TIGERS
Best Practices
For
Modernized eFile
Forms Based Schemas
2008 V2.0

Table of Contents Page

INTRODUCTION

States are encouraged to follow these best practices when developing their state specific schemas. While many of the examples listed within this document relate to individual (personal) income tax, the guidance applies to both personal and business taxes in the forms based paradigm.

The best practices are divided into tiers. All states are strongly urged to incorporate all Tier One (T^1) practices into their schema(s) and/or business rules. Tier Two (T^2) practices should be incorporated as the state product(s) evolves.

ELEMENT/TAG NAMES: A.K.A. CHILD ELEMENT T^1

Each state must examine their tax forms and schedules to determine which forms will be included in the MeF project. In addition, a thorough analysis should be conducted to establish which line item(s) or element must be "captured" in order to process a state's return. Once the determination has been made, each element must be assigned a name. The name is commonly referred to as a tag name. States should use the established tag names from the TIGERS' library whenever possible.

HELPFUL HINT

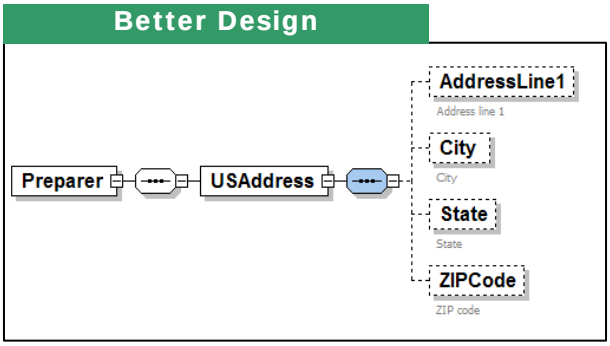
Try to avoid having the word "Type" at the end of an element name. By following this practice, it will be easier to distinguish between elements and eFileTypes.

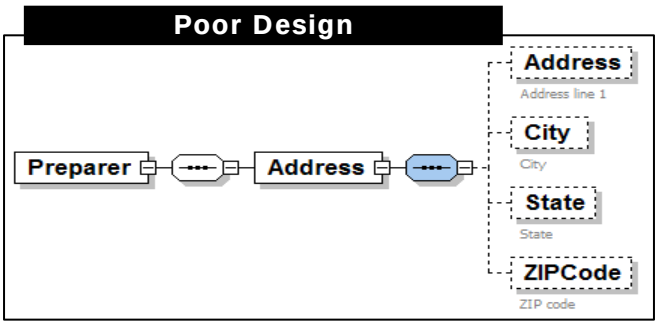
For example, entity type should be called TypeOfEntity rather than EntityType.

Once an element has been given a tag name don't change it unless absolutely necessary.

In some instances, the form field name changes during the annual tax form change process. The tag name may not change if the intent or meaning of the field has not changed.

When selecting a name for a parent element do not use the same name for the child element as it can generate some confusion. If the parent element only contains a child with the same name, then simply remove the unnecessary parent. If the parent contains several children, and one of the elements is named the same as the parent, consider more of what the parent element represents. A good example using address is shown below:





DEFINING DATA T^1

If a field is a duplicate of another field on the same return, do not create an element for the duplicate field. Create an element for the source field only. However, if the field is a duplicate of another field on a different return document, do create an element for it. (Example: 1040 line 37 AGI, 1040 Line 38 Amount from Line 37 – do not create

separate elements for the same data when it is repeated within a form).

If a field is conditionally duplicate (i.e., it will have a value if another field has a value) the element must be identified with a tag name.

For a fixed literal value, do not create an element for it (the literal) in the schema file. For example line 1(b) in Schedule C in Form 1120 has a fixed value of '70'. Hence no element is created for it in the schema file.

All literal values should be in uppercase characters. For example, when the instructions say to write "Form Sch K-1 F1120S" on the dotted line, the value for the element or the attribute should be "FORM SCH K-1 F1120S".

When only the line numbers change in a business rule, but underlying element names remain the same do not consider it to be rule deleted and added. It is just a "cosmetic" change (see example to the right).

TY 2007:
FederalTaxableIncome (line 1 of the return) + FederallyTaxExemptInterest (line 2 of the return) + OtherAdditionsAmt (line 3 of the return) must equal TotalIncome (line 4 of the return

TY 2008:
FederalTaxableIncome (line 1 of the return) + OtherAdditionsAmt (line 2 of the return) + FederallyTaxExemptInterest (line 3 of the return) must equal TotalIncome (line 4 of the return

Note that the line numbers changed in this example, but the business rule and element names did not.

TIGERS will provide standard schemas for information returns (i.e. W-2s & 1099s). If a state requires any other federal forms within the state return the federal form must add them to the state schemas.

GROUPING ELEMENTS A.K.A. PARENT T[1]

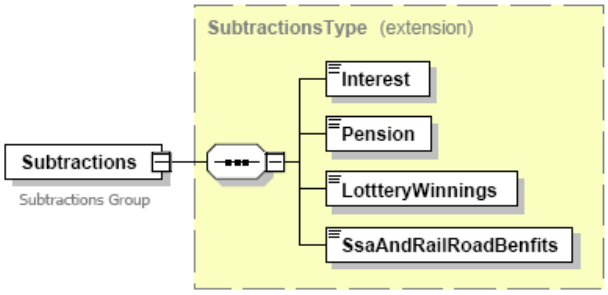
States are encouraged to group individual elements based on their grouping on the form. Below is a cross-section of the Arizona form and listing of the parent and child names for each element

PART C: Subtractions from Income

1	Interest on U.S. obligations such as U.S. savings bonds and treasury bills
2	Exclusion for federal, Arizona state or local government pensions (up to $2,500 per taxpayer)
3	Arizona state lottery winnings included as income on your federal return (up to $5,000 only)
4	U.S. Social Security or Railroad Retirement Act benefits included as income on your federal return (the taxable amount)

In this example, the parent name for this group is **Subtractions**. The individual element names (tagname) are:

Line 1 – Interest
Line 2 – Pension
Line 3 – LotteryWinnings
Line 4 - SsaAndRailRoadBenfits



STATE SPECIFIC DATA TYPES – SIMPLE AND COMPLEX T[1]

XML has the concept of "simple" and "complex" element definitions or "types." A simple type, such as a single amount with eighteen significant digits and two decimal places, stands alone. A complex type is made up of sub-elements, such as an address made up of street address, city, state, and zip code. Multi-layer complex types are used to represent the various table structures that often appear in tax forms and schedules.

What is an Efile Type?

The IRS has created an XML file of standardized, commonly used simple and complex type structures, and coined the term "efileTypes" to refer to these structures. TIGERS has augmented this file with a similar file of state efileTypes, that is, simple and complex XML structures that appear across state tax filings.

States should use the established EfileTypes from the IRS and TIGERS' library whenever possible without restrictions. If restrictions are needed, the state should create a new efile type specific to their needs. The files are: efileTypes.xsd – The set of efile types defined by the IRS and StateefileTypes.xsd – The set of efile types defined by TIGERS for use by states.

Your state's eFile Types will need to be stored in type library schema file(s) whose name(s) should be formatted as SSSxxxefileTypes.xsd where "SSS" is your jurisdiction code and "xxx" is the optional project ID (such as "Bus" or "Ind").

The following rules apply when naming your state-specific eFile types:

- The rules for naming elements also apply when naming eFile types
- Prefix your eFile type names with your jurisdiction code (ex. SSSDependentType where "SSS" is your jurisdiction code). This example protects you in the event that TIGERs or the IRS create their own eFile type called DependentType.
- All eFile type names should end with "Type"

States are encouraged to use complex types for data structures and may need to develop a state specific complex type if there is not an established one in the efiletype libraries. Complex types are recommended for the following situations:

- When data appears on multiple forms such as the voluntary contribution area on the resident and non-resident forms.
- When data needs to be captured in tabular format such as Allocation and Apportionment.
- When data is a repeating group such as multiple occurrences of a credit form.
- When separating data by taxpayer/spouse, in or out of state, full or part-year resident (see Illustrations one and two. Also view Example One and Two on Page seven and eight for the text version of the options).

All repeating groups should have attributes of minOccurs = "0" if optional and a maxOccurs = *nn* to "unbounded" if the element can appear an infinite number of times. maxOccurs may also be the number that equals the number of times the group can repeat.

Illustration One – *Option 1*

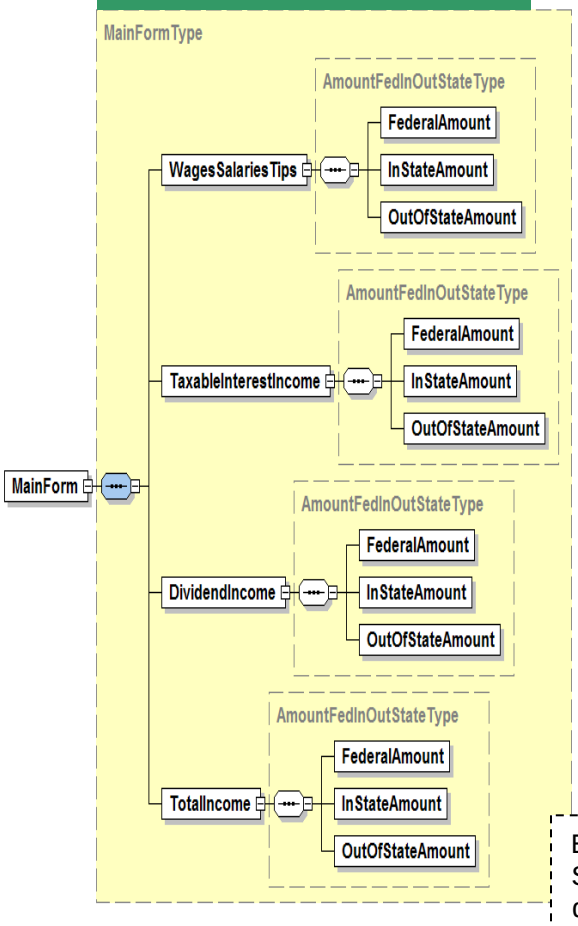
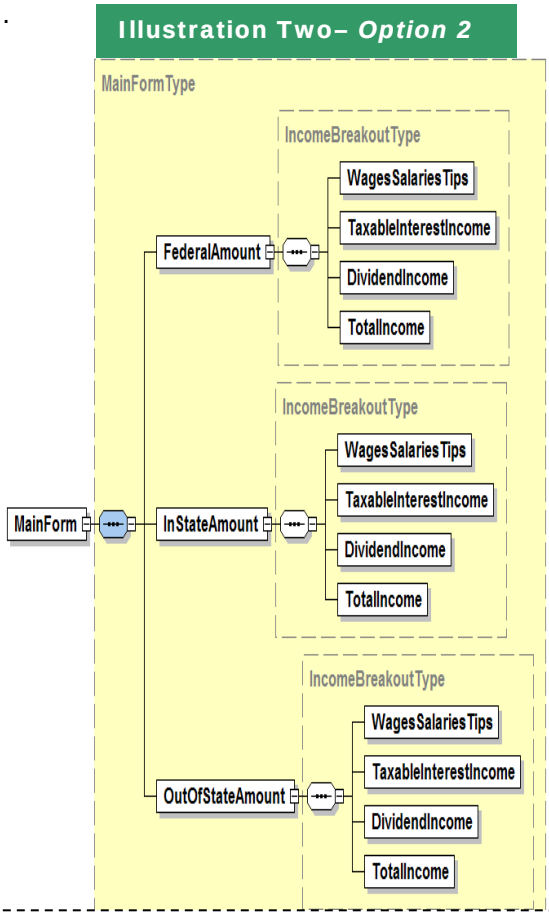


Illustration Two– *Option 2*



Either method is acceptable to industry, however, States are encouraged to choose rows or columns depending on how the data is calculated.

Example One – Option One

	Federal	In State	Out of State
1. Wages, salaries, tips, etc.	50000	30000	20000
2. Taxable interest income	2000	500	1500
3. Dividend income	100	25	75
4. Total income	52100	30525	21575

Create a complex type with child elements for each column:

```
<xsd:complexType name="AmountFedInOutStateType">
        <xsd:sequence>
            <xsd:element name="FederalAmount" type="USAmountType" />
            <xsd:element name="InStateAmount" type="USAmountType" />
            <xsd:element name="OutOfStateAmount" type="USAmountType"
            />
        </xsd:sequence>
</xsd:complexType>
```

Then create an element with that complex type for each line on the form:

```
<xsd:element name="MainForm" type="MainFormType" />
<xsd:complexType name="MainFormType">
        <xsd:sequence>
            <xsd:element name="WagesSalariesTips"
            type="AmountFedInOutStateType" />
            <xsd:element name="TaxableInterestIncome"
            type="AmountFedInOutStateType" />
            <xsd:element name="DividendIncome"
            type="AmountFedInOutStateType" />
            <xsd:element name="TotalIncome"
            type="AmountFedInOutStateType" />
        </xsd:sequence>
</xsd:complexType>
```

The instance document would look like this:

```
<MainForm>
        <WagesSalariesTips>
            <FederalAmount>50000</FederalAmount>
            <InStateAmount>30000</InStateAmount>
            <OutOfStateAmount>20000</OutOfStateAmount>
        </WagesSalariesTips>
        <TaxableInterestIncome>
            <FederalAmount>2000</FederalAmount>
            <InStateAmount>500</InStateAmount>
            <OutOfStateAmount>1500</OutOfStateAmount>
        </TaxableInterestIncome>
        <DividendIncome>
            <FederalAmount>100</FederalAmount>
            <InStateAmount>25</InStateAmount>
            <OutOfStateAmount>75</OutOfStateAmount>
        </DividendIncome>
        <TotalIncome>
            <FederalAmount>52100</FederalAmount>
            <InStateAmount>30525</InStateAmount>
            <OutOfStateAmount>21575</OutOfStateAmount>
        </TotalIncome>
</MainForm>
```

Example Two – Option Two

	Federal	In State	Out of State
1. Wages, salaries, tips, etc.	50000	30000	20000
2. Taxable interest income	2000	500	1500
3. Dividend income	100	25	75
4. Total income	52100	30525	21575

Create a complex type with child elements for each line of the form:

```
<xsd:complexType name="IncomeBreakoutType">
        <xsd:sequence>
            <xsd:element name="WagesSalariesTips" type="USAmountType"
            />
            <xsd:element name="TaxableInterestIncome"
            type="USAmountType" />
            <xsd:element name="DividendIncome" type="USAmountType" />
            <xsd:element name="TotalIncome" type="USAmountType" />
        </xsd:sequence>
</xsd:complexType>
```

Then create an element with that complex type for each column on the form:

```
<xsd:element name="MainForm" type="MainFormType" />
<xsd:complexType name="MainFormType">
        <xsd:sequence>
            <xsd:element name="FederalAmount"
            type="IncomeBreakoutType" />
            <xsd:element name="InStateAmount"
            type="IncomeBreakoutType" />
            <xsd:element name="OutOfStateAmount"
            type="IncomeBreakoutType" />
        </xsd:sequence>
</xsd:complexType>
```

The instance document would look like this:

```
<MainForm>
    <FederalAmount>
        <WagesSalariesTips>50000</WagesSalariesTips>
        <TaxableInterestIncome>2000</TaxableInterestIncome>
        <DividendIncome>100</DividendIncome>
        <TotalIncome>52100</TotalIncome>
    </FederalAmount>
    <InStateAmount>
        <WagesSalariesTips>30000</WagesSalariesTips>
        <TaxableInterestIncome>500</TaxableInterestIncome>
        <DividendIncome>25</DividendIncome>
        <TotalIncome>30525</TotalIncome>
    </InStateAmount>
    <OutOfStateAmount>
        <WagesSalariesTips>20000</WagesSalariesTips>
        <TaxableInterestIncome>1500</TaxableInterestIncome>
        <DividendIncome>75</DividendIncome>
        <TotalIncome>21575</TotalIncome>
    </OutOfStateAmount>
</MainForm>
```

SCHEMA DEVELOPMENT - BUSINESS RULES AND ERROR VALIDATIONS T^2

The great advantage of XML over legacy e-file is that much of the logic for accepting or rejecting a return can be in the schema. Having this logic in the schema makes it clear as to exactly what constitutes much of the acceptance or rejection criteria. The business rules should be:

- Atomic - that there is generally only one cause for the business rule
- Clearly worded
- Properly organized by category
- Rule number indicates form to which it applies
- Could not be prevented by a schema validation error
- Has an equivalent ERC in the legacy system. Do not add new rules for MeF which don't apply to legacy efile

The downside of validation errors is that there are nearly an infinite number of ways that they can occur and can be difficult to communicate to taxpayers when they do occur. Therefore, states should strive to not be too restrictive compared to legacy e-file – at least during the first few years using MeF.

Other practices to avoid in schemas are:

- Required parent tag with no required child tags
- Requiring information in the schema that is not on the form. (i.e. splitting out an address field that is one line on the form into city, state, and zip). This method depends solely on your form/system requirements.
- Too many required tags.
- States should refrain from adding new simple types that are not in EfileTypes.xsd or StateEfileTypes.xsd just to restrict the maximum length and/or allowable characters. Any restriction or unique characteristics for an element should be included in the business rule document rather than creating a new type.

For example, some agencies have implemented a non-negative US amount type for several fields and the re-released the schema when it was discovered that the tags in question may actually contain negative numbers in practice.

States should write business rules so they can be easily understood by taxpayers. Avoid use of tag names in the business rule text, use line numbers on the tax forms instead.

For schema validation errors, states are encouraged to use X0000-005 as the business rule number.

DOCUMENTING ELEMENTS WITHIN A SCHEMA T[1]

Include the following information in the documentation section of each element whenever possible (see example below):

TIGERS Best Practices suggests:
Description
LineNumber
ELFFieldNumber
ReferenceNumber (optional) for states referencing spreadsheet line numbers

The ELFFieldNumber is only used during the first year of transition from the legacy efile program to MeF. This will assist industry to associate the two programs.

DOCUMENTING ELEMENTS WITHIN A SCHEMA

```
<!-- State Tax -->
<xsd:element name="StateTaxAmount" type="USAmountType">
      <xsd:annotation>
      <xsd:documentation>
            < Description>
                  State tax (from Tax Table or Computation
                  Worksheet)
            </Description>
            LineNumber>32a</LineNumber>
            <ELFFieldNumber>0580</ELFFieldNumber>
            <ReferenceNumber>0780</ReferenceNumber>
      </xsd:documentation>
      </xsd:annotation>
</xsd:element>
```

HANDLING QUESTIONS AND LISTS OF OPTIONS T[1]

When your form has a question with a single corresponding checkbox you should represent that question as an element of type CheckboxType. Elements defined as CheckboxType should never be required. This is because "X" is the only valid value for a CheckboxType element. Examples: 1) Check here if 65 or older. 2) If apportionment percentage is from Form4B-1, check box.

Essentially, if the element is required, it is implied that the box must always be checked on your form. Another form of a checkbox is a Yes/No or True/False checkbox. While these would appear to be a CheckboxType, you want to apply the BooleanType to these fields, as this will allow you to require an entry (see further for BooleanType explanation) .

A Boolean data type's accepted values as (0,1, true and false).
The literals translate as:
(1, or True) = "Yes"
(0, or False) = "No"

For example: Were you a resident for the entire tax year? . . . Yes No

BooleanType should be used for lines on the form that have both Yes and No check boxes. An element of Boolean type should be required (both in the schema and in the developer's software) to force a deliberate decision by the user. This is a common scenario.

However, required Booleans should only be used if the schema follows the form/instruction logic. For example, if a line has both Yes and No, and it is not a required entry (eg the line is skipped in certain scenarios) then the field should not be required in the schema.

OVERFLOW STATEMENTS T[1]

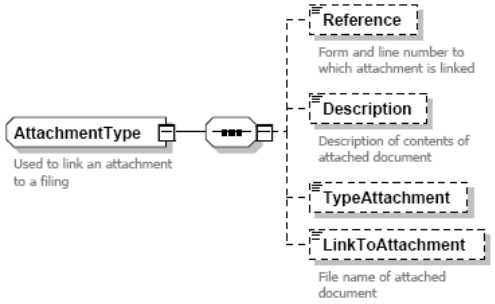
Tax forms often call for more information than can be shown on the tax form. For paper returns, and often in legacy e-file, the information that did not fit on the form was shown in an overflow statement. These statements are also referred to as itemization statements.

An example is dependent information on IRS Form 1040. There is only space to show 4 dependents on the form. If a taxpayer has 5 or more dependents, they must attach a statement to the paper return showing the additional dependents. For legacy e-file, the dependents are shown on a statement in electronic return because of the limited amount of sequence numbers reserved for dependent data.

Modernized e-file has no such restrictions. Therefore, overflow or itemization statements are unnecessary. Repeating elements can be used within the form schema to accommodate all the data. Eliminating statements from modernized e-file will allow for a straightforward and error free implementation which benefits taxpayers, government agencies, and software developers alike.

BINARY ATTACHMENTS T[1]

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 (see OVERFLOW STATEMENTS). Required copies of third party documents, like a property tax bill, or an authorization letter from another agency to support a credit claim (things that may have been mailed in under the legacy system) are examples of where PDF attachments may be useful. PDF format is the only type of attachment that is supported in MeF.



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 it's 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.

To make this kind of a link to a form or element in an XML document, create an attribute on the element with a name of referenceDocumentId and set it's value to the corresponding documented attribute in ReturnState/BinaryAttachment. (All documentId values must be unique within the submission.)

Example: linking a FORM to an attachment

```
ReturnState>
<ReturnHeaderState>....</ReturnHeaderState>
<ReturnDataState>
                <Form1040 referenceDocumentId="PDFAttach1">
                        ....
                </Form1040>
</ReturnDataState>
<BinaryAttachment documentId="PDFAttach1">
                <Reference>Form 1040</Reference>
                <DocumentType>PDF</DocumentType>
                <Description>Signature document for Form 1040</Description>
                <AttachmentLocation>Form1040Signature.pdf</AttachmentLocation>
</BinaryAttachment>
</ReturnState>
```

Example: linking a ELEMENT to an attachment

```
<ReturnState>
<ReturnHeaderState>....</ReturnHeaderState>
<ReturnDataState>
                <Form1040>
                        ....
                        <HeritageCreditAmt referenceDocumentId="PDFAttach1">
                12000
        </HeritageCreditAmt>
                        ....
                </Form1040>
</ReturnDataState>
<BinaryAttachment documentId="PDFAttach1">
                <Reference>1040 Line 12</Reference>
                <DocumentType>PDF</DocumentType>
                <Description>Heritage Credit Certification</Description>
                <AttachmentLocation>HeritageCreditCert.pdf</AttachmentLocation>
</BinaryAttachment>
</ReturnState>
```

SIGNATURE REQUIREMENTS T[2]

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 T[1]

Government agencies should put the xpath for validation errors in the acknowledgement. Sometimes the parser used by the government agency returns different errors than the one used by software vendors. In this case, it is helpful to have the xpath so we can diagnose the issue. This xpath can also be used by vendors to associate a validation error to a solution for the user.