E-Filing Schema Developer’s Guide

Introduction

The Department of Income Tax (Systems) has introduced a scheme to facilitate filing of returns online by E-Return Intermediaries. Details of the scheme and the procedure to be followed for enrolment by the E-Return Intermediaries are available on the web site of the Department athttp://www.incometaxindia.gov.in and at www.tin-nsdl.com.
After passing the Data Connectivity test and registering on the E-Filing site, E-Return Intermediaries will be able to file tax returns through the internet.
The Department has proposed a mechanism for data exchange between the Intermediaries and its systems that will facilitate offline creation of returns (without being connected to the Internet or the E-Filing website) in XML format and uploading of the same to the website when connected to the Internet. The Department has therefore designed an XML schema which defines the data structures and data types for each Return form.
The advantages of this mechanism are:
1. The E-Return Intermediary can prepare Returns in XML format offline (using a client application of its choice) without needing an Internet connection for long periods.
2. It is compatible with multiple platforms and operating systems.
3. It follows standard data exchange protocols and mechanisms for data transfer over the Internet.
4. It lends itself to easy interpretation at the server end.
5. The Intermediaries can develop client applications (on any platform and operating system) to generate the XML Return files and embed the validations to be carried out for data Exchange within the application.
The XML schema for each form contains elements which correspond to the various columns /form fields being filled in the Paper Return. The XML schema describes the data types that can be used against each element and the validations being carried out to check the accuracy of the data submitted by the Intermediaries.
It is important to understand the contents of the schema in order to be able to transmit Return files to the E-Filing web site that are in accordance with the data standards published.
The purpose of this document is to explain the various data types used in the Schema, the validations and constraints imposed while submitting the Return information.
This document is a technical guide targetted at Software Developers who wish to develop a client application to be used by Intermediaries to generate XML files for each Form Type.

Overview of E-Filing process

The details of the E-Filing schema for E-Return intermediaries can be downloaded from the website. The details provided in this document are only a supplement and capture the salient points.
1. Individuals and Firms who wish to enrol themselves as E-return intermediary need to apply online at the web site of NSDL, www.tin-nsdl.com.
2. The applicants have to submit the relevant documents and the fees prescribed by the Registar (NSDL).
3. After scrutiny of the same, NSDL will allot a temporary User ID and password to the applicant to carry out the data connectivity test.
4. The applicant shall logon the site http://incometaxindiaefiling.gov.in/ with the user ID and password allotted by NSDL and carry out the test

Details of the Data Connectivity Test

The applicant will upload XML files for each Form type mentioned. He has to

upload one XML file for each individual form. He will be given 20 attempts to pass the test. The uploaded XML files will be validated against the schema published by the Department and each upload (successful or otherwise) will be considered as an attempt.
The pre-requisites to carry out the test are:
1. Receipt of User ID and Password from NSDL
2. Internet connection (Broadband would be preferred)
3. Download of schema to prepare the returns in XML format

Intermediary client application to prepare the Returns in XML format

Do I need a client application to prepare the XML files in specified format before I attempt the data connectivity test?

No. Theoretically it is possible to prepare an XML file of specified format using an ordinary editor like Notepad or Wordpad. Sample XML files for the various form types are available in the Appendix section of this document. It is possible to prepare such files using Notepad and save them with .XML extension.
However in practice, preparing such return files in XML using Notepad and changing the data within the XML tags for each assessee manually without any application will prove tedious and cumbersome.
It is therefore recommended that the intermediary uses a client application that can help to prepare the XML files.
E-filing Portal Schema Developer’s Guide
The client application above should have the minimum following features:
1. Easy to use with a neat interface
2. Work on any platform/operating system
3. Should launch as an Executable from any desktop running standard OS like Windows 2000 Professional, Windows XP etc
4. The system should capture Return details as specified in the paper form fields.
5. Save the output in XML format which confirms to the schema published by the Department
6. Perform client side validations for date and other formats as specified by the Department by validating against the schema. This will minimize the rejections during upload to the web server if validations are already carried out at the client end itself.

Do I need to enter “real” data of an assessee in the XML file while attempting the data connectivity test?

No. Applicants can use dummy data in the XML file while attempting the test. However the Data types used in the appropriate elements have to be as per schema specification. For example, in the <PAN> element tag, dummy data can be entered and it need not correspond to an actual PAN issued by the department. However, the data type and pattern match specified in the schema should be followed. That is the first five digits should be alphabets in capital letters, the next 4 digits should be Integer and the last should be an alphabet in capital letter. The following PAN AAAPN9999N used in the XML for data connectivity test will be validated against the schema. However when actual returns data is processed after passing the test and registration at the site, such data will be verified against the database to verify the genuineness of the return information and the assessee.
The intermediary has been provided with the flexibility to design the application to suit the level of sophistication required by him and is based on feedback received during the pilot operation of the scheme last year. The application can be designed to just feed in data for each return and provide an XML as output or it can also be designed to provide other value added features like tax computation etc.
The mandatory output in either of the two cases above is a XML file containing multiple records (one for each assessee) as per the Schema published by the Department.
After passing the data connectivity test, the applicant can register online at the site and create a permanent user ID and password. He will also be allotted an ERIIN code which should be used in all future correspondence with the Department.

Once registered, the Intermediary can upload actual return in XML format. The website will issue a token number for each XML file uploaded. The Intermediary can check the status of the upload using this token number against the E-Filing Re menu option. The system will allow him to print a provisional acknowledgement receipt for each return.
This acknowledgement number and date will be uploaded in the E-Filing website against the acknowledgement number and receipt issued earlier to the Intermediary. This will complete the E-Filing cycle.
The following sections of the document are technical in nature and can be used as a reference by Software Developers while designing the Client application for Intermediaries.

Why XML Schema
The various reasons for choosing the XML schema as a standard for data exchange are explained below:
One of the greatest strengths of XML Schemas is the support for data types.
With the support for data types:

- It is easier to describe permissible document content
- It is easier to validate the correctness of data
- It is easier to work with data from a database
- It is easier to define data facets (restrictions on data)
- It is easier to define data patterns (data formats)
- It is easier to convert data between different data types

XML Schemas use XML Syntax
Another feature of the XML Schemas is that they are written in XML.
Because XML Schemas are written in XML:

- You don't have to learn another language
- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schema with the XML DOM
- You can transform your Schema with XSLT

XML Schemas secure data communication
When data is sent from a sender to a receiver it is essential that both parts have the same "expectations" about the content.
With XML Schemas, the sender can describe the data in a way that the receiver will understand the content.
A date like: "03-11-2004" will, in some countries, be interpreted as 3rd November and in other
countries as 11th March, but an XML element with a data type like:
<date type="date">2004-03-11</date> ensures a mutual understanding of the content because the XML data type date requires the format YYYY-MM-DD.

XML Schemas are Extensible
XML Schemas are extensible, just like XML, because they are written in XML.
With an extensible Schema definition you can:

- Reuse your Schema in other Schemas
- Create your own data types derived from standard types
- Reference multiple schemas from the same document

Well-Formed XML can be easily interpreted
A well-formed XML document is a document that conforms to the XML syntax rules:

- must begin with the XML declaration
- must have one unique root element
- all start tags must match end-tags
- XML tags are case sensitive
- all elements must be closed
- all elements must be properly nested
- all attribute values must be quoted
- XML entities must be used for special characters

Data types used in ITR Forms

Built-in Datatypes

Built-in datatypes, which are defined in the W3C XML Schema Datatype Specification, must be supported by all W3C XML Schema-compliant parsers. There are two classifications of built-in datatypes: primitive and derived. The differences between the two have little relevance for the user, but we will examine them here to demonstrate the mechanics and utility of datatype generation.

Built-in Primitive Datatypes used in ITR forms

Primitive datatypes are indivisible. They are not defined in terms of other datatypes; they exist independently. For example, decimal is a well-defined mathematical concept that cannot be defined in terms of any other datatypes. There are the 19 built-in primitive datatypes supported by the XML Schema Datatypes Specification. The highlighted ones are extensively used in the Schema published by the Department of Income Tax.

- **string**
- **boolean**
- decimal
- **float**
- double
- duration
- dateTime
- time
- **date**
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- hexBinary
- base64Binary
- anyURI
- QName
- NOTATION

Some of the important data types are described in detail.

string	A sequence of Unicodecharacters.	"This is a sample string'
boolean	One of of either true (1), or false (0).	true
float	A single precision 32-bit floating point type	-1E4, 2442, 342.34, 0, INF, NaN
double	A double precision 64-bit floating point type	-1E4, 2442, 342.34, 0, INF, NaN
decimal	A decimal number of arbitrary precision	3.14158265358979323846264338327950288419716939937510...

User-Derived Datatypes in ITR Forms

User-derived datatypes are the ones specified by the user in an XML Schema Definition, and are created by restriction, list, or union. The XML Schema construct <simpleType> is used to create user-derived datatypes. Such a datatype can be named if one wants to re-use it or can be anonymous if it is to be used only once.
There is some confusion because the specification currently categorizes list and union as userderived datatypes. They should rather be categorized as user-defined datatypes for clarity. This confusion may be addressed in the next version of XML Schema.

User-Derived Datatype by Restriction

Every built-in datatype has a set of allowed constraining facets, which can be used to constrain or restrict that datatype, leading to the creation of a new datatype categorized as a user-derived datatype. A constraining facet is an optional property that can be applied to a datatype to constrain its "value space." Constraining the "value space" consequently constrains the "lexical space." The value space of a datatype can only be restricted and not extended. The XML Schema construct <restriction> is used to create user-derived datatypes by restricting an existing datatype with the allowed constraining facets. For example, a string of length 3 can be expressed as:

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema
xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mydatatypes.edu"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
        <element name="Currency">
                <simpleType>
                        <restriction base="string">
                                <length value="3" />
                        </restriction>
                </simpleType>
        </element>
</schema>
```

In the above example, an anonymous user-derived datatype—the base datatype being string—is defined along with the constraining facet, length. The same example can be written using a named user-derived datatype for re-usability:

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mydatatypes.edu"
xmlns:tns="http://mydatatypes.edu"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
        <element name="Currency" type="tns:currency_type" />
        <element name="MoreCurrency" type="tns:currency_type" />
        <simpleType name="currency_type">
                <restriction base="string">
                        <length value="3" />
                </restriction>
        </simpleType>
</schema>
```

Following are the 12 constraining facets in XML Schema, which can be used to create a userderived datatype from other available built-in datatypes. The constraining facets might change however depending on the base datatype. The highlighted ones are extensively used in ITR schema definition:

- **length**
- **minLength**
- **maxLength**
- **pattern**
- **enumeration**
- whiteSpace
- maxInclusive
- maxExclusive
- minExclusive
- minInclusive
- **totalDigits**
- fractionDigits

Design Details of the ITR Schema

Complex Datatypes in ITR Schema

At the top of the schema definition a complex type is defined which encapsulates all the return form types:

```
<xs:element name="ITR1" id="ITR-1">
<xs:complexType>
<xs:sequence>
<xs:element ref="ITRForm:CreationInfo"/>
<xs:element ref="ITRForm:Form_ITR1" id="Heading"/>
<xs:element ref="ITRForm:PersonalInfo" id="Section1"/>
<xs:element ref="ITRForm:FilingStatus" id="Section2"/>
<xs:element ref="ITRForm:ITR1_IncomeDeductions" id="Section3"/>
<xs:element ref="ITRForm:ITR1_TaxComputation" id="Section4"/>
<xs:element ref="ITRForm:TaxPaid" id="Section5"/>
<xs:element ref="ITRForm:Refund" id="Section6" />
<xs:element ref="ITRForm:Schedule80G" minOccurs="0"/>
<xs:element ref="ITRForm:TDSonSalaries" id="Section21" minOccurs="0"/>
<xs:element ref="ITRForm:TDSonOthThanSals" id="Section22" minOccurs="0"/>
<xs:element ref="ITRForm:TaxPayments" id="Section23" minOccurs="0"/>
<xs:element name="TaxExmpIntInc" default="0" id="Section25" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:unsignedLong">
<xs:totalDigits value="4"/>
<xs:maxInclusive value="5000" />
<xs:minInclusive value="0" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element ref="ITRForm:Verification" id="Section26"/>
<xs:element ref="ITRForm:TaxReturnPreparer" id="Section27" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

Each of the elements inside the complex type “ITR1” is a reference to another Complex type. Each of this inner complex type element defines xml schema for data entry to CreationInfo, Form_ITR1, PersonalInfo etc.

Complex data type elements are sequence or collection of simple datatypes grouped into a logical unit, in such as way so that these elements may be used in iteration (multiple number of times). A typical example is “CreationInfo” datatype. CreationInfo will contain simple types like SWVersionNo, SWCreatedBy, XMLCreatedBy, XMLCreationDate, IntermediaryCity etc.

```
<xs:element name="CreationInfo" id="CreatSec0">
<xs:annotation>
<xs:documentation>This element will be used by third party vendors
and intermediaries to give details of their software or xml
creation.
</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:sequence>
<xs:element name="SWVersionNo" default="1.0">
<xs:simpleType>
<xs:restriction base="nonEmptyString">
<xs:minLength value="1" />
<xs:maxLength value="10" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="SWCreatedBy" default="DIT">
<xs:simpleType>
<xs:restriction base="nonEmptyString">
<xs:minLength value="0" />
<xs:maxLength value="25" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="XMLCreatedBy" default="DIT" id="CreatSec1">
<xs:simpleType>
<xs:restriction base="nonEmptyString">
<xs:maxLength value="25" />
<xs:minLength value="0" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="XMLCreationDate" id="CreatSec2">
<xs:simpleType>
<xs:restriction base="xs:date">
<xs:minInclusive value="2013-04-01" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="IntermediaryCity" default="Delhi" id="CreatSec3">
<xs:simpleType>
<xs:restriction base="nonEmptyString">
<xs:maxLength value="25" />
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
```

An instance of this schema for complex type may look like :

```
<ITRForm:CreationInfo>
     <ITRForm:SWVersionNo>1.0</ITRForm:SWVersionNo>
     <ITRForm:SWCreatedBy>DIT-EFILING-JAVA</ITRForm:SWCreatedBy>
     <ITRForm:XMLCreatedBy>DIT-EFILING-JAVA</ITRForm:XMLCreatedBy>
     <ITRForm:XMLCreationDate>2014-01-27</ITRForm:XMLCreationDate>
     <ITRForm:IntermediaryCity>Delhi</ITRForm:IntermediaryCity>
</ITRForm:CreationInfo>
```

Appendix 1 – The Complete Schema

ITR is subject to change based on modifications to the Income Tax Act. Please refer to the “**Schema Downloads**” Section of E-filing web site at https://incometaxindiaefiling.gov.in for the latest versions of the schema