence, boundary=MIME1120Boundary, and boundary=Mime1120Boundary are not equivalent.

Content-Description Header Field

The Content-Description header field is **optional** for all MIME parts. It describes contents of the MIME part. It is not processed by the MeF System.

X-eFileRoutingCode Header Field

The X-eFileRoutingCode header field is **required** in the message header of the transmission file. It is a custom (not a standard MIME header) header field created specifically for use by the MeF System. Its purpose is to indicate the kind of data included in the transmission file.

For MeF use the value "MEF."

Content-Transfer-Encoding Header Field

The Content-Transfer-Encoding header field is *required* for MIME parts whose Content-Type header has the value "text/xml" or "application/octet-steam". Its value must be set to "8Bit" for MIME parts of Content-Type "text/xml", and it must be set to "Binary" for MIME parts of Content-Type "application/octet-steam". These values indicate that no encoding has been applied to the body part. The value for this header must be specified as follows:

1. The Content-Transfer-Encoding header for the MIME part that contains the XML data must be:

```
Content-Transfer-Encoding: 8Bit
```

2. The Content-Transfer-Encoding header for the MIME part that contains the ZIP file must be:

```
Content-Transfer-Encoding: Binary
```

Guidelines for Composing the Transmission File

The following guidelines apply when creating the MIME multi-part structure of the transmission file. These guidelines are the results of errors encountered by transmitters in prior releases of the MeF System

Guidelines on Creating MIME headers

1. Each MIME header must be on its own line, i.e., the Hex pattern: 0D 0A (CR and LF), is required before and after the declaration of each MIME Header.
2. The two preceding hyphens ("- -") are required in the beginning of MIME boundary
3. MIME boundary must not be followed by a blank line.
4. A blank line is required right before the declaration of a MIME boundary. Since each MIME boundary is required to be on its own line, and this guideline requires that there be a blank line before it, this means that the CRLF sequence must repeat (i.e., CRLFCRLF) before the first hyphen of the MIME boundary starts.

   Note: In some instances, the data might already have a CR and LF (Hex pattern: 0D 0A) before the two hyphens ("- -") where a MIME boundary starts, but error(s) can still be thrown due to "Missing start boundary". In such instances, the proper Hex pattern needed to correct the problem is: 0D 0A 0D 0A
5. A blank line is required before the declaration of the SOAP envelope. Since the XML declaration of the

SOAP envelope must start on its own line, and this guideline requires that there be a blank line before it, this means that the CRLF sequence must repeat (i.e., `CRLFCRLF`) before the XML declaration.

General Guidelines for Creating Transmission Files

1. Some editors insert BOM (Beginning of Message) characters in text files, which result in X0000-005 errors. These characters are visible using od –c command on Unix or similar command in Windows platform. Please make sure you remove any BOM characters before sending to MeF.
2. When creating submission zip archive make sure to choose only the actual files when adding them to the zip archive. When adding directories, some zip utilities add extra entries for directories as well as files which result in X0000-0018 error. As an example if you have a manifest/manifest.xml and xml/submission.xml, to verify make sure you only have two entries in your zip package and not four entries.
3. For IFA transmissions make sure your transmission file is not named with an .xml extension. This causes the browser to treat the file as xml and add encodings to it. MeF does not read such transmission files.

What does the transmitter do when composing a transmission file?

A transmitter is an IRS authorized *e-file* provider who sends transmission files to the MeF System. The transmitter receives electronic data for the submissions from the EROs (Electronic Return Originators), creates the transmission file, and transmits it to the MeF System. This chapter describes the activities involved in composing a transmission file.

Transmission File Components

A transmission file contains two parts: a SOAP envelope (also referred to as the *transmission envelope)*, and a SOAP attachment, referred to as the attachment ZIP file. The SOAP envelope, an XML structure, maintains transmission level information, and the SOAP attachment is a ZIP file that contains submissions. Each submission is also a ZIP file in itself. So, the attachment ZIP file is a ZIP of ZIP files, i.e., a ZIP file that contains other ZIP files, each of which is a submission.

The transmitter creates the SOAP envelope (which includes a SOAP Header and a SOAP Body) and the attachment ZIP file, as described in the following sections.

Creating the SOAP Envelope

The SOAP envelope consists of a SOAP header and a SOAP body. The SOAP header, also referred to as the *transmission header* in the MeF System, contains information about the transmitter and the transmission. The SOAP body, also referred to as the *transmission manifest*, contains a list of all submissions in the attachment ZIP file.

To create the SOAP envelope, the transmitter needs to know the `SubmissionID` of each submission that is to be included in the transmission file. The ERO provides the `SubmissionID`s and the ZIP files that contain data for the submissions to the transmitter. The transmitter uses these IDs to create the transmission manifest that enumerates each submission in the attachment ZIP file. The transmitter also needs to know the `ElectronicPostmarkTs` of each submission that is to be included in the transmission file after the `SubmissionID`. The `ElectronicPostmarkTs` is the date when the transmitter received the return from the taxpayer or ERO before sending it on to the MeF System.

Creating the Attachment ZIP File

The transmitter creates the attachment ZIP file by grouping the submission ZIP files into an attachment ZIP file. The attachment ZIP file must not be compressed; it merely groups the submission ZIP files. All submission ZIP files must reside in the root directory of the attachment ZIP file. The names of the submission ZIP files must be globally unique, and the name (without the extension) must match

the SubmissionID provided in the transmission manifest. For example, if the SubmissionID specified for a submission in the transmission manifest is 00123420132421234567, then a file named "00123420132421234567.zip" must be found in the attachment ZIP file. Note that this submission ZIP file is one of many submission ZIP files in the attachment ZIP file.

The transmitter then attaches this attachment ZIP file as the 2nd MIME part in the MIME-multi-part structure of the transmission file.

Sample Transmission File Structure

A sample transmission file is depicted below. The Cnt element in the TransmissionManifest indicates that this transmission file (actually the attachment ZIP file) contains two submissions. The attachment ZIP file is included as the 2nd part in this two-part MIME structure. The structure of the attachment ZIP file is not shown here.

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary="MIMEBoundary"; type="text/xml"
Content-Description: Transmission File containing one Submission.
X-eFileRoutingCode: MEF

--MIMEBoundary
Content-Type: text/xml
Content-Transfer-Encoding: 8bit
Content-Location: Envelope1120

<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns="http://www.irs.gov/efile"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:efile="http://www.irs.gov/efile"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/ ../message/SOAP.
xsd http://www.irs.gov/efile ../message/efileMessage.xsd">
<SOAP:Header>
   <IFATransmissionHeader>
         <TransmissionId>31070</TransmissionId>
         <TransmissionTs>2013-02-22T14:05:57+06:43</TransmissionTs>
                     <TransmitterDetail>
                           <ETIN>00111</ETIN>
                     </TransmitterDetail>
   </IFATransmissionHeader>
</SOAP:Header>
<SOAP:Body>
   <TransmissionManifest>
         <SubmissionDataList>
               <Cnt>2</Cnt>
               <SubmissionData>
               <SubmissionId>00777720130520001060</SubmissionId>
               <ElectronicPostmarkTs>2013-02-22T14:05:57+06:43</ElectronicPost-
markTs>
         </SubmissionData>
         <SubmissionData>
               <SubmissionId>00777720130520001070</SubmissionId>
               <ElectronicPostmarkTs>2013-02-22T14:05:57+06:43</ElectronicPost-
markTs>
         </SubmissionData>
   </SubmissionDataList>
   </TransmissionManifest>
</SOAP:Body>
```

```
</SOAP:Envelope>

--MIMEBoundary
Content-Type: application/octet-steam
Content-Transfer-Encoding: Binary
Content-Location: SubmissionZip

<<< 'attachment zip file' is attached here >>>

--MIMEBoundary--
```

What does the ERO do when composing a submission?

An ERO (Electronic Return Originator) is an authorized Modernized e-File provider who composes the submissions in the format required by the IRS using IRS-approved software. A submission consists of XML data and, optionally, binary attachments (PDF files). The files containing the XML data and any PDF files are compressed and grouped into a ZIP file, also referred to as submission ZIP file elsewhere in this document.

Each file packaged within this submission ZIP file is referred to as *ZIP Entry*. The submission ZIP file is provided to the transmitter who groups all submission ZIP files received from EROs into an attachment zip file, includes the attachment zip file in the transmission file, and sends the transmission file to the MeF System.

Note: The structure of the ZIP file is documented in the *ISS MEF State and Trading Partners ICD* and is reproduced here for reference.

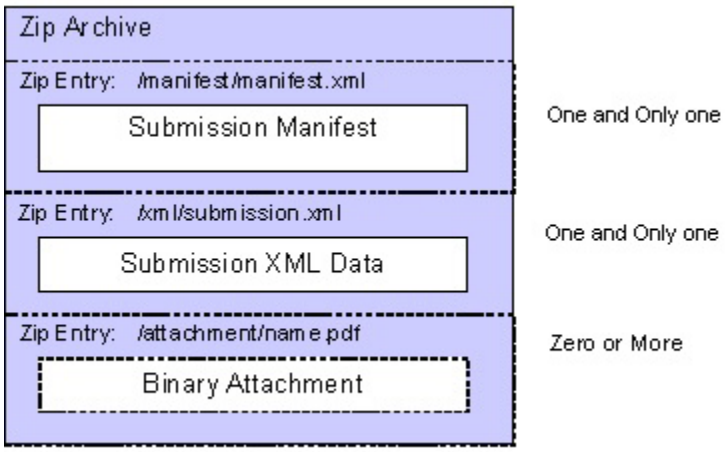
Composing an IRS Submission

An IRS submission consists of two XML files that contain data for the IRS submission, and zero or more PDF files as supporting documents for the IRS submission, as depicted in the figure below.

- The `manifest.xml` file contains information about the submission, for example, the ERO who composed the submission, tax year for which the IRS submission is being filed, etc.
- The manifest schema (IRS Submission Manifest and State Submission Manifest) is defined in eFileAttachments.xsd file which can be found in all Schema Packages under the common folder.
- The `submission.xml` file contains submission data in XML format.
- The PDF files, also referred to as *binary attachments*, are supporting documents in PDF format.

All of these files are packaged and compressed into a submission ZIP file.

Figure 5–1: Structure of an IRS Submission ZIP file



Notes:

The submission ZIP Archive containing the submission data must be named <SubmissionID.zip> where SubmissionID is the ID assigned to the submission by the ERO.

- The files must reside in the specified directories.
- The file containing metadata about the submission, `manifest.xml`, must be named as such and must reside in the `/manifest` directory in the submission ZIP file. The submission ZIP file is also referred to as the *ZIP Archive*.
- The file containing the submission data can have any name but, it must reside in the `/xml` directory, and must be the only file in that directory. The XML data in this file must conform to the published XML Schema for the submission type. The total number of characters of the SubmissionID is twenty. The first six digits contain the EFIN, the next four digits contain the processing year (the year that the return was sent to IRS, i.e. in year 2013, it would be 2013), the next three digits contain the julian date, and the last seven digits contain a sequence number to uniquely identify messages sent within a day with the given EFIN.
- The PDF files, if any, must reside in the /attachment directory in the ZIP file. They should be named in a way that describes the contents of the file (see Publication 4164 for recommendations on naming PDF files).There must be one Binary Attachment XML document for each PDF file present in the submission data file. The Binary Attachment XML document describes the PDF file.

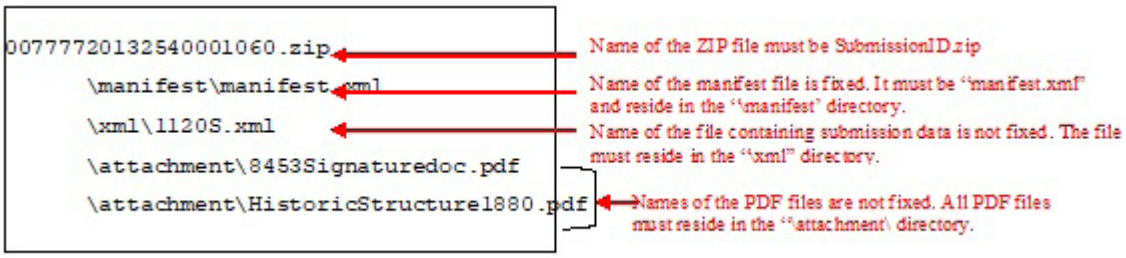
How are the supporting documents attached? on page 11 provides details on how references to the binary attachments (PDF files) are made from within the submission data.

Sample IRS Submission Structure

The structure of the ZIP file that contains a sample IRS 1120S submission is depicted below. The content of the XML files conform to the published Schemas, and the PDF files are binary files that are provided as supporting documents.

Refer to the published Schema for the IRS 1120S submission and the IRS manifest to determine contents of these files.

Figure 5–2: Sample IRS Submission Structure



Composing a State Submission

A state submission consists of two XML files that contain XML data for the state submission, zero or more PDF files as supporting documents for the state submission, and optionally, a copy of the IRS submission (both XML data and supporting PDF files), as depicted in the figure below.

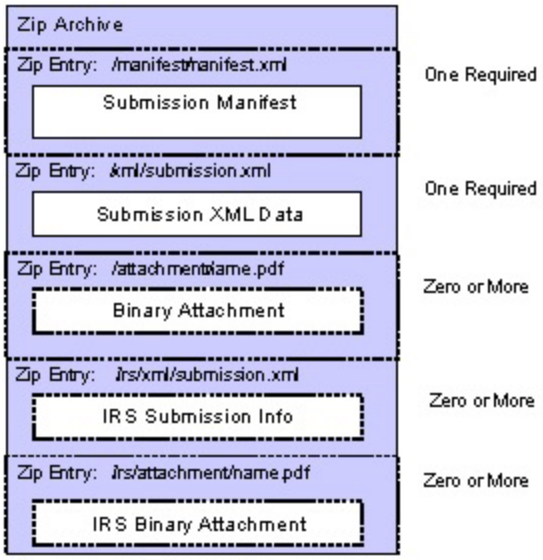
- The ZIP Entry named /manifest/`manifest.xml` contains information about the submission. For example, it contains the ERO who composed the submission, tax year for which the state submission is being filed, etc.

- The ZIP Entry named /xml/`submission.xml` contains state submission data in XML format, as prescribed by the states.
- The ZIP Entries in the /attachment directory contain PDF files, also referred to as *binary attachments*, that are supporting documents for the state submission in PDF format. Note that this directory can contain multiple PDF files.
- When a state requires that a state submission contain a copy of the IRS submission, then the ERO *must* include the file containing IRS submission data, and any PDF files that support the IRS submission in the ZIP file. The file containing IRS submission data (the XML document) must reside in the /irs/xml directory, and the files containing any PDF documents in support of the IRS submission must reside in the /irs/attachment directory.

Note: It is up to the states to specify the IRS data they need included with the state submissions. The structure allows the taxpayer to include the federal data if the state defines that it should be there. States will enforce that required elements of the state return will be present when they do their validation.

All of these files are packaged and compressed into a ZIP file.

Figure 5–3: Structure of a State Submission



The MeF System performs minimal validations on the state submission. MeF examines the manifest.xml file to determine the destination state, MeF validates that the state participates with MeF for that agency code, and makes the returns that have passed validation to that state for retrieval. It is up to the states to specify rules, if any, they wish to enforce on this structure.

How are the supporting documents attached?

The supporting documents can be XML documents whose structure has been published by the IRS or they can be non-XML documents (PDF files). Both types of supporting documents can be “attached” to either a line, or a form/schedule, or to the submission itself.

The supporting documents that are provided in the XML format are typically the following:

- Forms
- Schedules
- Statements

- Elections
- Attachments
- Payment records
- Notices

The structure of each supporting document that can be included in XML format is defined by the IRS in an XML Schema. When supporting documents are present in a submission, they may be attached to a location within submission data. In the MeF System, when a document is attached to an (XML) element then there is a reference from the element to the attached document.

Guidelines on Attaching XML Documents

- When a supporting XML document is attached to a line, there is a reference from the line (i.e., the element that represents the line) to the attached document.
- When a supporting XML document is attached to a form/schedule there is a reference from the form/schedule (i.e., the root element of the form/schedule) to the attached document.
- When a supporting XML document is attached to the submission there is *no* reference to it from anywhere in the submission data. In this case, it can also be said that the supporting XML document is "included" with the submission.

For example, when the IRS Payment Record (XML element name `"IRSCorporatePayment"`) is attached to the return, it is just included within the `ReturnData` element of the return; there is no reference to it from any line or from any form/schedule.

- Two attributes have been defined to indicate the attachment of a supporting document to an XML element.
  - The attribute named "referenceDocumentName", wherever it appears, is used to enumerate the document(s) that can be attached to the element.
  - The attribute named "referenceDocumentId" establishes a reference from the element where it appears, to the attached document(s).
- Each XML document must have a unique `documentId` within a submission.
- When the business rule is looking for supporting XML document attached to a line, form/schedule or return, and the supporting XML document is not attached there, the return would be rejected.

Sample XML Document Attached to a Line

This section provides an example of how the supporting document named "ItemizedOtherIncomeSchedule" is attached to line 10 on form 1120.

Note: There is a reference (using the "referenceDocumentId" attribute) from the element that represents line 10 (`OtherIncome`) to the `documentId` (`"DEPa1"`) of the attached XML document.

The schema for form 1120 (`IRS1120.xsd`) specifies that you can attach the document named "ItemizedOtherIncomeSchedule" to line 10. The name of the element that represents line 10 is `"OtherIncome"`.

Here is how the document is attached (create a reference) to the element that represents line 10:

```
...
<OtherIncome referenceDocumentId="DEPa1">8880</OtherIncome>
...
```

The referenceDocumentId attribute contains a reference to (or points to) a document whose documentId is "DEPa1". In this example it is the documentId of the "ItemizedOtherIncomeSchedule" document, as indicated below:

```
<ItemizedOtherIncomeSchedule documentId="DEPa1">
      <ItemizedIncome>
      ...
      </ItemizedIncome>
</ItemizedOtherIncomeSchedule>
```

Sample XML Document Attached to a Form

This section provides an example of how two supporting XML documents, "StockOwnershipForeignCorpStmt" and "DualConsolidatedLossesStmt" are attached to form 1120.

Note: There is a reference from the element that represents form 1120 (IRS1120) to the two document IDs ("DEPb1 DEPc1") of the attached XML documents. The list of document IDs is a space delimited list.

```
<IRS1120 documentId="DOC0001" referenceDocumentId="DEPb1 DEPc1" >
   ...
   ...
</IRS1120>
...
...
<!--  Supporting  document attached to Form 1120 -->
<StockOwnershipForeignCorpStmt documentId="DEPb1">
   ...
   ...
</StockOwnershipForeignCorpStmt>
...
...
<!--  Supporting  documents attached to Form 1120 -->
<DualConsolidatedLossesStmt documentId="DEPc1">
   <Statement>Explanation goes here</Statement>
   </DualConsolidatedLossesStmt>
```

Attaching Non-XML Documents to a (consolidated or sub-consolidated) return

To attach a PDF file, the following steps must be performed:

1. Create the PDF file.
2. Create a Binary Attachment XML document for each PDF file in the submission data that describes the PDF file. In a consolidated or sub-consolidated 1120 return, create the Attachment XML Document in the entity (consolidated or sub-consolidated, parent, subsidiary) where the PDF file is to be attached. If the PDF file is to be attached to the consolidated entity then create the Attachment XML document in the consolidated part of the return; if the PDF file is to be attached to the sub-consolidated entity then create the Attachment XML document in the sub-consolidated part of the return; if the PDF file is to be attached to the Parent entity, then create the Attachment XML document in the parent part of the return; if the PDF file is to be attached to a subsidiary return, then create the Attachment XML document in the subsidiary return. If the same PDF file is to be attached to more than one entity, say to the Consolidated entity, and one of subsidiaries, then create one Attachment XML document in the consolidated part of the return, and one Attachment XML document in the subsidiary. The documents will have the same content i.e., Document Type, Description, and AttachmentLocation, but will have distinct documentIDs.
3. If you need to attach the PDF file to a line, a form, or a schedule, then create a reference from the element that represents the line or the form or the schedule to the Binary Attachment XML document

that describes the PDF file. In a consolidated or sub-consolidated 1120 return, references are created to the Attachment XML document within the scope of the entity (consolidated, sub-consolidated, parent, subsidiary). You must not create references to Attachment XML documents created in one entity (i.e., in consolidated entity) from elements in another entity (i.e., the parent, or the subsidiary entity).

4. Include the PDF file in the Submission ZIP file that constitutes the submission

New Rule: If multiple Attachment XML documents are created for the same PDF file in a return, they must all have the same Document Type, and Description. Otherwise, the return will be rejected.

Note that multiple Attachment XML documents will need to be created if the same PDF file is to be attached to multiple entities within a consolidated or sub-consolidated 1120 return.

To create the Binary Attachment XML document, the ERO needs to know the name of the PDF file and a brief description of the file.

The ERO provides the name of the PDF file, including the extension, in the `AttachmentLocation` element and provides a brief description in the `Description` element of the Binary Attachment XML document. The schema for the Binary Attachment document is defined in the file named `BinaryAttachment.xsd`.

If the PDF file is to be attached to an element (a line, a form, or a schedule), the ERO creates a reference from the element to the Binary Attachment XML document. In a consolidated or sub-consolidated 1120 return, the references must be from elements in the entity to the Attachment XML documents within the entity (consolidated, sub-consolidated, parent, or subsidiary) in which the Attachment XML documents reside. You must not create references to Attachment XML documents created in one entity (i.e., in consolidated entity) from an element in another entity (i.e., the parent, or the subsidiary entity). There can be references from multiple locations (distinct elements) in the submission data to the same Binary Attachment XML document within the scope of the entity (consolidated, sub-consolidated, parent, subsidiary). This is similar to attaching the same PDF file at multiple locations in the submission data.

Note: The reference is created from the element to the Binary Attachment XML document, not to the PDF file. If no reference is created to the Binary Attachment XML document, then the PDF file is considered to be attached to the main form of the entity (consolidated, parent, subsidiary).

It is important to note that creating reference(s) to PDF files is needed only when the IRS specifies the conditions under which the reference must be created, and the reference locations within return data where the reference must exist.

The attribute binaryAttachmentCount of element `ReturnHeader` is used to indicate the number of PDF files included in the submission.

As a final step, the ERO packages the PDF files in the Submission ZIP file.

Sample Non-XML Document Attached to the Submission

This section illustrates how a non-XML document is attached to the submission. In the example below, a PDF document titled “8453 Signature Document” and file name is `8453SignatureDoc.pdf` is attached to a submission.

Attaching to a submission means including the PDF file in the submission ZIP Archive and creating a Binary Attachment XML document that describes it in the submission data, but having no reference to the Binary Attachment XML document from anywhere within the submission data.

The `Description` element of the Binary Attachment XML document contains title of the file, and the `AttachmentLocation` element contains the name of the file. The file itself is packaged along with other files that make up the submission in the submission ZIP file.

```
<Return>
   ...
<ReturnData>
   ...
<BinaryAttachment documentId="SignatureDoc8453" softwareId="00000000" software-
Version="String">
   <DocumentType>PDF</DocumentType>
   <Description>8453 Signature Document</Description>
   <AttachmentLocation>8453SignatureDoc.pdf</AttachmentLocation>
</BinaryAttachment>
   ...
</ReturnData>
</Return>
```

Sample Non-XML Document Attached to a Form

This section provides an example of how a file titled "Historic Structures circa 1880", named `HistoricStruct1880.pdf` is attached to form 1120.

If the same PDF file, `HistoricStruct1880.pdf`, is to be attached to other locations within the return, then a reference will be created (using the "referenceDocumentId" attribute) from those locations to this same Binary Attachment XML document `(documentId="PDFAttachment01"`, that describes the physical file). Only one instance of the physical PDF file is included in the submission ZIP file.

```
<Return>
   ...
<ReturnData>
   ...
<IRS1120 documentId="DOC0001" referenceDocumentId="PDFAttachment01">
         ...
         ...
</IRS1120>
         ...
         ...
<BinaryAttachment documentId="PDFAttachment01" softwareId="00000000" software-
Version="String">
              <DocumentType>PDF</DocumentType>
              <Description> Historic Structures circa 1880 </Description>
              <AttachmentLocation> HistoricStruct1880.pdf</AttachmentLocation>
</BinaryAttachment>
   ...
</ReturnData>
</Return>
```

Sample Non-XML Document Attached to the Submission, AND to one of the subsidiaries in a Consolidated 1120 return

This section provides an example of how a file titled “Merger Agreement 2013”, named `MergerAgreement2013.pdf` is attached to form 1120, and to a subsidiary return.

Since the same PDF file is to be attached to two entities (consolidated and subsidiary), two Attachment XML documents need to be created; one in the consolidated part of the return, and one in the subsidiary return. However, only one copy of the physical PDF file is included with in the submission ZIP file.

```
<Return>
   ...
<ReturnData>
   ...
<IRS1120 documentId=”DOC0001” referenceDocumentId=”PDFAttachment01”>
         ...
         ...
</IRS1120>
         ...
         ...
<BinaryAttachment documentId=”PDFAttachment01” softwareId=”00000000” software-
Version=”String”>
         <DocumentType>PDF</DocumentType>
         <Description> Merger Agreement 2013</Description>
         <AttachmentLocation> MergerAgreement2013.pdf </AttachmentLocation>
</BinaryAttachment>
   ...
</ReturnData>

   <SubsidiaryReturn>
         ...
   <ReturnData>
         ...
<IRS1120 documentId=”DOC0002”
   referenceDocumentId=”PDFAttachment02”>
   ...
   ...
</IRS1120>
   ...
   ...
<BinaryAttachment documentId=”PDFAttachment02”
softwareId=”00000000” softwareVersion=”String”>
   <DocumentType>PDF</DocumentType>
   <Description> Merger Agreement 2013</Description>
   <AttachmentLocation> MergerAgreement2013.pdf </AttachmentLocation>
</BinaryAttachment>
   </ReturnData>
   </SubsidiaryReturn>
</Return>
```

General Philosophy on Data Elements in the XML Schemas

In general, most data elements in the schemas for each form, schedule, and supporting document have been declared optional. Most of the required elements are in the schema for the return header. The schema for the return header contains identifying information about the entity filing the return, the officer responsible for the data in the return, the preparer, and the preparing firm. Hence there are very few data elements that are required.

This philosophy of keeping most data elements optional in the schemas is consistent with the way paper returns are filed, i.e., the taxpayer and return preparer have the responsibility to provide information as specified by IRS forms, instructions, and regulations.

How do I validate my return against XML Schemas?

Structure of a Return

Following is a high-level content model of a Corporate `Return` XML document.

```
   <?xml version="1.0" encoding="UTF-8"?>
<!-- Return1120.xsd for and 1120 Return -->
<Return returnVersion="2012v2.1">
   <!-- ReturnHeader.xsd for Return Header -->
   <ReturnHeader binaryAttachmentCount="0" subsidiaryReturnCount="0">
   </ReturnHeader>
   <!-- ReturnData1120.xsd for All Forms, schedules and XML Attachments -->
   <ReturnData documentCount="22">
         <!-- IRS1120.xsd, XML Schema for Form IRS 1120 -->
         <IRS1120 documentId="DOC0001">
               ...
               <!-- Form Fields -->
               <OtherIncome referenceDocumentId="DEPa1">8770</OtherIncome>
   ...
         </IRS1120>
   ...
   <!-- XML Schema for a Dependency, Itemized Other Income Schedule-->
         <ItemizedOtherIncomeSchedule documentId="DEPa1">
               ...
         </ItemizedOtherIncomeSchedule>
   ...
   </ReturnData>

</Return>
```

Here is a brief description of above content model:

- The `Return` is the root element in each return. The file named `Return<TaxType>.xsd` (where <TaxType> denotes the tax form that's being filed, e.g., 1120, 990, 8868, 1065, etc.) defines the content model of a return. The element `Return` is defined in this file.
- The `Return` is comprised of two mandatory child elements: `ReturnHeader` and `ReturnData`. In general, each return has its own `ReturnHeader` and `ReturnData` elements that are specific to that return. However, the ReturnHeader for an 1120 and an 1120S return is the same, and is defined in the `ReturnHeader1120x.xsd` file. Similarly, the ReturnHeader for a 990, 990-EZ, and a 990-PF return is also the same, and is defined in the `ReturnHeader99x.xsd` file. The ReturnHeader for a 1065 and a 1065-B return is the same, and is defined in the `ReturnHeader1065x.xsd`. The ReturnHeader for a 1040 form family of returns is the same, and is defined in the `ReturnHeader1040x.xsd`.

- A `ReturnHeader` contains common elements of a return, e.g., taxpayer name, EIN, address, etc.
- The `ReturnData` element contains data for the return, i.e., all return documents (forms, schedules and supporting materials) that make up the return.
- Each individual return document is either a form or schedule or supporting material and has a separate XML Schema defined for it.

Below are some XML resources regarding XML Schemas and software tools and parsers. (These resources are provided for information only—the IRS is not endorsing any product.)

- W3C XML Home Page: http://www.w3.org/XML/
- W3C XML Schema Home Page: http://www.w3.org/XML/Schema
- XML Spy: http://www.xmlspy.com/
- Apache Xerces parser toolkit: http://xml.apache.org/
- Microsoft MSDN library: http://www.microsoft.com/xml

Note: You may choose any third party parser toolkit or use your own.

Validating a Single Return Document

Important! When validating a return document or a whole return, it is not sufficient to only validate against the version of the schema used for creation. When submitted, the return is validated against the most current **minor** version. For example, if the return is created using v1.1 but the current version is v1.4, the return will be validated against v1.4 when it is submitted. Hence, when creating and validating return documents, validate against the version used for creation AND the most current version.

Note: The prior major version will never be validated against a new major version, i.e., a return created using version 1.x will never be validated against version 2.y.

Each return document (form, schedule, supporting material etc) has its own XML Schema defined. Therefore, you don't need to compose the whole return (XML document e.g., per `Return1120.xsd` for an 1120 return) in order to validate a single return document (a form, schedule, a supporting material). You can assemble data pertaining to that individual return document (a form, schedule, supporting material) and immediately validate it using its own XML Schema.

Here is an example of how to validate a single Return Document, "Itemized Other Income Schedule" XML document.

1. Assemble all the elements needed for the `ItemizedOtherIncomeSchedule.xsd` document as follows:

```
<ItemizedOtherIncomeSchedule documentId="ABC00010" >
   <ItemizedIncome>
         <CorporationName>
               . . .
         </Corporationname>
               . . .
         <IncomeType>Trade</IncomeType>
         <Amount>22330</Amount>
               . . .
         <PartnershipAmount>11120</PartnershipAmount>
   </ItemizedIncome>
</ItemizedOtherIncomeSchedule>
```

2. Add text and other required attributes shown in bold to the root element `ItemizedOtherIncomeSchedule` to make a stand-alone XML document and point to appropriate XML Schema file. Please realize that you need to specify the directory where the

ItemizedOtherIncomeSchedule.xsd file resides on your machine. It is specified as part of the value for the “xsi.schemaLocation” attribute.

Here is a sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<ItemizedOtherIncomeSchedule xmlns="http://www.irs.gov/efile" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.irs.gov/efile

d:/TY2012v11/CorporateIncomeTax/Corp1120/IRS1120/ItemizedOtherIncomeSchedule.
xsd" documentId="ABC00010">
   <ItemizedIncome>
         <CorporationName>
               . . .
         </Corporationname>
               .     .     .
         <IncomeType>Trade</IncomeType>
         <Amount>22330</Amount>
         .     .     .
         <PartnershipAmount>11120</PartnershipAmount>
   </ItemizedIncome>
</ItemizedOtherIncomeSchedule>
```

3. Now, validate the above XML document using your own XML parsers/validators.

Validating the Whole Return

Following are two ways you can validate a return against its xml schemas.

1. One-step approach

   Prepare all individual XML documents for the return (e.g., ReturnHeader, forms, schedules, and supporting materials, etc.), and then assemble and validate against the appropriate schema, for example, validate an 1120 return against the schema specified for the 1120 return in the `Return1120.xsd file`.

2. Multi-step approach

   There are several different ways you can do multi-step validation of the return. Here is one way:

   - Prepare a `ReturnHeader` document and validate against its schema, `ReturnHeader1120x.xsd`.
   - Prepare each individual return document (forms, schedules, and supporting materials) and validate against its schema.
   - Assemble all the return documents into `ReturnData` and validate against the ReturnData schema of the return e.g., `ReturnData1120.xsd` for an 1120 return data.
   - Assemble `ReturnHeader` and `ReturnData` into `Return` and validate against the appropriate schema e.g., `Return1120.xsd` for an 1120 return. This completes the whole return process validation.

Here is a variation on the multi-step approach:

- Prepare a `ReturnHeader` document. You may or may not validate it against its schema, `ReturnHeader1120x.xsd` before proceeding.
- Prepare each individual return document (forms, schedules, and supporting materials). You may or may not validate it against its schema before proceeding.
- Assemble all the return documents into `ReturnData`. You may or may not validate it against the schema for the Header.

- Assemble `ReturnHeader` and `ReturnData` into `Return` and validate against the appropriate schema, e.g., validate an 1120 return against the schema for an 1120 return defined in the `Return1120.xsd` file. This completes the whole return validation process.

Choose the procedure that is convenient and best fits your needs.

Validating the Transmission Envelope Including Contents

The transmission file is a MIME multi-part document that conforms to the "SOAP 1.1 with attachments" standard. It consists of two parts: the SOAP envelope and the SOAP attachment. The SOAP envelope maintains transmission level information, and the SOAP attachment contains the returns. MIME boundaries separate the two parts in the multi-part document.

The SOAP envelope consists of a SOAP header and a SOAP body. The SOAP header, also referred to as the *transmission header* in the MeF System, contains information about the transmitter and the transmission. The SOAP body, also referred to as the *transmission manifest*, contains a list of all returns included in the SOAP attachment.

Validation of the SOAP envelope (a.k.a., transmission envelope) including its contents consists of the following steps.

1. Validate the SOAP envelope XML instance against the SOAP schema, `SOAP.xsd`. The standard SOAP schema has been used without modification: `http://schemas.xmlsoap.org/soap/envelope/`.
2. Transmission header SOAP structure must consist of a single element, `TransmissionHeader`. Validate the `TransmissionHeader` element against the schema for the TransmissionHeader defined in the `efileMessageIFA.xsd` file.

   See Validating a Single Return Document on page 26 for details on how to validate an individual XML document (in this case, transmission Header XML document).
3. Transmission body SOAP structure must consist of a single element, `TransmissionManifest`. Validate the `TransmissionManifest` element against the schema for the TransmissionManifest defined in `efileMessageCommon.xsd`.

   See Validating a Single Return Document on page 26 for details on how to validate an individual XML document (in this case, transmission manifest XML document).

Sample Transmission File

Below is a sample transmission file that contains a SOAP envelope based on `SOAP.xsd`. Note that the contents for `SOAP:Header` and `SOAP:Body` are based on the `TransmissionHeaderType` defined in `efileMessageIFA.xsd`, and `TransmissionManifest` defined in `efileMessageCommon.xsd`.

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary="MIMEBoundary"; type="text/xml"
X-eFileRoutingCode: MEF

--MIMEBoundary
Content-Type: text/xml
Content-Transfer-Encoding: 8bit
Content-Location: Envelope1120

<?xml version="1.0" encoding="UTF-8"?>
<!— SOAP:Envelope must be validated against SOAP.xsd -->
<SOAP:Envelope xmlns="http://www.irs.gov/efile" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance" xmlns:SOAP="http://schemas.xmlsoap.org/soap/en-
velope/"  xmlns:efile="http://www.irs.gov/efile" xsi:schemaLocation="http://sche-
mas.xmlsoap.org/soap/envelope/ ../message/SOAP.xsd http://www.irs.gov/efile ../
message/efileMessageIFA.xsd">
```

```
<SOAP:Header>
   <!– TransmissionHeader must be validated against efileMessageIFA.xsd -->
   <IFATransmissionHeader>
         <TransmissionId>00002</TransmissionId>
         <TransmissionTs>2013-02-22T14:17:52+05:57</TransmissionTs>
         <TransmitterDetail>
               <ETIN>00112</ETIN>
         </TransmitterDetail>
   </IFATransmissionHeader>
</SOAP:Headader>
<SOAP:Body>
   <!– TransmissionManifest must be validated against efileMessageCommon.xsd -->
   <TransmissionManifest>
         <SubmissionDataList>
               <Cnt>1</Cnt>
               <SubmissionData>
                     <SubmissionId>00000120130520000001</SubmissionId>
                     <ElectronicPostmarkTs>2013-02-22T14:17:52+01:56</Electron
                     icPostmarkTs>
                     </SubmissionData>
               </SubmissionDataList>
         </TransmissionManifest>
   </SOAP:Body>
</SOAP:Envelope>

--MIMEBoundary
Content-Type: application/octet-steam
Content-Transfer-Encoding: Binary
Content-Location: SubmissionZip
... (attachment ZIP file containing submissions goes here)
--MIMEBoundary--
```

How are errors reported?

The structure and content of the transmission file and each return/extension included in the transmission file is validated to ensure that it conforms to the structure published by the IRS and the rules established by the IRS.

- The structure of the transmission file is checked for conformance to MIME standard, and the structure of the return/extension data is checked to ensure that it conforms to the XML Schemas published by the IRS.
- The return/extension data is validated against the IRS databases and checked for conformance to business rules published by the IRS.

When either structural violations are discovered or the data fails some business rules, errors are generated and reported back in an acknowledgement file.

The MIME headers in the transmission file are validated to ensure that their values (and their parameter values, if any) are set correctly. For example, the MIME Header `X-eFileRoutingCode` must be present and its value must be `'MEF' when using MeF`..

Similarly, the content of the transmission envelope is validated to ensure that it is structurally correct (per SOAP1.1) and each reference in the transmission manifest is found in the transmission file. If the transmission file structure violates some MIME rules and/or business rules published by the IRS, the whole transmission file is rejected. The returns/extensions included within the transmission file are NOT checked for errors in this case.

If the transmission file structure conforms to the MIME standard and IRS business rules, then each return/extension in the transmission file is validated to make sure that the data is structurally correct and conforms to the published business rules. Structural correctness means that the data conforms to the published XML Schemas. For example, all required elements are present and they conform to their established cardinality. Conforming to business rules means that the relationships among the data elements hold as stated in the published business rules. When either structural violations are discovered or the data fails some business rules, errors are generated and reported back in an acknowledgement file.

Three kinds of data validation are performed by the MeF System-

- Structural Validation – Conformance of XML data against the published schemas, and conformance of the file structure to the MIME standard
- Database Validation – Conformance of data with controls established in the IRS databases
- Business Rule Validation – Conformance of data to the established relationships among data elements

When a structural violation is discovered in the transmission file, the whole transmission file is rejected. For example, if the ID of the transmitter (i.e., the ETIN) is not included in the transmission header, the transmission file is rejected and the contents of the file are not examined.

When a structural violation is discovered in a return/extension, the return/extension is rejected. How are the structural errors reported? on page 34 provides details of the error structure reported for such errors.

When the data violates a business rule that checks data against an IRS database (e.g., the EFIN provided is not listed in the IRS database) or when the data violates a business rule that checks for data consistency (e.g., on Form 1120, Item D “Total Assets” must equal Schedule L line 15d), then the return/extension is rejected.

Business Rule Severity and Its Implications

Each business rule is assigned a severity by the IRS that determines if a return is accepted or rejected. There are two severity levels:

1. Reject And Stop
2. Reject

When a business rule with a severity of Reject And Stop is violated, the return/extension is rejected, an error is reported in the Acknowledgement file, and no further processing of the data is done. In this case, the acknowledgement file may not contain all errors that may exist in the return/extension data. The Acknowledgement file has a field named “Completed Validation” that will return a value of “false” when the return was rejected with a reject and stop severity.

When a business rule with a severity of Reject is violated, the return/extension is rejected, an error is reported in the acknowledgement file, and the system continues to find other errors, if any. In this case, the Acknowledgement file contains either all errors discovered by the system (if the number of errors is less than the IRS established threshold), or the number of errors equal to the threshold value. “Completed Validation” field in the Acknowledgement file will return a value of “true” when the return was rejected with a reject severity.

Error Structure

Each error generated contains the following information:

- Document ID – The document Id of the document which contains the violation, when available
- Path – (Xpath) to the data element causing the violation, when available
- Error Category – Errors are grouped into a small number of categories

- Error Message – Rule text or XML validator message
- Rule Number – Each rule is identified by a unique rule number.
- Severity – Reject And Stop or Reject
- Data value – Data value causing the violation, when appropriate

```
Errors are reported back to the sender in an acknowledgement file. The structure
of the error element in the acknowledgement file is shown below:
<Error>
   <DocumentID></DocumentID>
   <Xpath></Xpath>
   <ErrorCategory></ErrorCategory>
   <ErrorMessage></ErrorMessage>
   <RuleNumber></RuleNumber>
   <Severity></Severity>
   <DataValue></DataValue>
</Error>
```

The following example shows a section of the acknowledgement file when the following business rule is violated:

```
Rule Number:    F1120POL-014
Rule Text:      Form 1120POL, Line 23d must equal the total amount reported on
                Form 8913, Line 16(e).
Severity:        Reject
```

```
Acknowledgement File
<Acknowledgement xmlns="http://www.irs.gov/efile"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.irs.gov/efile../message/efileMessage.xsd">
   <SubmissionId>04025920130520017535</SubmissionId >
   <Errors errorCount=1 >
   <Error errorId="1">
         <DocumentID>Document12345</DocumentID>
         <Xpath>/Return/ReturnData/IRS1120POL/TelephoneTaxCredit</Xpath>
         <ErrorCategory>Math Error</ErrorCategory>
         <ErrorMessage> Form 1120POL, Line 23d must equal the total amount
         reported on Form 8913, Line 16(e).
         </ErrorMessage>
         <RuleNumber>F1120POL-014</RuleNumber>
         <Severity> Reject </Severity>
         <DataValue>1</DataValue>
   </Error>
</Errors>
   </Acknowledgement>
```

How are the structural errors reported?

When the transmission file violates MIME structure established by this document per MIME standard, or when the return/extension data does not conform to the XML Schemas, structural errors are generated. Structural errors are reported using a category named "XML Error".

- XML Error

  When the data violates the schema (Content Model) specification (e.g., missing a required element or multiple occurrences of an element when only one is allowed, e.g., Form 1120) or when the transmission file structure does not conform to the MIME structure established by this document.

The example below shows the structure of the error message when the return data does not conform to

the schema. Here an XML Error is generated because a required element, "City", was not provided as part of the preparer firm's address. Note that the xpath points to the next data element, "State" when "City" is not provided.

Business Rule

The return (XML documents) must conform to the version of the XML Schema indicated by the 'version' element in the return header.

This business rule covers all XML structural errors that are not specifically inspected by the Modernized e-File system. When violated, the following data is reported in the Acknowledgement file:

DocumentID: NA

Xpath:

/Return/ReturnHeader/PreparerFirm/PreparerFirmUSAddress/State

Error Category: XML Error

Error Message: The return (XML documents) must conform to the version of the XML Schema indicated by the 'version' element in the return header. *{//COMPLEX_E_UNEXPECTED_CONTENT// Unexpected Content "State"; expected "City".}*

Rule Number: X0000-005

Severity: Reject And Stop

Data Value:

The DocumentID is 'NA' since there is no DocumentID associated with the ReturnHeader.

Text in *italics* is provided by the XML Validator (currently the XML Validator by TIBCO is used).

How are the data errors reported?

When the transmission file contains invalid values for the MIME headers (or their parameters) or when the return/extension data violates one or more business rules, data errors are generated. Examples of errors of this kind include missing supporting information, violation of certain relationships among data elements, and mismatch of data against IRS databases. Errors of this nature are classified into the following categories:

- **Database Validation Error**: When some data provided in the return must be present in the IRS database, but is not (e.g., SoftwareID in the Return Header must have passed testing for the return type and tax year).
- **Missing Documen:t** When a return document is required per some business rule and is not included in the return. (For example: If Form 1120, Item A1, checkbox "Consolidated Return" is checked, then Form 851 must be attached.)
- **Multiple Documents:** When more than the allowable number of documents (per some business rule) is included in the return. (For example: If Form 1120, Schedule J, Line 4 has a non-zero value, then no more than one Form 4626 must be attached.)
- **Missing Data:** When data that should have been provided per some business rule is not provided. (For example: If Form 5471 is attached, then Schedule N (Form 1120) Line 4b must have a non-zero value.)
- **Incorrect Data:** When data is syntactically correct, but violates some business rule. (For example: The CUSIP Number cannot equal all zeros.)
- **Data Mismatch:** When the value in two fields should be the same (may be the source is different), but is not (e.g., Form 1120, Schedule J, Line 4 must equal Form 4626, Line 14).

- **Duplicate Condition:** When a return or transmission is a duplicate of a previously accepted return or transmission.
- **Math Error:** When Form 1120, Line 31[TotalTax] plus(+) Line 33[EstimatedTaxPenalty] is less than Line 32g[TotalPayments], then Line 32g minus (-) [Line 31 plus(+) Line 33] must equal Line 35[OverpaymentAmount].
- **Not On Time**: When a return/extension is received after the due date.
- **Unsupported:** When a submitted item (form, feature, format of something etc.) is sent to a location that does not accept it, or an unusual condition is encountered in data. (For example: The Transmission File must be free of virus. A virus was found in this file.)

The Error structure remains the same as described for schema validation. The Error Message in this case is the text of the business rule, as shown by the example below:

The following example illustrates the error structure when a Data Mismatch error is encountered:

Business rule: Form 1120, Schedule J, Line 4 must equal Form 4626, Line 14.

Values provided for these fields are as follows-

Form 1120, Schedule J, Line 4 = 12345,
Form 4426, Line 14 = 55555

When this business rule is violated, the following data is reported in the Acknowledgement file:

DocumentID: *DocumentID*
where *DocumentID* is the Id of Form 1120, Schedule J

Xpath:
/Return/ReturnData/IRS1120/IRS1120ScheduleJ/AlternativeMinimumTax

Error Category: Data Mismatch

Error Message: Form 1120, Schedule J, Line 4 must equal Form 4626, Line 14

Rule Number: F1120-060

Severity: Reject

Data Value: 12345