E-STANDARDS FOR SCHEMA DEVELOPMENT

Version 1.0

May 14, 2021

Contents

INTRODUCTION: WHAT HAS CHANGED

This is a complete rewrite of the E-Standards, the first since 2014. Here are our reasons for the rewrite, and what we are trying to accomplish.

- The official "TIGERS" – as we were known then - standards document dealt solely with schemas for Modernized eFile (MeF). This document will apply to all E-Standards schemas, including both the "Primary" aka "Master" schemas such as FSET and Levy, and the "Forms-based" schemas such as MeF.
- The tedious and page-consuming schema diagrams and element-by-element descriptions have largely been dropped, for several reasons:
    - This is intended to be a public document, available to anyone who is interested in the work of E-Standards. Most E-Standards schemas, especially those for MeF, are now secured in the FTA State Exchange System, and schema details require approved access.
    - Specialized documents for most of the schema sets, providing complete description, are in place elsewhere. This document focusses on the rules, processes, and recommendations for state schema development.
    - Schema sets are living, evolving things, with at least small tweaks annually. Standards, while also not carved in stone, are intended to be more stable, and should not become out of synch or obsolete with each new schema release.
    - To find the release level and location of all E-Standards schema sets, please see the matrix on the home page of the E-Standards website, www.statemef.com.
- This document combines both the E-Standards and the schema Best Practices in a single document. While the standards related to schema structure are comparatively firm, there is a softer line between design standards and best practices. This document is based on the E-Standards review practice, which has evolved with experience, and reflects the key issues that the review team looks for in evaluating a state schema set. The E-Standards review is designed as an educational process, to help states understand how the E-Standards and Best Practices help ensure that the state receives the data it needs.

OBJECTIVES

Much has been written over the years about the value of technical standards for the states. FTA has advocated formal standards since its involvement in the creation of the TXP addenda record for the National Association of Automated Clearing Houses (NACHA) ACH debit and credit transactions in the late 1980s. The process of consensus approval has proven valid through the years of ANSI Accredited Standards Committee X12 EDI, and since 2003 with XML schemas.

State standards for electronic commerce:

- Make it feasible for software suppliers and multistate trading partners to support nationwide programs by supporting some basic level of code reuse across jurisdictions.
- Provide frameworks for state development, so that states do not have to start from the ground up for each new program.
- Enable Fed/State and multistate programs when parties follow the same technologies.

- Provide a measure of stability to the tax ecosystem.

Specific objectives for the standards and best practices presented in this document are:

- To ensure that state return and other XML documents are technically valid and able to be processed.
- To enable coding by a variety of products utilized by states, software developers, and trading partners.
- To ensure that states receive the data that they need, as clearly, efficiently, and unambiguously as possible.
- To encourage optimal standardization among states. Note that this is not maximum standardization – the word “optimal” means that we strive for technical standardization but recognize the unique laws, regulations, and even terminology of each state must be acknowledged.

STANDARD SCHEMA SETS

E-Standards schema sets fall into two classes: “primary” schemas and “forms-based” schemas. Primary schemas, which include FSET for Withholding and Unemployment Insurance, Motor Fuel, and Tobacco schema sets, are fully developed to detail field level. The intent is to provide a complete subject matter schema needing minimal customization by the state. Primary schemas are good for tax types where either there is a strong model, such as the federal 94X program for Withholding, that most states followed at time of development, or where the states have already worked together to make the data uniform, as in the programs of the FTA Motor Fuel and Tobacco Uniformity Committees.

The first E-Standards XML program, FSET, is a primary schema, and it was originally believed that this would be the pattern for subsequent schema sets. However, this model did not work for MeF. The first MeF program developed was for Corporate tax, and the diversity of laws, regulations, terminology, and tax programs themselves proved impossible to codify in a single set of elements. Thus was born the “forms-based” schema set, where the XML framework and a number of basic functions are in fact fully coded, but a methodology is provided for states to code schemas for their own tax forms, and attach them to that framework schema.

However, **both primary and forms-based schema sets must follow the same basic set of rules**, which are the heart of E-Standards:

- States may not change the structure or order of the provided schemas, whether fully coded or framework.
- If an element is coded in the E-Standards schema as optional, then a state may omit that element in the state’s version of the schema (with some exceptions) if the state does not need or want to receive that element. This rule extends from simple data elements to high-level complex elements such as FinancialTransaction if there is no electronic payment program associated with the filing.
- However, states may **not** add any element, simple or complex, to the E-Standard schema set, without a vote of the full E-Standards group. (This of course is except for the state form schemas in a forms-based schema set.) For tax types supported by the FTA Uniformity Committees, this is a two-step process; the new data content must be approved by the appropriate Uniformity group, and the coding must then be approved by E-Standards.
- Both follow the same rules for schema set packaging and version naming.
- There are a few additional rules for state customization of the E-Standards schemas where allowed, which will be covered later in this document.

TECHNICAL E-STANDARDS

The following sections cover items that the E-Standards Review Team looks for when reviewing state schema packages, and which will cause the schema to be returned to the state for correction. Of course, there are exceptions to every rule. But unless there is a reasonable business justification, the schema set will not be approved until the corrected package is resubmitted.

XML CODING CONVENTIONS – E-STANDARDS FOR STATE XML

- All schemas within state schema packages must be valid XML. This seems obvious, but it is not uncommon for a state to make one last tweak before packaging and sending off their schema set, and to delete or add just one little pointy bracket, unleashing a torrent of cryptic error messages when the recipient tries to open the package. E-Standards are not intended to encompass all XML syntax rules, such as not starting an element name with a number or special character, so please use a validating parser to ensure that your XML is technically correct before you send it to the Review Team or share it with an Industry partner.
- While there are various stylistic choices that can be made when coding XML, E-Standards has selected a set of conventions to establish consistency among E-Standards schema sets and state schemas.
    - Elements are to be coded in "Upper Camel Case," where the words that make up the element name are concatenated and the first letter of each word is capitalized, such as "PaidPreparerInfoGroup."
    - Attributes are to be coded in "Lower Camel Case," where the first letter of the concatenation is NOT capitalized, such as "stateSchemaVersion."
    - Element/attribute names must not contain punctuation or special characters, and hyphens are discouraged.
- All primary data is to be coded using elements; attributes are reserved for metadata to clarify the associated elements, such as counts of repeating elements.
- All element/attribute names must be 30 characters or less. This rule is necessary in order to support the variety of technology products utilized by software developers and Industry partners. Some products parse XML data into database columns, where column names cannot exceed 30 characters.
- Element/attribute names must meaningfully label the data but may NOT simply consist of form and line numbers. This presents a maintenance nightmare for both states and software developers, when the following year a new field appears in the third position on the form and all line numbers below three are increased by one, so dozens of element names must be changed and the data that was associated with "Line09" last year is now "Line10." This was a Best Practice, but after years of experience, this is now an E-Standard and such names will not be approved by the Review Team.
- For ease of maintenance, all forms, schedules, submitted worksheets, and schema functional components are coded as separate sub-schemas, and tied together into a single submission schema through the use of the XML xs/xsd:include and xs/xsd:ref operators. Within XML, there are other technical ways to combine components, but as with the use of upper/lower camel case, the use of includes and refs has been adopted by both IRS and E-Standards, and no other

method will be approved. The one exception is that very small (six lines or less) worksheets may be hardcoded into the form to which they apply.

- States may not delete namespaces in the E-Standards schemas or add state namespaces either to the E-Standards schemas or to the state's form schemas, even if there is a state standard namespace. State schemas used in MeF must also use the IRS namespace as given in the E-Standards MeF schemas. These namespace rules are necessary because of the use of the include and ref operators to combine state schemas with both E-Standards and IRS schemas; not to deep dive into XML syntax, but any modifications will cause namespace conflicts which will prevent schemas from validating and are difficult to resolve.

PACKAGING AND VERSION CONTROL

For each E-Standards program, whether MeF or non-MeF, each state must build its own schema package. For MeF programs, this is necessary because each state must code schemas for its own forms, schedules, and worksheets. For non-MeF programs, even programs with complete primary schema sets, each state has some customization possible because the primary schema provides more options than any state realistically uses.

Each E-Standards schema package has a version/release number indicated by "Vaa.bb" where aa is the version number and bb is the release number. Each state package must also have a unique version/release number. A change in release number only (a "minor release") means that the release is backwards compatible; that is, any instance document (e.g. return) generated from the previous release will still validate against the new release. In general, a change in release number only represents the addition of optional new components. A change in version number (a "major release") means that the release is **not** backwards compatible; that is, an instance document generated from the previous release may not be valid against one or more schemas in the new release. Note that changing an element from optional to required in the schema is a major release – it is not backwards compatible. Deleting a formerly required element or changing the structure of a complexType are also generally not backwards compatible.

State schema packages are to be named XXAAAYYYYVaa.bb where:

- XX is the standard state abbreviation (3 or 4 characters for a city or regional jurisdiction)
- AAA is the name of the efile program. For programs with a single tax type, or a primary schema, this is the name of the E-Standards package, such as "FSET" or "MFET." For MeF, this is the name of the E-Standards MeF program name, such as "Business" or "Individual."
- For MeF, YYYY is the tax year. Non-income taxes may not be tax year dependent, so this is generally omitted for those program types.
- Vaa.bb is the version/release number, such as V1.2 for version 1, release 2 of the production schema package.
    - While a package is under initial development, the version number is zero, and only the release number is incremented, regardless of the nature of the change, starting with 0.1, 0.2, etc.
    - The first production-ready release, published on the SES, must be V1.0.
    - After that, the rules for numbering major and minor releases apply.

Side note: each state must decide how many versions/releases of a given schema package to support concurrently. Each situation is different, but considering the lead time needed for software developers to code to the new package, a state issuing a minor release may want to support both old and new releases for a specified period of time, while a major version change may need to be implemented more quickly and the old version retired. Communicating early and often with software developers when changes are in the pipeline will help with the lead time and allow them to adjust resources needed to implement the changes.

It has been noted that each E-Standards release includes a folder named "Documents" containing two Excel spreadsheet Change Logs – one for the E-Standards Common schemas, and one for the program-specific schemas. Each spreadsheet is a running list of changes, listed by version/release, giving the component changed, a very brief description of the change, who requested it, and whether it has been approved. It is highly recommended that states add a third Change Log for all version/release changes they have made. This is not an E-Standard, and schema packages will not be rejected if this is not done, but it is a request from Industry Partners, who say it is convenient for the actual code developers to have in the schema set they are coding to.

A critical best practice is for the states to follow versioning rules and update the version/release number with each new schema set, even if the change is small. The version/release number serves to inform the software developer whether or not they are working from the most current schema set.

BUILDING THE STATE RETURN SCHEMA

Here are basic standards for putting together the state's schema package.

- The root element of each return schema is named "ReturnState" regardless of the tax type. However, the .xsd schema file built for each state return must be named to match the type of return, such as "ReturnSC1040" or "ReturnSC1040EZ." This naming distinguishes the XML for the various types of filings within the tax type.

- Each ReturnState element has a single required attribute, stateSchemaVersion. This is a 50-character string containing the name of the schema version/release, as specified in the Packaging and Version Control section above. This attribute has a fixed value; states must change it for each new version and release. The incoming XML document (return) echoes this stateSchemaVersion back to the state, so that it can immediately determine the version/release of the document and validate and parse accordingly. Note that this does not limit the number of concurrent versions/releases that a state may choose to support; it just ensures that any document received is correctly identified.

- The ReturnDataState schema included in the E-Standards StateIndividualPackage is only a stub – a schema with a root element only. This stub ReturnDataState is identical for the StateIndividual and StateBusiness packages. The state must retain that root element, and may not change its name, but will use that root element to create state-specific filing structures by including schemas for state forms and schedules for the respective individual filing program(s). As with "ReturnState" above, the root name does not change, but the .xsd file name changes with each schema to reflect the return type for that file, such as "ReturnDataSC1040."

- The root element of the schemas for all forms are to begin with "Form" followed by the name of the form; root elements of schedules begin with "Sch" followed by the name of the schedule, and root elements of worksheets begin with "Wks" followed by the name of the worksheet. If any other common prefixes are desired, they can be proposed to E-Standards for a vote. In almost all cases, the .xsd schema file names should equal the root elements; ReturnState and ReturnDataState are the main exceptions.

- All forms are to be maintained as separate schemas. All schedules and worksheets containing more than just a few lines (six was recommended previously) are also to be maintained as separate schemas, and even very small schedules and worksheets **may** be maintained as separate schemas if the state desires for ease of reuse in multiple locations. This allows forms, schedules, and worksheets to be assembled as needed for various filing types, without having to be maintained multiple times. It also allows the state to modify the schema for a form or schedule as business requirements dictate, without having to modify the schemas of forms or schedules that have not changed.

- Schedules pertaining to a specific form are to be "attached" to the form through the use of an include statement and a reference element. Worksheets are to be attached to forms or schedules in the same manner. Extremely short schedules or worksheets may be included as complex types inline in the appropriate schemas; however, the use of the include statement and reference element is preferred where feasible. Schedules that apply to multiple forms may occur at the same level as the forms in the overall schema. In general, states are advised to keep the schema structure as "flat" as possible, by including forms and schedules for the filing at the same level.

- States are to create as many different versions of the ReturnState and ReturnDataState schemas as the state needs to specify different types of filings. To continue the example used above, the schema IndividualReturnXX1040.xsd might include the schema ReturnDataXX1040.xsd, while the schema IndividualReturnXX1040EZ.xsd would contain the schema ReturnDataXX1040EZ.xsd. This allows the state to control which forms and schedules can be used for each type of filing. For example, the ReturnDataXX1040.xsd schema for a state's long form may include a number of schedules that are not allowed to be filed with the state's XX1040EZ form and are therefore not included in the ReturnDataXX1040EZ.xsd schema.

EFILETYPES AND COMMON FOLDERS

Quick XML refresher: a "type" is a pre-formatted element that can be used to format other elements. A "simpleType" is a single field, such as an amount with 14 significant digits and two decimal places, which can then be used for fields such as Adjusted Gross Income. A "complexType" is a field made up of multiple subfields, such as an address with two address lines, city, state, and zip code. IRS coined the term "efileType" to refer to a collection of simple and complex types useful as "building blocks" in development of tax returns, and coded them into a schema named "efileTypes.xsd." Agreeing that this was a good idea, both to speed development and to encourage standardization, E-Standards created a similar schema "StateeFileTypes.xsd" including a number of types not included in the IRS schema, but felt to be useful to the states.

Within the schema package, IRS stored the efileTypes schema in a folder named "Common," since these types can be used in any XML tax program. E-Standards built an equivalent Common folder for efileTypes and StateeFileTypes. Other schemas that are used across programs are also stored in the Common folder, such as the core ReturnHeaderState, completed for each tax program, the FinancialTransaction schema for payments and refunds, and the BinaryAttachment schema for linking pdf attachments to the return XML. Although the efileTypes and Common folders were developed for MeF, E-Standards now utilizes these useful constructs across multiple E-Standards schema sets. States may NOT modify either the efileTypes.xsd or StateeFileTypes .xsd schemas.

Most non-MeF E-Standards packages contain a program-specific types schema containing building blocks unique to the subject matter, such as MotorFuelseFileTypes.xsd containing elements such as TerminalNameType. These schemas also cannot be modified independently by the states.

MeF States and other efiling jurisdictions are encouraged to build types unique to their state, that may be used across their own tax programs. States must create an "XXeFileTypes.xsd" schema to hold their

state types, where “XX” is the standard abbreviation for the jurisdiction, and store it in a new folder named XXCommon. At the same time, states are **not** to create their own XXefileTypes which are variations of established efileTypes or StateeFileTypes, if the originals can be used in the state schemas. For example, the standard AddressType has address lines that are 35 characters in length. States may not create a new type simply to expand the address line to 40 characters. Think about it: software developers are not going to programmatically keep track of “these states agree with 35 characters, these states have 40, and these have 45….**” e**specially since all of them support IRS returns that use the 35 character standard – they will code to 35 characters, since it will fit in a field built for 35, 40, or 45! If a state determines that a new efileType is in fact needed that is similar to an existing one but must be modified to meet the business needs of the state, the state can justify the new eFileType to the Review Team. The new type must be named XX….. where XX is the standard abbreviation for the jurisdiction and ….. is the name of the original type that is being modified, and it must be stored in the XXeFileTypes.xsd schema within the XXCommon folder.

Most non-MeF primary schema programs do not support efileTypes or Common folders unique to a state.

Non-MeF E-Standards schema sets may also have folders named CommonSO and ExtendedCommon. The schemas in these folders are needed for non-MeF programs because they do not use facilities provided by IRS for MeF.

CommonSO (“State Only”) contains versions of the efileTypes.xsd and StateeFileTypes.xsd schemas from which the IRS namespace has been removed. In MeF, because of the way IRS elements “wrap” and contain state schema elements, as well as the intent for efileTypes.xsd to be a copy of the exact IRS current schema, it is necessary for state MeF to utilize the IRS namespace. However, in non-MeF programs, this can cause issues for some states, so some (not all) non-MeF states, and some (again, not all) of these non-MeF schema packages use the CommonSO versions. It is E-Standards’ intent for these two schemas to be kept identical to the MeF versions except for the removal of the IRS namespace.

The ExtendedCommon folder contains schemas that either have been modified as requested by the states, for state use, or do not exist in MeF. In particular, the added schemas provide data and replace functionality of the State Manifest and other IRS enveloping, since the return transmission is directly to the states and does not flow through IRS. This data includes a transmission ID, timestamp, and transmitter identification.

CODING RULES and BEST PRACTICES

The following guidance applies to all E-Standards programs.

For the forms-based programs such as MeF and Sales Tax (FER), states must completely code their own forms and schedules. But even for primary schema packages, as previously noted, there are rules for tailoring the package; additionally, a state may have a business need to submit a modification request for a primary schema package, and the requested revised code must also follow the same coding rules.

The most basic rule is that the state must always pull down the most current version/release of the E-Standard package and work from it to apply customization. This can be cumbersome, especially if a state has significantly modified some permitted components such as FinancialTransaction. However, although all of the changes to an E-Standards release are given in the Excel Change Log, the Log does not replicate all of the coding detail, so important code changes may be missed if the state tries to apply the changes themselves to a previous release. If a state schema package is submitted for review, and components are found that are not current E-Standards, the package will NOT be approved.

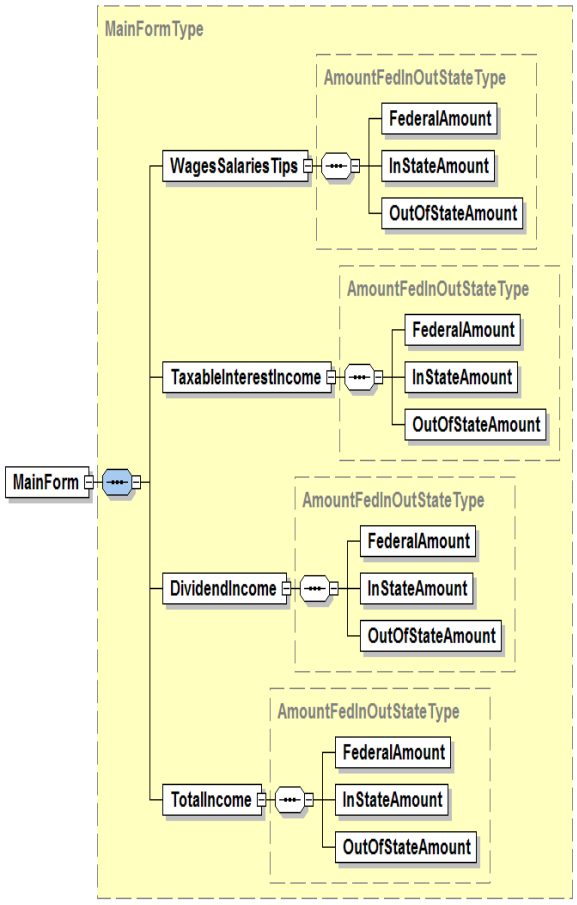
When deleting optional elements and components that the state does not wish to receive, always delete at the top level of the component, and do not leave a "stub" element. Note that efileTypes and StateeFileTypes may NOT be modified. Always delete the entire element of that type if it is optional within the E-Standards schema and the state does not wish to allow the data within the type to be submitted. Optional elements within the standard complexTypes must remain – they can only be deleted by creating a state-unique version of the type, which must be prefixed with the state identifier and stored in the state Common folder if, there is a valid business need to eliminate the elements.

One design choice for state coding is the grouping of elements. For example, rather than simply coding a sequence of elements that represent subtractions from income at the same level as other kinds of elements, normal practice is to create a parent element "SubtractionsFromIncome" and then to create a sequence of child elements for the subtraction categories. This has the benefit of making the schema content clearer for the human users, both state and industry, especially in the graphic view of the schema. There are no technical benefits, but there is valid benefit in human clarity. However, states are asked to avoid unnecessary levels, such as a parent element with only one child element. Also, states should avoid having a parent element and child element with the same element name – this is confusing both to human readers and to XML parsers.

One coding structure that will be questioned by the Review Team is that of a mandatory parent element where all of the child elements are optional. This falls under the objective of ensuring that agencies receive the data that they need to process the return. If all of the child elements are optional, then the agency may receive the electronic equivalent of a blank form – the form, but no data. The agency is encouraged to ask why the form/section is mandatory. There may well be a piece of information on the form – a child element – that is in fact required. If not, then make the parent element – the form or section of a form – optional, not mandatory. If the justification for making the parent mandatory is simply to ensure than the filer has seen and considered it, work with the software developers and/or NACTP directly to determine the best approach to accomplish this.

How to Code a Table Within a Form

Many tax forms include tables, which should be coded using complex types. First, use the column headings of the table to create a new complex type. For example, if the columns of the table are federal income, state income, and out of state income, then those become the three child elements of a new complex type. To create the table, the table rows become a sequence of elements, each of which is of the new complex type consisting of the columns.



If these columns only occur on one form, then this complex type - in the illustration above, "AmountFedInOutStateType" - can simply be coded as part of the form schema. However, if they are used in multiple tables or across multiple forms, then it is efficient to make this an XXefileType within the state's XXCommon folder.

Checkboxes and Booleans

The simplest way to think of this is that a checkbox has only one value: checked. A Boolean element has two values: yes or no. "CheckboxType" is not built into XML; it is an efileType created by IRS. The only permissible value for an element of CheckboxType is "X" to indicate that the box is checked. For this reason, any element of CheckboxType must always be optional – If a checkbox element is made mandatory, it means that the checkbox is always checked, whether or not the filer intended for it to be! The E-Standards schema review team will question any checkbox that is coded as mandatory.

While BooleanType is also an IRS efileType, it is based on an XML Boolean operator. The values for an element of BooleanType is either "0" or "1" or either "no" or "yes." If a state needs a definitive response to a question, either positive or negative, then a BooleanType should be used, rather than a CheckboxType.

Enumerations and Multiple Choice

Sometimes a filer needs to choose between alternatives, rather than simply indicating yes or no. There are several ways to do this. If only one alternative may be chosen, then an XML xs: or xsd:choice operator may be the most straightforward method, especially if the list of choices is fairly limited. The choices can be simple elements or elaborate complexes with multiple levels of child elements to include data needed for that choice.

If there are a large number of choices, or if the choices will change over time, then an enumerated list may be a better technique. Enumerated lists can be long if needed – for example, StateType is an enumerated list of all US states and territories. They are composed of literals that can be descriptive, such as "POST-REFUND FINANCIAL PRODUCT (REFUND TRANSFER)" in the MeF AuthenticationHeader. The literal value must be exact in the incoming return, but the software developer only needs to copy and paste the list into the code one time – the filer does not have to key it in. Enumerated lists used in one state schema may be coded in that schema. If an enumerated list is used in more than one schema, however, it should be stored in a single location and shared by the schemas for ease of maintenance. All MeF schema packages, and a few non-MeF packages, contain a schema called "StateEnumerations." This schema, one of the few which states are allowed – even encouraged – to modify, is "seeded" with two enumerated lists for two elements used in the ReturnHeaderState. One is for the type of state form being filed, and the other is for any special program for the filing, such as FreeFile or a multilingual program. States can feel free to store enumerated lists here, or else in their XXCommon.

But what if the filer needs to be able to select more than one condition? One approach is to make each alternative a separate checkbox or Boolean element, so that each choice that applies to the filer can be included in the return. Another choice can be to use an enumerated list, but to allow the element to which the enumerated list applies, to repeat an appropriate number of times, so that multiple enumerated values can be made part of the return.

FinancialTransaction

The FinancialTransaction schema contains elements that define how a balance due will be paid electronically, or how an agency is to pay out a refund electronically. Since a state may not support all electronic payment channels, it also serves to define which methods of payment the state will accept or utilize. As noted previously, states must delete entire complex types; it may not delete individual elements within a StateeFileType. For example, if a state does not wish to accept ACH credit payments for individual income tax, it can delete the entire ACHCreditInfo structure from FinancialTransaction in the state's individual schema package. Similarly, states not wanting to receive estimated payments with the return may delete that structure.

Note that the state may NOT modify the FinancialTransaction schema within the Common folder in a state schema package. The correct process is:

- Create a copy schema named XXFinancialTransaction.xsd stored in the XXCommon folder.
- Make the desired deletions to the XXFinancialTransaction schema.
- Do NOT change the name of the root element – it must remain “FinancialTransaction” so that the ref statement will still point to it.
- Change the xs: or xsd:include statement for FinancialTransaction in the Return schema to point to XXCommon/XXFinancialTransaction.xsd. This will make the ref statement point to XXFinancialTransaction.xsd.

A brief word about the IAT - which simply stands for “International ACH Transaction". This means that the payment or the refund deposit involves a financial institution outside of the United States. While the format was originally created to help financial institutions detect drug money, in the tax world the use is generally much more ordinary – a filer asking to send the refund home to assist family. The main difference is simply that because the foreign financial institution is not in the ACH network, ACH needs its name and address to process the transaction.

A number of states originally chose to delete the “FullIAT” needed to format an IAT transaction and rejected any payments or refunds that were labeled as “IsIATTransaction.” Actually, the agency may well be receiving IAT payments that were converted to ACH by an intermediary financial institution, with a minimal fee included. This is a state policy issue, not a technical one, and therefore not intended as either a standard or a best practice.

BinaryAttachments

Technical:

PDF is the only document format permitted in MeF; IRS will reject any other format. This was done for two reasons: first, the PDF format was less likely to harbor malware, compared to formats that could contain embedded macros. Secondly, the “P” stands for “Portable” – PDF is designed to be accessible and consistent across platforms. If an agency wanted to include spreadsheets or written documents, then what software would it support, would it be bound to updating releases each year, and how many versions would it support?

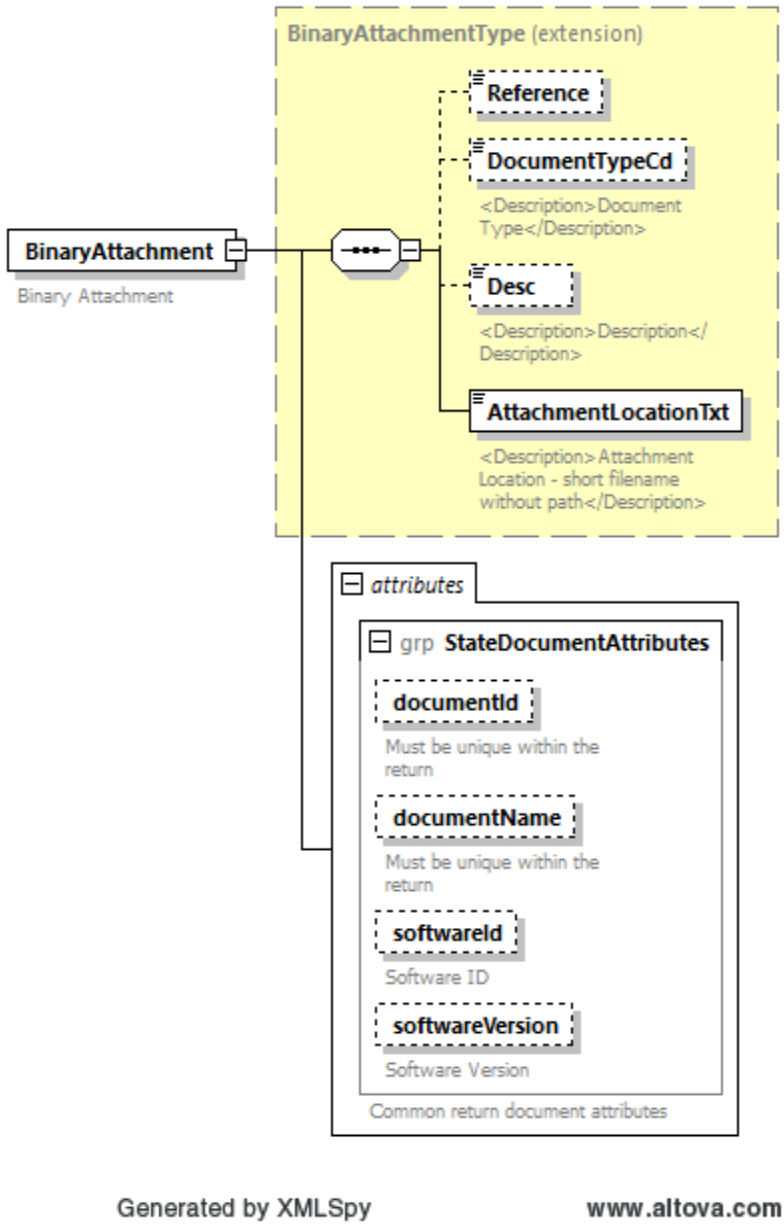
The E-Standards concerning the use of BinaryAttachments are these:

- In a state return, all uses of the BinaryAttachments must be tied to forms within ReturnDataState, that is, the pdf must refer either to a state form or to an element within a state form.
- Within the state XML, the attribute “referenceDocumentId” must be added within the state schema to the form root or to the element to which the PDF is to be attached. This attribute may repeat if necessary and should be optional unless an attachment is required for every use of that form or element.
- Within the BinaryAttachment schemas, the attribute “documentId” must equal the referenceDocumentId added to the state form schema within ReturnDataState; a separate occurrence of BinaryAttachment is required for each attached document. This allows the agency to match the attachment to the correct form or element.

- If used, the element “Reference” within BinaryAttachment should contain the form and line number where the referenceDocumentID is located.
- If used, the element “DocumentTypeCd is limited to “PDF.”
- Although the element is optional, Best Practice is to use the “Desc” to contain a text description of the attachment, such as “Signature Document” or “Heritage Credit Certification.”
- The one required element in BinaryAttachments schema, “AttachmentLocationTxt,” must contain the file name of the actual PDF file. It is simply of the format “filename.pdf” with no path needed. This is a required E-Standard, as it is the only link to the actual PDF. All PDF files attached to the return are zipped together and stored in the Attachment folder in the state return package, following the StateManifest and the state XML folder.

Example:

Here is the diagram of the BinaryAttachment schema.



In this example, say that a state has a return of the form:

<ReturnDataStateXX1040>

<FormAAAAA>……</FormAAAAA>

<FormBBBBB>……</FormBBBBB>

<FormCCCCC>……</FormCCCCC>

where FormCCCCC is a form for claiming credits, some of which require affidavits, available only as third-party documents. The element for FormCCCCC must be coded as:

<xsd:element name="FormCCCCC" attribute="stateDocumentAttributes">

where the element "documentID" within the attribute group equals "FormCCCCC" (or, if there are multiple pdf attachments, "FormCCCCC01").

Then in the BinaryAttachment schema, the "Reference" element must equal "FormCCCCC" (or "FormCCCCC01") to point back to the form to which it is attached. The "AttachmentLocationTxt" element then must equal the file name of the pdf attachment, such as "CreditAffidavit.pdf" as shown below:

<BinaryAttachment>

<Reference>FormCCCCC,/Reference>

<DocumentTypeCd>PDF</DocumentTypeCd>

<Desc>Affidavit_for_Credit</Desc>

<AttachmentLocationTxt>CreditAffidavit.pdf</AttachmentLocationTxt>

</BinaryAttachment>

Within the return zipped package, all BinaryAttachmentpdf documents are located within the Attachment folder following the returnsubmission XML.

- Manifest\
  - Manifest.xml (mainifest formatted to IRS standards)
- XML\
  - Submission.xml ("Submission" is replaced by the unique state submission id)
- Attachment\
  - CreditAffidavit.pdf (pdf files corresponding to binary attachments)
  - Attachment2.pdf (etc)

Best Practices:

- States are encouraged to limit the use of attachments. If attachments are needed to support information on the tax return, attempt to create an XML document to capture the data. Required copies of third-party documents, like a property tax bill, or an authorization letter from another agency to support a credit claim are examples of where PDF attachments may be useful. PDF format is the only type of attachment that is supported in MeF.
- The use of BinaryAttachments is especially problematic for individual/personal income taxes and should be avoided where possible. The main reason is that although the technology has become much easier to use and much less expensive in the last ten years, there are still many individuals – and even street corner preparers – who do not have the hardware and software to scan a document, convert it to PDF format, and upload it to attach to an electronic return. For the agency, a PDF attachment must still be viewed by a human, potentially delaying the processing of the return.
- Each binary attachment should be listed in a corresponding occurrence of ReturnState/BinaryAttachment. Elements in the BinaryAttachment eFileType let you enter the Form and line number the attachment relates to. There is a field to describe the attachment (e.g. property tax bill).
- In some cases, for example where there is more than one iteration of a credit form and each iteration has its own attachment, you may want a more formal link to the attachment so that you know which attachment goes with which iteration of the credit form.

ADDITIONAL GUIDANCE AND BEST PRACTICES

BUSINESS RULES AND ERROR VALIDATION

One ongoing question that must be decided as part of every schema development is that of which edits to hardcode as part of the schema, and which to handle on the back end as business rules.

The advantage of schema code is that it is a simple way to stop obvious errors, indicate permissible choices, and prevent bad returns from getting into your system. In theory, if the software developers are rigorous, no production return should ever have an XML validation error.

The downside of schema validation errors is that if there is a change necessary to one of the edits, a schema release would be necessary. For this reason, schema edits are best used for rules that are fairly stable for the filing season. If there is a possibility that a rule will be subject to later clarification or change, then it is best handled as a business rule.

Additionally, schema validation errors can be difficult to communicate the results to taxpayers when they do occur, because return developers often simply pass along the XML parser wording. Therefore, states should strive to not be too restrictive. Always doublecheck with your subject matter experts, for example, “are you positive this field can never be negative?” If there is any possibility, code the amount as USAmountType, not USNNAmountType.

Additionally, state agencies should, if possible, put the xpath for validation errors in the designated element in the Acknowledgement schema. Sometimes the parser used by the government agency returns different errors than the one used by software developers. In this case, it is helpful to have the

xpath so that the software developer can diagnose the issue. This xpath can also be used by software developers to associate a validation error to a solution for the actual user – filer or return preparer.

DOCUMENTING ELEMENTS WITHIN A SCHEMA

As an aid to the software developers, it is a best practice when implementing a new tax type program or adding a new form or schedule to an existing program, to include annotation that helps the developer map each XML element back to the form and line for context.

Include the form and line numbers information in the documentation child elements of each element whenever possible. Be aware, however, that this can cause the same issues which led the Review Team to prohibit the use of line numbers in element names – they can be obsolete the following year and be a maintenance headache. Best practice is to put a date on the annotation, to map back to one specific version of the form for that first year of implementation, and then not update it or carry it forward.

E-Standards Best Practices suggests:

<LineNumberForYYYY> where YYYY is the tax year of the form

<Description>

SIGNATURE REQUIREMENTS

Of course the necessity and manner of signing an electronic return is governed by state law or regulation. The only E-Standards are to utilize the SignatureOption elements in the ReturnHeaderState to indicate the signer and method of signing, and the signature PIN if appropriate. However, our industry partners offer the following as best practice:

> To make the filing paperless, the signature documents should be retained by practitioner and/or the taxpayer OR eliminate the signature document where allowed by law. States should adopt the IRS PINs as their signature alternative rather than requiring the taxpayer to attach a scanned Signature document. If not allowed by state law, then a state signature alternative should be put into place.

ACKNOWLEDGEMENTS

MeF was developed with a two-stage process of sending a Receipt to indicate successful transmission of a return, and an Acknowledgement to indicate whether the return was accepted by the agency. With advances in technology and many states moving to real-time processing, best practice is currently dropping the Receipt if the Acknowledgement can be returned within four hours. Most non-MeF E-Standards Efile programs have only the Acknowledgement.

It should also be noted that by general agreement of the Efile community:

a) An Acknowledgement indicating the return is accepted is considered proof of filing, while
b) It does not necessarily mean that the return is 100% correct and the return may still be adjusted.

SUMMARY – CONTACT THE E-STANDARDS TEAM EARLY AND OFTEN!

This document may be full of do's and don'ts, but the E-Standards review process really is not a "gotcha!" game. This document has attempted not simply to list those do's and don'ts, but to explain some of the thinking behind them and why they are there. To that aim, the E-Standards team is always ready to work with you, to better explain the standards and best practices, and to help you with your schema development.

Schema review is required for each new E-Standards program, but the review team is happy to review any schema with significant changes, such as those resulting from a major agency forms change, and to take a look at any situation where coding may be trickier, or where the state is not satisfied that what they have is clear or efficient.

Please don't wait to submit schemas until they are "completed" and the deployment date is rapidly approaching! Feel free to send schemas still in their early stages. The review team really does hate as much as the agency does when their developers have put in a significant work effort to be told that it is not acceptable! Additionally, please remember that the review team members all have "real" jobs in their own respective agencies or companies. The review team tries hard to complete all reviews within a two-week timeframe, but that may not always be possible.

Finally, as stated previously, over time best practices may evolve, and the E-Standards may need to be adjusted. There is an established process for this; please send an email to E-Standards, using the process described below, stating your requested change, the full resulting revised E-Standard or best practice, and the business case for the change. A valid business case is required. The Change Request will be reviewed for impact to other standards and best practices as well as to the agencies, and if acceptable, put to a vote of the full E-Standards membership.

CONTACTS:

E-Standards Staff:

Ryan Minnick	FTA Support
Steve Thimsen	E-Standards State Co-Chair
Sanford Schmidt	E-Standards Industry Co-Chair
Jacob Dubreuil	E-Standards Technical Support

E-Standards Review Submission:

If a jurisdiction wishes to submit either an E-Standards Change Request or a schema set for review, please contact the E-Standards team by sending an email to **Support@taxadmin.org**. The jurisdiction will be sent a unique link to use to upload their input to the secure State Exchange System. Every effort will be made to respond within ten working days. Please understand that most of the staff are acting in volunteer capacity and cannot guarantee this timeframe.