Individual Information Web Service
XML Guide

Release 2014.3

Table of Contents

Introduction

This feature provides Broker Dealer firms the ability to download residential and employment history information for individuals that they intend to file U4 Initial/Relicense filings for. Filers will be required to submit requests for this pre-filing information using the web service. This request will be subject to the same rules as applicable on the Pre-registration search page on Web CRD i.e. they will be required to attest that they have the individual's acknowledgement to preview this information. The system shall send the responses immediately in an XML format and shall only include the Employment and Residential history. The web service interface shall require a user name and password. For more information on obtaining a username and password please refer to the FINRA Entitlement Program page. To request a Pre- Registration download the firm will be required to provide:

a. Attestation;
b. Individual's CRD #;
c. Date of Birth and
d. SSN

The response will include:

a. CRD #
b. First, Middle, Last name
c. Any suffix in the name
d. Residential History(including sequence numbers)
e. Employment history (including sequence numbers)

Table of Contents

Pre-Filing Request Via Individual Information Web Service

Firms may submit a Pre-Filing Individual Information machine to machine XML request via the Individual Information web service - GetIndvlInfoXml operation to obtain individual information. The Individual Information Web service is created with .Net 4.0 using https and wsHttpBinding conforming to WS-* standards.
The number of individuals requested in the XML request file submitted via GetIndvlInfoXml operation should not exceed 100 and must also conform to the schema described in the Request Schema Reference section of this document.
The GetIndvlInfoXml operation returns a *Result* XML file that is described in the Result Schema Reference section of this document. The *Result* returned by GetIndvlInfoXml also contains the status of the request and results or error messages if any.

Refer to the Web Service Reference section of this document and Web Service Description Language (WSDL) file (available from the URL below) for detailed information on the operations supported by the web service. Sample code for requesting pre-filing information via the web service is also provided.

The Individual Information Service Web Service URL is:

Production environment (Release 2014.3 will deploy on September 6, 2014):
https://crd.finra.org/Webservices/IndividualInformationService.svc

Table of Contents

IndividualInformationService.GetIndvlInfoXml Operation

Use this operation to submit a Pre-Filing Individual Information Request. In order to invoke this operation the caller must have a FTP account with a special privilege. See the Introduction for more information on obtaining a username and password with the proper privileges.

Syntax

C#

```
public string GetIndvlInfoXml(string xmlStrIn)
```

Java

```
public String GetIndvlInfoXml(String xmlStrIn)
```

Parameters

xmlStrIn

Type: System.String

An XML string in iso-8859-1 encoding containing the individual information request.

Returns

Type: System.String

An XML string in iso-8859-1 encoding containing the result of the operation.

Table of Contents

Pre-Filing Individual Information Request Via the Web Service - (Java)

Java client was tested with Metro JAX-WS with WSIT to conform to WS-*standards (for wsHttpBinding).

Reference article to configure WS-Security for Metro: http://www.ibm.com/developerworks/java/library/j-jws10/index.html

Please contact Support@finra.org if you need test sample code.

Pre-Filing Individual Information Request Via the Web Service - Sample Code (C#)

The sample code provided on the next page may be used in Microsoft Visual Studio 2012 by performing the following steps:

1. Launch Microsoft Visual Studio 2013.
2. From the *File* menu, open the *New Project* dialog by selecting *New*, and then *Project.*
3. In the *New Project* dialog, select the *Visual C#, Windows* node from the *Installed Templates* list and then click on *Console Application*.
4. In the *Name* field, enter *IndividualInformationWebServiceTestUtilityConsole* and then press the *OK* button. The new Console Application is created.
5. From the *Project* menu, select *Add Reference*.
6. On the *.NET* tab of the *Add Reference* dialog, select the *System.ServiceModel* component and then press the *OK* button.
7. From the *Project* menu, select *Add Service Reference*.
8. In the *Address* field enter the Individual Information Web Service URL (https://crd.finra.org/Webservices/IndividualInformationService.svc) and then press the *Go* button.
9. In the *Namespace* field enter *IndividualInformationServiceReference* and then press the *OK* button.
10. Finally, copy-and-paste the sample code from the next page of this document into the Program.cs file, overwriting all of the existing code in that file.

```
usingSystem;
usingSystem.IO;
usingSystem.ServiceModel;
usingSystem.ServiceModel.Security;
using System.Net;
usingSystem.Net.Security;
using System.Security.Cryptography.X509Certificates;
using IndividualInformationWebServiceTestUtilityConsole.IndividualInformationServiceReference;
namespaceIndividualInformationWebServiceTestUtilityConsole
{
classProgram
{
privateEndpointAddress_serviceendPoint;
privateWSHttpBinding_binding;
privatestringfileName = @"C:\Request1File.xml";
staticvoidMain(string [] args)
{
Programp =newProgram();
p.InitializeService();
p.Execute();
}
publicvoidInitializeService()
{
ServicePointManager.ServerCertificateValidationCallback=newRemoteCertificateValidationCallback(CustomCertificateValidation);
_binding = newWSHttpBinding();
_binding.Name = "WSHttpBinding_IndividualInformationService";
_binding.CloseTimeout = newTimeSpan( 0, 3, 0);
_binding.OpenTimeout = newTimeSpan( 0, 3, 0);
_binding.ReceiveTimeout = newTimeSpan( 0, 10, 0);
_binding.SendTimeout = newTimeSpan( 0, 3, 0);
_binding.Security.Mode = SecurityMode.TransportWithMessageCredential;
_binding.Security.Transport.ProxyCredentialType = HttpProxyCredentialType.None;
_binding.Security.Transport.ClientCredentialType = HttpClientCredentialType.None;
_binding.Security.Transport.Realm = "";
_binding.Security.Message.ClientCredentialType = MessageCredentialType.UserName;
_binding.Security.Message.NegotiateServiceCredential = true;
_binding.Security.Message.AlgorithmSuite = SecurityAlgorithmSuite.Default;
_binding.Security.Message.EstablishSecurityContext = true;
_binding.ReaderQuotas.MaxStringContentLength = 10000000;
_binding.MaxReceivedMessageSize = 10000000;
}
publicstaticboolCustomCertificateValidation( objectsender,X509Certificatecert,X509Chainchain,SslPolicyErrorssslError)
{
// Actual certificate validation goes here
returntrue;
}
publicvoidExecute()
{
stringxmlIn;
stringxmlOut = null;
_serviceendPoint = newEndpointAddress(newUri("https://crd.finra.org/WebServices/IndividualInformationService.svc"));
using(variisC = newIndividualInformationServiceClient(_binding, _serviceendPoint))
{
iisC.ClientCredentials.UserName.UserName = "username";
iisC.ClientCredentials.UserName.Password = "password";
try
{
iisC.Open();
using(StreamReaderreader =newStreamReader(fileName))
{
xmlIn = reader.ReadToEnd();
}
xmlOut = iisC.GetIndvlInfoXml(xmlIn);
}
catch(Exceptionex)
{
xmlOut = ex.ToString();
}
finally
{
if (iisC.State == CommunicationState.Opened)iisC.Close();
}
}
Console.WriteLine(xmlOut);
}
}
}
```

Schema Reference Legend

How to Read Schema Reference Pages

The following "Address Type" schema element is provided with annotations that explain the layout and content of the actual schema elements and types in the remainder of the *Schema Reference* sections of this document.

Complex Type: **AdrsType**
Schema Component Type — Schema Component Name

Super-types:	None
Sub-types:	AdrsType (by extension)

If this schema component is a type definition, its type hierarchy is shown in a gray-bordered box.

Name	AdrsType
Abstract	no

The table above displays the properties of this schema component.

XML Instance Representation

```
<...
 Str1="Str50[0..1]"
 Str2="Str50[0..1]"
 City="Str50[0..1]"
 State="StateCdType[0..1]"
 Cntry="Str50[0..1]"
 PostlCd="Str11[0..1]" />
```

The XML Instance Representation table above shows the schema component's content as an XML instance.

- The minimum and maximum occurrence of elements and attributes are provided in square brackets, e.g. [0..1].
- Model group information are shown in gray, e.g. Start Choice ... End Choice.
- For type derivations, the elements and attributes that have been added to or changed from the base type's content are shown in **bold**.
- If an element/attribute has a fixed value, the fixed value is shown in green.
- Otherwise, the type of the element/attribute is displayed.
  - If the element/attribute's type is in the schema, a link is provided to it.
  - For local simple type definitions, the constraints are displayed in angle brackets.

Schema Component Representation

```
<xsd:complexType name="AdrsType">
   <xsd:attribute name="Str1" type="Str50" use="optional"/>
   <xsd:attribute name="Str2" type="Str50" use="optional"/>
   <xsd:attribute name="City" type="Str50" use="optional"/>
   <xsd:attribute name="State" type="StateCdType" use="optional"/>
   <xsd:attribute name="Cntry" type="Str50" use="optional"/>
   <xsd:attribute name="PostlCd" type="Str11" use="optional"/>
</xsd:complexType>
```

The Schema Component Representation table above displays the underlying XML representation of the schema component. (Annotations are not shown.)

Table of Contents

Table of Contents

Schema Properties

Documentation	

Global Declarations

Element: IndvlInfoRqsts

Name	IndvlInfoRqsts
Type	IndvlInfoRqstsType
Nillable	no
Abstract	no

XML Instance Representation

```
<IndvlInfoRqsts>
   <IndvlPrRgstnRqsts> IndvlPrRgstnRqstsType </IndvlPrRgstnRqsts> [1]
</IndvlInfoRqsts>
```

Schema Component Representation

```
<xsd:element name="IndvlInfoRqsts" type="IndvlInfoRqstsType"/>
```

Table of Contents

Global Definitions

Complex Type: IndvlInfoRqstsType

Super-types:	None
Sub-types:	None

Name	IndvlInfoRqstsType
Abstract	no
Documentation	The IndvlInfoRqstsType describes the request information required for individual information service.

XML Instance Representation

```
<...>
   <IndvlPrRgstnRqsts> IndvlPrRgstnRqstsType </IndvlPrRgstnRqsts> [1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="IndvlInfoRqstsType">
   <xsd:sequence>
      <xsd:element name="IndvlPrRgstnRqsts" type="IndvlPrRgstnRqstsType"/>
   </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: IndvlPrRgstnRqstsType

Super-types:	None
Sub-types:	None

Name	IndvlPrRgstnRqstsType
Abstract	no
Documentation	The indvlPrRgstnRqstType describes the request information required for the pre-filing request.

XML Instance Representation

```
<...>
   <!-- This firm is requesting the residential and employment history data for the following individual(s) for the
   purpose of completing and submitting via Web EFT an Initial, Dual and/or Relicense Form U4 for such individual(s).
   This firm affirms that it intends to employ each listed individual as a broker-dealer agent and/or an investment
   adviser representative and that it will use the requested data solely for the purpose of completing and submitting
   via Web EFT a Form U4 for such individual(s). -->
   <RqstAtstn> AnswerYType </RqstAtstn> [1]
   <Rqst> RqstType </Rqst> [1..100]
</...>
```

Schema Component Representation

```
<xsd:complexType name="IndvlPrRgstnRqstsType">
   <xsd:sequence>
      <xsd:element name="RqstAtstn" type="AnswerYType" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="Rqst" type="RqstType" maxOccurs="100"/>
   </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: RqstType

Super-types:	None
Sub-types:	None

Name	RqstType
Abstract	no
Documentation	Describes the information required for an individual pre-filing request record.

XML Instance Representation

```
<...>
   <!-- CRD Number - CRD unique individual identifier -->
   <CRDNb> xsd:integer (0 < value <= 2147483647) (total no. of digits = 10) </CRDNb> [1]
   <!-- Individual SSN -->
   <SSN> xsd:token (pattern = [0-9]{3}-[0-9]{2}-[0-9]{4}) </SSN> [0..1]
   <!-- Individual date of birth in --MM-DD format(xsd:gMonthDay) -->
   <DOB> xsd:gMonthDay </DOB> [1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="RqstType">
  <xsd:sequence>
    <xsd:element name="CRDNb" minOccurs="1" maxOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:totalDigits value="10"/>
          <xsd:minExclusive value="0"/>
          <xsd:maxInclusive value="2147483647"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="SSN" minOccurs="0" maxOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:token">
          <xsd:pattern value="[0-9]{3}-[0-9]{2}-[0-9]{4}"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="DOB" minOccurs="1" maxOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:gMonthDay"/>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Simple Type: AnswerYType

Super-types:	xsd:string < **AnswerYType** (by restriction)
Sub-types:	None

Name	AnswerYType
Content	Base XSD Type: string *value* comes from list: {'Y'}

Schema Component Representation

```
<xsd:simpleType name="AnswerYType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Y"/>
  </xsd:restriction>
</xsd:simpleType>
```

Table of Contents

Glossary

Abstract (Applies to complex type definitions and element declarations). An abstract element or complex type cannot used to validate an element instance. If there is a reference to an abstract element, only element declarations that can substitute the abstract element can be used to validate the instance. For references to abstract type definitions, only derived types can be used.

All Model Group Child elements can be provided *in any order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-all.

Choice Model Group *Only one* from the list of child elements and model groups can be provided in instances. See: http://www.w3.org/TR/xmlschema-1/#element-choice.

Collapse Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32). Then, collapse contiguous sequences of space characters into single space character, and remove leading and trailing space characters.

Disallowed Substitutions (Applies to element declarations). If *substitution* is specified, then substitution group members cannot be used in place of the given element declaration to validate element instances. If *derivation methods*, e.g. extension, restriction, are specified, then the given element declaration will not validate element instances that have types derived from the element declaration's type using the specified derivation methods. Normally, element instances can override their declaration's type by specifying an `xsi:type` attribute.

Key Constraint Like Uniqueness Constraint, but additionally requires that the specified value(s) must be provided. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Key Reference Constraint Ensures that the specified value(s) must match value(s) from a Key Constraint or Uniqueness Constraint. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Model Group Groups together element content, specifying the order in which the element content can occur and the number of times the group of element content may be repeated. See: http://www.w3.org/TR/xmlschema-1/#Model_Groups.

Nillable (Applies to element declarations). If an element declaration is nillable, instances can use the `xsi:nil` attribute. The `xsi:nil` attribute is the boolean attribute, *nil*, from the *http://www.w3.org/2001/XMLSchema-instance* namespace. If an element instance has an `xsi:nil` attribute set to true, it can be left empty, even though its element declaration may have required content.

Notation A notation is used to identify the format of a piece of data. Values of elements and attributes that are of type, NOTATION, must come from the names of declared notations. See: http://www.w3.org/TR/xmlschema-1/#cNotation_Declarations.

Preserve Whitespace Policy Preserve whitespaces exactly as they appear in instances.

Prohibited Derivations (Applies to type definitions). Derivation methods that cannot be used to create sub-types from a given type definition.

Prohibited Substitutions (Applies to complex type definitions). Prevents sub-types that have been derived using the specified derivation methods from validating element instances in place of the given type definition.

Replace Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32).

Sequence Model Group Child elements and model groups must be provided *in the specified order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-sequence.

Substitution Group Elements that are *members* of a substitution group can be used wherever the *head* element of the substitution group is referenced.

Substitution Group Exclusions (Applies to element declarations). Prohibits element declarations from nominating themselves as being able to substitute a given element declaration, if they have types that are derived from the original element's type using the specified derivation methods.

Target Namespace The target namespace identifies the namespace that components in this schema belongs to. If no target namespace is provided, then the schema components do not belong to any namespace.

Uniqueness Constraint Ensures uniqueness of an element/attribute value, or a combination of values, within a specified scope. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Table of Contents

Pre-Filing Result Schema Reference

Schema Properties

Documentation	

Global Declarations

Element: IndvlInfo

Name	IndvlInfo
Type	IndvlInfoType
Nillable	no
Abstract	no

XML Instance Representation

```
<IndvlInfo>
   <IndvlPrRgstnRcrds> IndvlPrRgstnRcrdsType </IndvlPrRgstnRcrds> [1]
</IndvlInfo>
```

Schema Component Representation

```
<xsd:element name="IndvlInfo" type="IndvlInfoType"/>
```

Table of Contents

Global Definitions

Complex Type: AdrsType

Super-types:	None
Sub-types:	None

Name	AdrsType
Abstract	no
Documentation	Contains Street1, Street2, City, State, Postal Code and Country fields and used in a address record.

XML Instance Representation

```
<...>
   <!-- Address - Street1. This information maps to Address Street1 field of a record. -->
   <Strt1> Str50 </Strt1> [0..1]
   <!-- Address - Street2. This information maps to Address Street2 field of a record. -->
   <Strt2> Str50 </Strt2> [0..1]
   <!-- Address - City. This information maps to Address City field of a record. -->
   <City> Str50 </City> [0..1]
   <!-- Address - State. This information maps to Address State field of a record. -->
   <State> StateCdType </State> [0..1]
   <!-- Address - Country. This information maps to Address Country field of a record. -->
   <Cntry> Str50 </Cntry> [0..1]
   <!-- Address Postal Code. This information maps to Address Postal Code field of a record. -->
   <PostlCd> Str11 </PostlCd> [0..1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="AdrsType">
   <xsd:sequence>
      <xsd:element name="Strt1" type="Str50" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Strt2" type="Str50" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="City" type="Str50" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="State" type="StateCdType" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Cntry" type="Str50" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="PostlCd" type="Str11" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: AdrsTypeShort

Super-types:	None
Sub-types:	None

Name	AdrsTypeShort
Abstract	no
Documentation	Contains City, State, Country fields and used in a address record.

XML Instance Representation

```
<...>
   <!-- Address - City. This information maps to Address City field of a record. -->
   <City> Str50 </City> [0..1]
   <!-- Address - State. This information maps to Address State field of a record. -->
   <State> StateCdType </State> [0..1]
   <!-- Address - Country. This information maps to Address Country field of a record. -->
   <Cntry> Str50 </Cntry> [0..1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="AdrsTypeShort">
   <xsd:sequence>
      <xsd:element name="City" type="Str50" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="State" type="StateCdType" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Cntry" type="Str50" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: EmpltHsType

Super-types:	None

Sub-types:	None

Name	EmpltHsType
Abstract	no
Documentation	Contains a collection of employment history details for the individual. This information maps to FORM U4 Section 12. Employment History section.

XML Instance Representation

```
<...>
   <Emplt> EmpltType </Emplt> [0..*]
</...>
```

Schema Component Representation

```
<xsd:complexType name="EmpltHsType">
   <xsd:sequence>
      <xsd:element name="Emplt" type="EmpltType" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: EmpltType

Super-types:	None
Sub-types:	None

Name	EmpltType
Abstract	no
Documentation	Contains one of many employment history details for the individual. This information maps to a FORM U4 Section 12. Employment History record.

XML Instance Representation

```
<...>
   <!-- The unique sequence number of the Employment history record in FORM U4 Section 12. Employment History record. --
   >
   <SeqNb> Int16 </SeqNb> [1]
   <!-- The organization name. This information maps to Name of Firm or Company field of a FORM U4 Section 12.
   Employment History record. -->
   <OrgNm> Str64 </OrgNm> [1]
   <!-- From Date. This information maps to FORM U4 Section 12. Employment History - From date field of a Employment
   history record. -->
   <FromDt> xsd:gYearMonth </FromDt> [1]
   <!-- To Date. This information maps to FORM U4 Section 12. Employment History - To date field of a Employment history
   record. -->
   <ToDt> xsd:gYearMonth </ToDt> [0..1]
   <!-- This information maps to FORM U4 Section 12. Employment History - City, State, Country fields of a Employment
   History record. -->
   <Adrs> AdrsTypeShort </Adrs> [0..1]
   <!-- Investment related business indicator. This information maps to Investment-Related Business? field of a FORM U4
   Section 12. Employment History record. -->
   <NvsmtRltdBus> AnswerYNType </NvsmtRltdBus> [0..1]
   <!-- Position Held. This information maps to Position Held field of a FORM U4 Section 12. Employment History record.
   -->
   <PstnHeld> Str512 </PstnHeld> [0..1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="EmpltType">
  <xsd:sequence>
    <xsd:element name="SeqNb" type="Int16" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="OrgNm" type="Str64" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="FromDt" type="xsd:gYearMonth" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="ToDt" type="xsd:gYearMonth" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="Adrs" type="AdrsTypeShort" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="NvsmtRltdBus" type="AnswerYNType" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="PstnHeld" type="Str512" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: ErrMsgType

Super-types:	None
Sub-types:	None

Name	ErrMsgType
Abstract	no

XML Instance Representation

```
<...>
  <!-- Error number. -->
  <Cd> xsd:unsignedInt </Cd> [0..1]
  <Type> xsd:NMTOKEN (value comes from list: {'SCHEMA' | 'SYSTEM'}) </Type> [0..1]
  <!-- Detailed error description. Contact FINRA Gateway Call Center at (301) 869-6699 for General System Error. -->
  <Desc> xsd:string </Desc> [0..1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="ErrMsgType">
  <xsd:sequence>
    <xsd:element name="Cd" type="xsd:unsignedInt" minOccurs="0"/>
    <xsd:element name="Type" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="SCHEMA"/>
          <xsd:enumeration value="SYSTEM"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Desc" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: IndvlInfoType

Super-types:	None
Sub-types:	None

Name	IndvlInfoType
Abstract	no
Documentation	Provides the requested information of the individuals. A maximum of 100 individuals is allowed per instance.

XML Instance Representation

```
<...>
  <IndvlPrRgstnRcrds> IndvlPrRgstnRcrdsType </IndvlPrRgstnRcrds> [1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="IndvlInfoType">
  <xsd:sequence>
    <xsd:element name="IndvlPrRgstnRcrds" type="IndvlPrRgstnRcrdsType"/>
  </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: IndvlPrRgstnRcrdsType

Super-types:	None
Sub-types:	None

Name	IndvlPrRgstnRcrdsType
Abstract	no
Documentation	Provides the requested pre-filing information of the individuals. A maximum of 100 individuals is allowed per instance.

XML Instance Representation

```
<...>
  <OprtnSt> xsd:NMTOKEN (value comes from list: {'PROCFAIL' | 'SUCCESS' | 'SCHEMAVALFAIL' | 'SCHEMAVALFAILAUTH'})
  </OprtnSt> [1]
  <Indvl> IndvlType </Indvl> [0..100]
  <!-- Collection of error messages (if any) for this request. -->
  <ErrMsgs> [0..1]
    <ErrMsg> ErrMsgType </ErrMsg> [1..*]
  </ErrMsgs>
</...>
```

Schema Component Representation

```
<xsd:complexType name="IndvlPrRgstnRcrdsType">
  <xsd:sequence>
    <xsd:element name="OprtnSt" minOccurs="1" maxOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="PROCFAIL"/>
          <xsd:enumeration value="SUCCESS"/>
          <xsd:enumeration value="SCHEMAVALFAIL"/>
          <xsd:enumeration value="SCHEMAVALFAILAUTH"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Indvl" type="IndvlType" minOccurs="0" maxOccurs="100"/>
    <xsd:element name="ErrMsgs" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="ErrMsg" type="ErrMsgType" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: IndvlType

Super-types:	None
Sub-types:	None

Name	IndvlType
Abstract	no
Documentation	Describes the information pertaining to an individual.

XML Instance Representation

```
<...>
   <CRDNb> xsd:integer (0 < value <= 2147483647) (total no. of digits = 10) </CRDNb> [1]
   <!-- Indicates whether the identifying information provided for the individual is valid. -->
   <ValidRqst> AnswerYNType </ValidRqst> [1]
   <Nm> NmType </Nm> [0..1]
   <RsdnlHs> RsdnlHsType </RsdnlHs> [0..1]
   <EmpltHs> EmpltHsType </EmpltHs> [0..1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="IndvlType">
   <xsd:sequence>
      <xsd:element name="CRDNb" minOccurs="1" maxOccurs="1">
         <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
               <xsd:totalDigits value="10"/>
               <xsd:minExclusive value="0"/>
               <xsd:maxInclusive value="2147483647"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="ValidRqst" type="AnswerYNType" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="Nm" type="NmType" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="RsdnlHs" type="RsdnlHsType" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="EmpltHs" type="EmpltHsType" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: NmType

Super-types:	None
Sub-types:	None

Name	NmType
Abstract	no

XML Instance Representation

```
<...>
   <Last> xsd:string (pattern = [^0-9]+) (length >= 1) </Last> [1]
   <First> xsd:string (pattern = [^0-9]+) (length >= 1) </First> [1]
   <Mid> xsd:string (pattern = [^0-9]+) (length >= 1) </Mid> [0..1]
   <Suf> xsd:string (pattern = [^0-9]+) (length >= 1) </Suf> [0..1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="NmType">
  <xsd:sequence>
    <xsd:element name="Last" minOccurs="1" maxOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/>
          <xsd:maxLength value="25"/>
          <xsd:pattern value="[^0-9]+"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="First" minOccurs="1" maxOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/>
          <xsd:maxLength value="25"/>
          <xsd:pattern value="[^0-9]+"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Mid" minOccurs="0" maxOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/>
          <xsd:maxLength value="25"/>
          <xsd:pattern value="[^0-9]+"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Suf" minOccurs="0" maxOccurs="1">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/>
          <xsd:maxLength value="25"/>
          <xsd:pattern value="[^0-9]+"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: RsdncType

Super-types:	None
Sub-types:	None

Name	RsdncType
Abstract	no
Documentation	Contains a residential address. This information maps to FORM U4 Section 11 Residential History record.

XML Instance Representation

```
<...>
  <!-- The unique sequence number of the residential history record in FORM U4 Section 11 Residential History section.
  -->
  <SeqNb> Int16 </SeqNb> [1]
  <!-- From Date. This information maps to FORM U4 Section 11. Residential History - From date field of a residential
  history record. -->
  <FromDt> xsd:gYearMonth </FromDt> [1]
  <!-- To Date. This information maps to FORM U4 Section 11. Residential History - To date field of a residential
  history record. -->
  <ToDt> xsd:gYearMonth </ToDt> [0..1]
  <!-- This information maps to FORM U4 Section 11. Residential History - Address fields - Street1, Street2, City,
  State, Postal Code and Country fields of a record. -->
  <Adrs> AdrsType </Adrs> [0..1]
</...>
```

Schema Component Representation

```
<xsd:complexType name="RsdncType">
  <xsd:sequence>
    <xsd:element name="SeqNb" type="Int16" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="FromDt" type="xsd:gYearMonth" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="ToDt" type="xsd:gYearMonth" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="Adrs" type="AdrsType" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Complex Type: RsdnlHsType

Super-types:	None
Sub-types:	None

Name	RsdnlHsType
Abstract	no
Documentation	Contains a collection individual residential addresses for the past five (5) years. This section maps to FORM U4 Section 11. Residential History.

XML Instance Representation

```
<...>
  <Rsdnc> RsdncType </Rsdnc> [0..*]
</...>
```

Schema Component Representation

```
<xsd:complexType name="RsdnlHsType">
  <xsd:sequence>
    <xsd:element name="Rsdnc" type="RsdncType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Table of Contents

Simple Type: AnswerYNType

Super-types:	xsd:string < **AnswerYNType** (by restriction)
Sub-types:	None

Name	AnswerYNType
Content	• Base XSD Type: string • *value* comes from list: {'Y' | 'N'}

Schema Component Representation

```
<xsd:simpleType name="AnswerYNType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Y"/>
    <xsd:enumeration value="N"/>
  </xsd:restriction>
</xsd:simpleType>
```

Table of Contents

Simple Type: Int16

Super-types:	xsd:integer < **Int16** (by restriction)
Sub-types:	None

Name	Int16
Content	• Base XSD Type: integer • *total no. of digits* = 16

Schema Component Representation

```
<xsd:simpleType name="Int16">
  <xsd:restriction base="xsd:integer">
    <xsd:totalDigits value="16"/>
  </xsd:restriction>
</xsd:simpleType>
```

Table of Contents

Simple Type: StateCdType

Super-types:	xsd:NMTOKEN < **StateCdType** (by restriction)
Sub-types:	None

Name	StateCdType
Content	• Base XSD Type: NMTOKEN • *value* comes from list: {'AL' | 'AK' | 'AR' | 'AZ' | 'CA' | 'CO' | 'CT' | 'DC' | 'DE' | 'FL' | 'GA' | 'HI' | 'IA' | 'ID' | 'IL' | 'IN' | 'KS' | 'KY' | 'LA' | 'MA' | 'MD' | 'ME' | 'MI' | 'MN' | 'MO' | 'MS' | 'MT' | 'NC' | 'ND' | 'NE' | 'NH' | 'NJ' | 'NL' | 'NM' | 'NV' | 'NY' | 'OH' | 'OK' | 'OR' | 'PA' | 'PR' | 'RI' | 'SC' | 'SD' | 'TN' | 'TX' | 'UT' | 'VA' | 'VI' | 'VT' | 'WA' | 'WI' | 'WV' | 'WY'} • *length* >= 2
Documentation	The state code

Schema Component Representation

```
<xsd:simpleType name="StateCdType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:minLength value="2"/>
    <xsd:maxLength value="2"/>
    <xsd:enumeration value="AL"/>
    <xsd:enumeration value="AK"/>
    <xsd:enumeration value="AR"/>
    <xsd:enumeration value="AZ"/>
    <xsd:enumeration value="CA"/>
    <xsd:enumeration value="CO"/>
    <xsd:enumeration value="CT"/>
    <xsd:enumeration value="DC"/>
    <xsd:enumeration value="DE"/>
    <xsd:enumeration value="FL"/>
    <xsd:enumeration value="GA"/>
    <xsd:enumeration value="HI"/>
    <xsd:enumeration value="IA"/>
    <xsd:enumeration value="ID"/>
    <xsd:enumeration value="IL"/>
    <xsd:enumeration value="IN"/>
    <xsd:enumeration value="KS"/>
    <xsd:enumeration value="KY"/>
    <xsd:enumeration value="LA"/>
    <xsd:enumeration value="MA"/>
    <xsd:enumeration value="MD"/>
    <xsd:enumeration value="ME"/>
    <xsd:enumeration value="MI"/>
    <xsd:enumeration value="MN"/>
    <xsd:enumeration value="MO"/>
    <xsd:enumeration value="MS"/>
    <xsd:enumeration value="MT"/>
    <xsd:enumeration value="NC"/>
    <xsd:enumeration value="ND"/>
    <xsd:enumeration value="NE"/>
    <xsd:enumeration value="NH"/>
    <xsd:enumeration value="NJ"/>
    <xsd:enumeration value="NL"/>
    <xsd:enumeration value="NM"/>
    <xsd:enumeration value="NV"/>
    <xsd:enumeration value="NY"/>
    <xsd:enumeration value="OH"/>
    <xsd:enumeration value="OK"/>
    <xsd:enumeration value="OR"/>
    <xsd:enumeration value="PA"/>
    <xsd:enumeration value="PR"/>
    <xsd:enumeration value="RI"/>
    <xsd:enumeration value="SC"/>
    <xsd:enumeration value="SD"/>
    <xsd:enumeration value="TN"/>
    <xsd:enumeration value="TX"/>
    <xsd:enumeration value="UT"/>
    <xsd:enumeration value="VA"/>
    <xsd:enumeration value="VI"/>
    <xsd:enumeration value="VT"/>
    <xsd:enumeration value="WA"/>
    <xsd:enumeration value="WI"/>
    <xsd:enumeration value="WV"/>
    <xsd:enumeration value="WY"/>
  </xsd:restriction>
</xsd:simpleType>
```

Table of Contents

Simple Type: Str11

Super-types:	xsd:string < **Str11** (by restriction)
Sub-types:	None

Name	Str11
Content	l Base XSD Type: string l *length* <= 11

Schema Component Representation

```
<xsd:simpleType name="Str11">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="11"/>
  </xsd:restriction>
</xsd:simpleType>
```

Table of Contents

Simple Type: Str50

Super-types:	xsd:string < **Str50** (by restriction)
Sub-types:	None

Name	Str50
Content	l Base XSD Type: string l *length* <= 50

Schema Component Representation

```
<xsd:simpleType name="Str50">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="50"/>
  </xsd:restriction>
</xsd:simpleType>
```

Table of Contents

Simple Type: Str512

Super-types:	xsd:string < **Str512** (by restriction)
Sub-types:	None

Name	Str512
Content	l Base XSD Type: string l *length* <= 512

Schema Component Representation

```
<xsd:simpleType name="Str512">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="512"/>
  </xsd:restriction>
</xsd:simpleType>
```

Table of Contents

Simple Type: Str64

Super-types:	xsd:string < **Str64** (by restriction)
Sub-types:	None

Name	Str64
Content	l Base XSD Type: string l *length* <= 64

Schema Component Representation

```
<xsd:simpleType name="Str64">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="64"/>
  </xsd:restriction>
</xsd:simpleType>
```

Table of Contents

Glossary

Abstract (Applies to complex type definitions and element declarations). An abstract element or complex type cannot used to validate an element instance. If there is a reference to an abstract element, only element declarations that can substitute the abstract element can be used to validate the instance. For references to abstract type definitions, only derived types can be used.

All Model Group Child elements can be provided *in any order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-all.

Choice Model Group *Only one* from the list of child elements and model groups can be provided in instances. See: http://www.w3.org/TR/xmlschema-1/#element-choice.

Collapse Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32). Then, collapse contiguous sequences of space characters into single space character, and remove leading and trailing space characters.

Disallowed Substitutions (Applies to element declarations). If *substitution* is specified, then substitution group members cannot be used in place of the given element declaration to validate element instances. If *derivation methods*, e.g. extension, restriction, are specified, then the given element declaration will not validate element instances that have types derived from the element declaration's type using the specified derivation methods. Normally, element instances can override their declaration's type by specifying an `xsi:type` attribute.

Key Constraint Like Uniqueness Constraint, but additionally requires that the specified value(s) must be provided. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Key Reference Constraint Ensures that the specified value(s) must match value(s) from a Key Constraint or Uniqueness Constraint. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Model Group Groups together element content, specifying the order in which the element content can occur and the number of times the group of element content may be repeated. See: http://www.w3.org/TR/xmlschema-1/#Model_Groups.

Nillable (Applies to element declarations). If an element declaration is nillable, instances can use the `xsi:nil` attribute. The `xsi:nil` attribute is the boolean attribute, *nil*, from the *http://www.w3.org/2001/XMLSchema-instance* namespace. If an element instance has an `xsi:nil` attribute set to true, it can be left empty, even though its element declaration may have required content.

Notation A notation is used to identify the format of a piece of data. Values of elements and attributes that are of type, NOTATION, must come from the names of declared notations. See: http://www.w3.org/TR/xmlschema-1/#cNotation_Declarations.

Preserve Whitespace Policy Preserve whitespaces exactly as they appear in instances.

Prohibited Derivations (Applies to type definitions). Derivation methods that cannot be used to create sub-types from a given type definition.

Prohibited Substitutions (Applies to complex type definitions). Prevents sub-types that have been derived using the specified derivation methods from validating element instances in place of the given type definition.

Replace Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32).

Sequence Model Group Child elements and model groups must be provided *in the specified order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-sequence.

Substitution Group Elements that are *members* of a substitution group can be used wherever the *head* element of the substitution group is referenced.

Substitution Group Exclusions (Applies to element declarations). Prohibits element declarations from nominating themselves as being able to substitute a given element declaration, if they have types that are derived from the original element's type using the specified derivation methods.

Target Namespace The target namespace identifies the namespace that components in this schema belongs to. If no target namespace is provided, then the schema components do not belong to any namespace.

Uniqueness Constraint Ensures uniqueness of an element/attribute value, or a combination of values, within a specified scope. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Table of Contents

Individual Information Service Request/Result Examples

Example - Pre-Filing Request

```
<? xml version = " 1.0 " encoding = " iso-8859-1 " ?>
< IndvlInfoRqsts xsi:noNamespaceSchemaLocation = " IndividualInformationRequest.xsd " xmlns:xsi = " http://www.w3.org/2001/XMLSchema-instance " >
        < IndvlPrRgstnRqsts >
                < RqstAtstn > Y </ RqstAtstn >
                < Rqst >
                        < CRDNb > 897865 </ CRDNb >
                        < SSN > 123-45-6789 </ SSN >
                        < DOB > --01-01 </ DOB >
                </ Rqst >
                < Rqst >
                        < CRDNb > 78975 </ CRDNb >
                        < SSN > 111-22-3333 </ SSN >
                        < DOB > --11-11 </ DOB >
                </ Rqst >
                < Rqst >
                        < CRDNb > 98375 </ CRDNb >
                        < DOB > --12-29 </ DOB >
                </ Rqst >
        </ IndvlPrRgstnRqsts >
</ IndvlInfoRqsts >
```

Example - Pre-Filing Result

```
<? xml version = " 1.0 " encoding = " iso-8859-1 " ?>
< IndvlInfo xsi:noNamespaceSchemaLocation = " IndividualInformationResults.xsd " xmlns:xsi = " http://www.w3.org/2001/XMLSchema-instance " >
    < IndvlPrRgstnRcrds >
        < OprtnSt > SUCCESS </ OprtnSt >
        < Indvl >
            < CRDNb > 250624 </ CRDNb >
            < ValidRqst > Y </ ValidRqst >
            < Nm >
                < Last > JOHN </ Last >
                < First > WINTER </ First >
                < Mid > ART </ Mid >
                < Suf > III </ Suf >
            </ Nm >
            < RsdnlHs >
                < Rsdnc >
                    < SeqNb > 12 </ SeqNb >
                    < FromDt > 1970-11 </ FromDt >
                    < ToDt > 1980-01 </ ToDt >
                    < Adrs >
                        < Strt1 > FIRST STREET </ Strt1 >
                        < Strt2 > MAIN STREET </ Strt2 >
                        < City > ROCKVILLE </ City >
                        < State > MD </ State >
                        < Cntry > UINTED STATES </ Cntry >
                        < PostlCd > 20790 </ PostlCd >
                    </ Adrs >
                </ Rsdnc >
                < Rsdnc >
                    < SeqNb > 9 </ SeqNb >
                    < FromDt > 1968-11 </ FromDt >
                    < ToDt > 1970-01 </ ToDt >
                    < Adrs >
                        < Strt1 > 2 STREET </ Strt1 >
                        < Strt2 > KEY STREET </ Strt2 >
                        < City > FAIRFAX </ City >
                        < State > VA </ State >
                        < Cntry > UINTED STATES </ Cntry >
                        < PostlCd > 20791 </ PostlCd >
                    </ Adrs >
                </ Rsdnc >
            </ RsdnlHs >
            < EmpltHs >
                < Emplt >
                    < SeqNb > 111 </ SeqNb >
                    < OrgNm > ABC CORP </ OrgNm >
                    < FromDt > 2002-02 </ FromDt >
                    < ToDt > 2004-04 </ ToDt >
                    < Adrs >
                        < City > BALTIMORE </ City >
                        < State > MD </ State >
                        < Cntry > UNITED STATES </ Cntry >
                    </ Adrs >
                    < NvsmtRltdBus > Y </ NvsmtRltdBus >
                    < PstnHeld > CFO </ PstnHeld >
                </ Emplt >
                < Emplt >
                    < SeqNb > 112 </ SeqNb >
                    < OrgNm > XYZ CORP </ OrgNm >
                    < FromDt > 2001-02 </ FromDt >
                    < ToDt > 2002-04 </ ToDt >
                    < Adrs >
                        < City > BALTIMORE </ City >
                        < State > MD </ State >
                        < Cntry > UNITED STATES </ Cntry >
                    </ Adrs >
                    < NvsmtRltdBus > N </ NvsmtRltdBus >
                    < PstnHeld > EDITOR </ PstnHeld >
                </ Emplt >
            </ EmpltHs >
        </ Indvl >
        < Indvl >
            < CRDNb > 1346678 </ CRDNb >
            < ValidRqst > N </ ValidRqst >
        </ Indvl >
        < Indvl >
            < CRDNb > 8866678 </ CRDNb >
            < ValidRqst > Y </ ValidRqst >
            < Nm >
                < Last > JAMES </ Last >
                < First > SUMMER </ First >
                < Mid > ART </ Mid >
```

```
                    < Suf > II </ Suf >
                </ Nm >
                < RsdnlHs >
                    < Rsdnc >
                        < SeqNb > 14 </ SeqNb >
                        < FromDt > 1995-11 </ FromDt >
                        < ToDt > 2000-01 </ ToDt >
                        < Adrs >
                            < Strt1 > THIRD STREET </ Strt1 >
                            < Strt2 > WEST STREET </ Strt2 >
                            < City > BALTIMORE </ City >
                            < State > MD </ State >
                            < Cntry > UINTED STATES </ Cntry >
                            < PostlCd > 20790 </ PostlCd >
                        </ Adrs >
                    </ Rsdnc >
                </ RsdnlHs >
            </ Indvl >
        </ IndvlPrRgstnRcrds >
</ IndvlInfo >
```