IRS Enterprise Data Standards and Guidelines

IRS XML Standards and Guidelines

Introduction

This document is a major revision to the IRS XML Standards and Guidelines found in Volume 4 of the EDSG, distributed for review. It will be incorporated within the official copy of the EDSG once approved. The cover page, and page headers and footers are solely for presentation purposes.

Volume 4. Special Tools and Techniques

4.1 Extensible Markup Language (XML)

4.1.1 IRS XML Background

The Extensible Markup Language (XML) is a meta-markup language syntax specification developed by the World Wide Web Consortium (W3C). The current syntax specification is the Extensible Markup Language (XML) 1.0 Third Edition W3C Recommendation dated February 4, 2004. XML does not have a fixed set of tags and elements because it is designed to be extended indefinitely. Each extension is called an XML application having its own syntax and vocabulary that adhere to the fundamental rules of XML.

As of January 2003, the IRS has already adopted XML as a de facto, recommended syntax specification in more than twenty different applications. Although each application had its own XML standards, there was also regular, informal coordination among IRS XML stakeholders and with the tax industry. On January 22, 2003, the IRS Enterprise Data Management Organization (EDMO) was assigned the role of serving as the single IRS enterprise XML Point of Contact (POC).

It is the on-going policy of EDMO to work collaboratively on the establishment of policy and standards with those organizations and projects experienced in or affected by the proposed work. The IRS XML Community of Practice (xmlCoP) was established to facilitate communication and collaboration on XML policy, standards and practices within the IRS.

4.1.1.1 IRS XML Naming and Design Rules (NDR) Development Process

The IRS XML xmlCoP convened a working a group to define Naming and Design Rules for the use of XML in the IRS. These rules were based on the International standard, Universal Business Language (UBL), as supported by the Federal CIO's Council and International technical committees (OASIS, OECD, and ISO). The NDR will be periodically reviewed and updated, if necessary, in order to keep current with UBL updates and any other relevant industry standards.

4.1.1.2 IRS XML Naming and Design Rules (NDR) Purpose

The EDMO is responsible for the IRS XML naming and design rules as well as for the standardized XML components and products resulting from their use. This section of this document includes the XML naming and design rules for constructing and naming XML components or products such as namespaces, schemas, elements, attributes, data types, and stylesheets.

This section intentionally does not include the specific standardized IRS XML components or products that are subsequently created by applying these IRS XML naming and design rules. When approved, these standardized IRS XML components and products will be stored separately in the IRS XML registry to promote reuse. Please visit the IRS XML registry for more information once it becomes operational.

Effective XML management requires a balance between diversity and conformity recognizing that diversity can impose increased transformation costs and performance penalties. The principal objective of all the XML naming and design rules is to facilitate consistency and interoperability across the IRS and with its data exchange partners.

4.1.1.3 Required Lexicon for IRS XML Naming and Design Rules

IRS XML stakeholders must clearly understand what IRS XML Naming and Design Rules address. Any discussion of XML immediately introduces the use of a large and complex lexicon. XML stakeholders may not already fully understand this lexicon and/or may not share identical interpretations of its terminology. As a result, this document presents appropriate XML lexicon at the beginning of each sub-section rather than in the Glossary. Once established this lexicon is then immediately followed by the XML naming and design rules for that sub-section.

Throughout the remainder of this section, this lexicon will sequentially develop basic relationships among XML components. A namespace is the highest-level XML component and more likely contains schemas than anything else. A schema consists of elements. An element start tag may contain attributes. Each element also has a type. A type is either a local or global data type, depending upon the extent of reuse. Complex data types are collections of simple data types. Each schema requires a corresponding style sheet to complete its presentation.

4.1.2 Terminology and Notation

Throughout this document, the terms XML components, XML artifacts and XML products will be used interchangeably.

The key words **MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY,** and **OPTIONAL** in this document are to be interpreted as described in Internet Engineering Task Force (IETF) Request for Comments (RFC) 2119. Non-capitalized forms of these words are used in the regular English sense.

Rules are identified with a Rule Prefix Token and a number.
[RRRn] – Identification of a rule that requires conformance to ensure that an XML Schema is IRS NDR conformant. The value RRR is a prefix to categorize the type of rule where the value of RRR is as defined in Figure 1, and n (1..n) indicates the sequential number of the rule within its category**.**

Figure 1 - Rule Prefix Token Value

Rule Prefix Token	Value
ATD	Attribute Declaration
CDL	Code List
COM	ComplexType
CVR	Component Versioning
DOC	Artifact Documentation
ELD	Element Declaration
FRM	Forms

GNR	General Naming
GXS	General Construct
IMP	Schema Import
IND	Instance Document
NMP	Namespace Prefix
NMS	Namespace
SCF	Schema File
SCH	Schema Organization
SIM	SimpleType
SSM	Schema Modularity
STA	Standards Adherence
VER	Schema Versioning

4.1.3 Industry XML Standards

The IRS XML Naming and Design Rules (NDR) will remain closely tied to the following industry standards:

- The World Wide Web Consortium (W3C) family of XML standards
- ebXML (Electronic Business using eXtensible Markup Language)
- UBL (Universal Business Language)

UBL is an implementation of ebXML. ebXML, originally an Oasis standard, is now an ISO standard (ISO15000-5) and focuses on the design of reuseable components.

4.1.3.1 IRS XML Standards Adherence Naming and Design Rules

The standards that IRS XML developers will adhere to are:

Rule #	Rule
STA1	All IRS schema design rules MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.
STA2	All IRS schema and messages MUST be based on the W3C suite of technical specifications holding recommendation status.
STA3	All IRS XML schemas are REQUIRED to be subject to change control procedures, specifically version control.
STA4	DTDs SHOULD NOT be used, except with a waiver from the IRS EDMO.

4.1.4 XML Namespaces

4.1.4.1 Namespace Lexicon

At the highest level, an XML namespace organizes standardized XML components and products in enterprise business areas to guarantee their uniqueness. An XML namespace is a collection of available XML information resources for a particular Community of Interest (COI). Each XML namespace has a unique Uniform Resource Identifier (URI), a namespace manager who determines its specific contents, and its existence is recorded and published in the IRS XML registry.

An XML namespace typically contains XML schemas consisting of element types and attribute names. (XML schemas, elements, and attributes are defined respectively in the subsection 4.1.5 and subsection 4.1.6). Namespace identification is specified by the *targetNamespace* attribute. Any element type or attribute name in an XML namespace can be uniquely identified by a two-part name that consists of its XML namespace and its local XML schema name. In addition to XML schemas, an XML namespace may contain other XML information resources such as submission packages, documents, XML samples, and Extensible Stylesheet Language (XSL) stylesheets.

At the lowest level, an XML namespace is declared with an *xmlns* attribute that may also associate a prefix with the namespace. The basic format for declaring an XML namespace anywhere in an XML document is *xmlns:prefix="URI"* as used in the following examples first with, and then without, a prefix:

- *xmlns:xsi="urn:us:gov:irs:Common:Sample"*
- *xmlns="urn:us:gov:irs:Common:Sample"*
- For the purposes of this document, an XML namespace is considered to be any or all of the contexts in the three lexicon paragraphs immediately above.

4.1.4.2 IRS XML Namespace Naming and Design Rules

All IRS developed schemas must follow a consistent approach to namespace declaration and must abide by the following rules:

Rule #	Rule
NMS1	The IRS will use multiple namespaces. Every schema and each subsequent version MUST have its own unique namespace.
NMS2	As the unique namespace identifier (Unique Resource Identifier: URI), all IRS developed schemas MUST use URN (Uniform Resource Name) namespace identifiers.

Rule #	Rule
NMS3	The tokens comprising the URN MUST adhere to the following guidelines: • Whitespace MUST NOT be used within the URN. • Special characters MUST NOT be used within the URN. Special characters are those characters outside of the range 0-9 or a-z. Periods are acceptable. A single hyphen is acceptable between the schema name and the version number. • Only lowercase letters MUST be used.
NMS4	Only IRS-developed schema modules MUST be contained in the IRS namespace: urn:us:gov:irs:
NMS5	The namespace name of all IRS developed schemas MUST conform to the following pattern: urn:us:gov:irs:{structure}:{schema name}-{major version number}.{minor version number} The structure portion of this pattern is REQUIRED to include the following predefined structure components; all others are REQUIRED to gain prior approval of EDMO before they become valid. urn:us:gov:irs:common – to store all IRS developed component schemas urn:us:gov:irs:codelist – to store all IRS developed code list schemas
NMS6	The namespace name of each IRS Code List Schema Module MUST conform to the following pattern: urn:us:gov:irs:codelist:{code list name}-{major version number}.{minor version number} For example, the proper namespace name for the country code list will be: urn:us:gov:irs:codelist:countrycode-1.0, where 1.0 is the version number.
NMS7	IRS published namespaces MUST never be changed.
NMS8	Every IRS XML namespace MUST be public.
NMS9	The standard namespace declaration MUST include the following defaults: xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified"
NMS10	Every IRS-defined or -used schema module MUST have a namespace declared using the xsd:targetNamespace attribute.
NMS11	The IRS MAY copy or use externally developed schemas provided that the namespace declaration in such schemas is not altered. Further, the prefixes used for these schemas MUST not conflict with existing IRS namespace prefixes.

176

4.1.4.3 IRS XML Namespace Prefix Naming and Design Rules

Namespace declarations can identify an associated prefix – shorthand identifier – that allows for compression of the namespace name. All IRS developed schemas must follow a consistent approach to namespace prefix declarations and must abide by the following namespace prefix rules:

Rule #	Rule
NMP1	All namespace prefix tokens for IRS developed schemas MUST begin with the letters "irs" and MUST be followed by the schema name.
NMP2	The IRS CommonAggregateComponents schema module MUST reside in its own namespace and MUST be represented by the prefix token "irscac".
NMP3	The IRS CommonBasicComponents schema module MUST reside in its own namespace and MUST be represented by the prefix token "irscbc".
NMP4	The IRS SpecializedDatatypes schema module MUST reside in its own namespace and MUST be represented by the prefix token "irssdt".

4.1.5 XML Schemas

4.1.5.1 Schema Lexicon

An XML schema is one of two ways to define XML structure and allow multiple organizations to use the same XML vocabulary. The Document Type Definition (DTD) is the alternative. A DTD defines the rules, or permitted markup, for a particular XML application and is required for a valid XML document. A DTD can be part of an XML document or referenced in a separate file.

In industry best practices, the XML schema is replacing the DTD. In comparison, the XML schema, written in XML syntax, more rigorously constrains the content type and data type of XML tags. An XML schema is a superset of a DTD and uses an *'*xs' or *'xsd'* (XML Schema Definition) prefix in its XML schema statements as shown in the examples below.

- <xsd:element name=”FirstName” type=”xsd.string”/>
- <xs:element name=”LastName” type=”xsd.string”/>

The XML document structure constraints in an XML schema address the unique element types and attribute names in an XML namespace. In addition, an XML schema may also have other components dealing with the content contained in entities and notations, inheritance, embedded documentation, application-specific constraints, and data types.

For the purposes of this IRS EDSG document, an XML schema is considered to be a formal definition of the structure, content, and semantics of XML documents.

4.1.5.2 IRS XML Schema File Name Naming and Design Rules

IRS schema files must adhere to the following naming conventions:

Rule #	Rule
SCF1	The schema file name of each IRS schema, that is not a code list, MUST be of the form: IRS-{Schema Name}-{Schema Version}.xsd For example, the proper name for the taxpayer schema will be IRS-Taxpayer-1.0.xsd, where 1.0 is the version number.
SCF2	The schema file name of each IRS maintained code list schema MUST be of the form: IRS-CodeList-{Code List Name}-{Code List Schema Version}.xsd For example, the proper name for the country code list will be IRS-CodeList-CountryCode-1.0.xsd, where 1.0 is the version number.
SCF3	Each internal schema module MUST be named {ParentSchemaModuleName}{InternalSchemaModuleFunction}{schema module}

206

207 #### 4.1.5.3 IRS XML Schema Organization Naming and Design Rules

208 In order to guarantee that each occurrence of an IRS conformant schema will have the same look
209 and feel, schema layouts must be consistent. Therefore, all IRS developed schemas must abide
210 by the following rules:

211

Rule #	Rule

Rule #	Rule
SCH1	All IRS developed schemas MUST be organized as follows: XML Declaration <!-- xsd:schema Element With Namespaces Declarations --> xsd:schema element to include version attribute and namespace declarations in the following order: xmlns:xsd Target namespace Default namespace CommonAggregateComponents CommonBasicComponents Datatypes Identifier Schemes Code Lists Attribute Declarations elementFormDefault=”qualified” attributeFormDefault=”unqualified” <!-- Imports --> CommonAggregateComponents schema module CommonBasicComponents schema module <!-- Global Attributes --> Global Attributes and Attribute Groups <!-- Root Element --> Root Element Declaration Root Element Type Definition <!-- Element Declarations --> alphabetized order <!-- Type Definitions --> All type definitions ordered by aggregates followed by basics (as below): <!-- Complex Type Definitions --> alphabetized order <!-- Simple Type Definitions --> alphabetized order

4.1.5.4 IRS XML Schema Modularity Naming and Design Rules

A schema module is a schema document containing type definitions and element declarations intended to be reused in multiple schemas. XML schemas can grow quite large. In addition to the logical taming of complexity that namespaces provide, dividing the physical realization of schemas into multiple files (schema modules) provides a mechanism whereby reusable components can be imported as needed without the need to import overly complex complete schemas.

When designing schemas, IRS developers should attempt to maximize reuse by:

- Re-using existing types and schemas.

- Extending or complementing existing standard types and schemas where they do not meet all your needs.
- Using Modular schema design:
  - Define a structure once that can be used in more than one schema. For example, if a particular subject matter is exactly the same in multiple return types, define the structure once in a generic subject matter schema as a local complex type and then incorporate this into various other schemas via an "include schemaLocation" declaration.
  - When performing analysis prior to schema design, take the broadest view practical, to anticipate modular, shareable fragments

All IRS developed schemas must abide by the following rules:

Rule #	Rule
SSM1	A document schema in one namespace that is dependent upon type definitions or element declarations defined in another namespace MUST import the document schema from that namespace.
SSM2	All internal schema modules MUST be in the same namespace as their corresponding document schema.
SSM3	IRS schema modules MUST either be treated as external schema modules or as internal schema modules of the document schema.

4.1.5.5 IRS XML Schema Import Naming and Design Rules

Because the IRS has committed to use URN identifiers for schema, the pointer to the actual schema files must be specified in the import statement.

Rule #	Rule
IMP1	The xsd:import element MUST contain the namespace and schema location attributes.
IMP2	An xsd:import element MUST be declared for every schema required by an IRS schema.
IMP3	Each schemaLocation attribute declaration MUST contain a persistent and resolvable URL.
IMP4	The schemaLocation attribute URL structure and physical storage location MUST be managed by EDMO.
IMP5	When externally developed schemas are copied locally into the IRS environment, they MUST be physically stored independently of IRS developed schemas. The schemaLocation attribute URL structure and physical storage location of these externally developed schemas will be managed by EDMO.

4.1.5.6 IRS XML Form Oriented Schema Naming and Design Rules

Form oriented schema are most applicable at the message level. Form oriented schema components should be avoided. Form oriented aggregate and basic components components are not readily compatible with component based standards.

Rule #	Rule
FRM1	When a form has multiple parts, for example, Section 1, Section 2, etc., a comment MUST be added that clearly delineates each part. Place the comment at the beginning of each part within the schema.
FRM2	When a field on the form has a preprinted fixed literal value and is not entered by the taxpayer, an element MUST NOT be created for it (the literal) in the schema.
FRM3	When literal values populate an element, they SHOULD be in uppercase characters.
FRM4	When the natural usage of a complex table (for example, Form 4684 Casualties and Thefts) assumes the population of one column before the next, then the XML elements in the schema SHOULD be sequenced Top-Down-Left-Right rather than Left-Right-Top-Down.
FRM5	A repeating group of elements SHOULD be created to replace paper attachments that are used to extend the limited space on the paper form (for example, a table showing only four rows but allows additional rows on a paper attachment).
FRM6	A field that is duplicated on the same document SHOULD be defined only once in the schema. However, a field that is a duplicate of a field on a different document SHOULD be defined in the schema.
FRM7	If a field is conditionally duplicate (that is, it will have a value if another field has a value) then an element for it SHOULD be created in the schema.
FRM8	The structure and sequencing of the elements in a form-based schema MAY resemble its paper counterpart. For example, elements for a simple table SHOULD be created in a sequence that each row is fully represented before the next row, i.e. Left-Right-Top-Down instead of Top-Down-Left-Right
FRM9	Boolean and Checkbox elements MUST be designated as Indictors (Ind), eg.g. AddressChangeInd.
FRM10	Names that tie to the layout or line number reference of a paper form MUST NOT be used. For example, do not use “Line 8” or “AddLine1ThruLine5”.
FRM11	Hard code values in an element name where the value is not static over time MUST NOT be used. As an illustration: do not use the year number (e.g., 2001) in the name of an element when it appears in the lavel on a form. Use “CurrentYear”, “PreviousYear”, “NextYear” labels instead. For example, label on line 6 on form 2800 for year 2001 reads “Passive Activity Credit allowed for year 2001”. Name the element “PassiveActivityCreditCYAmt” not “PassiveActivityCreditYear2001”.

Rule #	Rule
FRM12	XML element names for form based XML schemas should be based on the attribute name from the project logical data model, or ELDM, or ELDM XML element name.

4.1.6 Elements and Attributes

4.1.6.1 Element and Attribute Lexicon

Each XML document contains one or more XML elements. Each XML element is delimited by a set of matching start and end tags, called XML markup tags. Each XML element also has a type, identified by name, sometimes called its generic identifier.

An XML construct string, hereinafter referred to as an element, encodes the semantic meaning and structural information about data with the data itself. For the purposes of this IRS EDSG document, "XML element" and "XML tag" will not be used interchangeably. An XML element consists of a particular unit of character data surrounded by XML markup tags, describing the character data. XML markup tags are separated from the character data through the use of angle brackets. In the XML construct string below, the XML element content is "$500" and the two matching XML tags are "<TaxPaymentAmt>."

<TaxPaymentAmt>$500</TaxPaymentAmt>

An XML start tag may contain a set of XML attribute specifications. An XML attribute is a source of additional information about the XML element. XML attribute values may be fixed in the XML schema or listed as name/value pairs (for example, attribute name="value") in the start tag of a given XML element. While an XML attribute provides extra metadata to better understand an XML element, once an XML element has been made an XML attribute, it cannot be further extended or parsed.

<TaxPaymentAmt units="$">500</TaxPaymentAmt

4.1.6.2 IRS XML Element Naming and Design Rules

XML elements must adhere to the following set of standards:

Rule #	Rule
ELD1	Empty elements MUST NOT be declared.
ELD2	The xsd:any element MUST NOT be used.

4.1.6.3 IRS XML Attribute Naming and Design Rules

XML attributes must adhere to the following set of standards:

Rule #	Rule
ATD1	It is strongly RECOMMENDED that projects create global attributeGroups for multiple attributes.
ATD2	The xsd built-in nillable attribute MUST NOT be used for any declared element.
ATD3	The xsd:any attribute MUST NOT be used.
ATD4	All xsd:attribute declarations MUST specify the "use" as being optional or required.
ATD5	The use of attributes SHOULD be minimized as they cannot be extended or parsed, and limit interoperability and reuse.

4.1.7 XML Data Types

4.1.7.1 Data Type Lexicon

A common XML data type is an XML element specifically defined for reuse. This definition is the format used for the collection of alphanumeric, digits, and/or symbols that depict the permitted values of the XML element. An XML data type is appropriately stored in an XML registry along with other standardized XML components or products.

For a common XML data type, there is no structural difference between further classifying it as a local or a global XML data type. The only difference is the extent of use of the particular XML data type. A local XML data type is used one or more times on the same XML schema. A global XML data type is applicable across schemas, and is stable and universal in nature.

The XML registry may contain simple XML data types, complex XML data types, and XML data type facets. A user-generated data type can be created by adding constraining facets to a primitive data type. Examples of simple XML data types include string, byte, integer, Boolean, date, and name. Complex data types are generally collections of simple data types.

4.1.7.2 IRS XML Simple Type Naming and Design Rules

XML simple types must adhere to the following set of standards:

Rule #	Rule
SIM1	For each simpleType an xsd:restriction element MUST be declared.
SIM2	For each simpleType xsd:restriction element, an xsd:base attribute MUST be declared and set to the appropriate xsd: built-in data type.
SIM3	Built-in XSD simple types SHOULD be used whenever possible.
SIM4	Use of an IRS defined type SHOULD be used whenever possible.

4.1.7.3 IRS XML Complex Type Naming and Design Rules

XML complex types must adhere to the following set of standards:

Rule #	Rule
COM1	For every project logical data model and ELDM data class, a named xsd:complexType MUST be defined. A corresponding global element for this complex type MUST also be declared.
COM2	Every xsd:complexType definition MUST use the xsd:sequence, xsd:simpleContent, or xsd:choice for its contents. The xsd:choice element SHOULD NOT be used where customization and extensibility are a concern.
COM3	Every common basic component datatype MUST have xsd:base set to a specialized datatype, an unspecialized datatype, or an xsd:built-in datatype.
COM4	For the simpleContent complex types, the simpleContent element MUST contain one xsd:extension element. This extension element MUST have its xsd:base element set to a specialized datatype, an unspecialized datatype, or an xsd:built-in datatype.

4.1.8 General XML Constructs

4.1.8.1 IRS XML General Construct Naming and Design Rules

XML schemas must adhere to the following set of standards:

Rule #	Rule
GXS1	All schema modules SHOULD be created with reuse as an objective.
GXS2	The xsd:SubstitutionGroups feature MUST NOT be used, with only one exception which is in the case of Code List schemas.
GXS3	The xsd:final attribute MUST be used to control extensions.
GXS4	xsd:notations MUST NOT be used.
GXS5	xsd:all MUST NOT be used
GXS6	The xsd:include feature MUST only be used within a document schema.
GXS7	The xsd:union technique MUST NOT be used except for Code Lists. The xsd:union technique MAY be used for Code Lists.
GXS8	IRS schemas SHOULD NOT use xsd:appinfo. If used, xsd:appinfo MUST only be used to convey non-normative information.
GXS9	Complex Type extension or restriction MAY be used where appropriate.
GXS10	IRS artifacts MUST use double quotes (“) to delimit attribute values rather than single quotes (‘).

4.1.8.2 IRS XML General Naming Conventions Naming and Design Rules

XML artifacts must be named in accordance with the following set of standards:

Rule #	Rule
GNR1	Application of IRS data naming standards are REQUIRED for all XML type names.
GNR2	XML component names MUST NOT be developed using database object names. XML component names are independent from database object names.
GNR3	IRS data naming standards MUST be applied to all XML element names.
GNR4	Meaningful, readable element and attribute names MUST be used. Avoid the use of terse, abbreviated names; they are difficult to understand and subject to misinterpretation.
GNR5	All complexTypes and simpleTypes MUST have the word "Type" appended to the end of the name. Elements declared as being of these MUST have the word "Type" dropped from the element name.
GNR6	Global elements whose type is set to a defined complexType SHOULD use the same name as the complexType with the word "Type" dropped.
GNR7	Spaces, punctuation, and other non-alphanumeric characters, including colon (":"), period (“.”), or underscore (”_”) MUST not be used.
GNR8	Ampersands (“&”) in element names MUST not be used. Either omit or replace with the word “And” if needed.
GNR9	In both the start tag and end tag, each word of the XML element name MUST be capitalized(upper Camel case).
GNR10	Lower Camel case MUST be used for an attribute name.
GNR11	When abbreviations are necessary, the authorized abbreviations for logical data model classes and attributes in Appendix J of this IRS EDSG document MUST be used.
GNR12	All element, attribute and type names MUST be in the singular form unless the concept itself is plural.
GNR13	All types MUST be named.
GNR14	Names MUST NOT begin with a number.

4.1.9 XML Artifact Versioning

The last part of the namespace name is called the document-id. Versioning information will be included as part of the document-id component. The version information is divided into major and minor fields.

The major version is “1” for the first release of a namespace. Subsequent major releases increment the value by 1. The IRS mandates that the major-version field of a namespace URN be incremented when a change occurs that breaks compatibility with the previous release of that namespace. If a change does not break compatibility, then only a minor version change is needed.

A minor revision may define new complex types by (1) extending, (2) restricting type or (3) by direct composition. It must not use the XSD redefine element to change the definition of a type or element from the previous version.

Similar to the schema versioning rules, versions are also tracked at the XML artifact, or component, level. Each component will begin with a default version number of 1.0 and will go through major and minor versions as the component changes. A change that breaks backward compatibility will require a major version change. A change that keeps backward compatibility in tact will not require a major version change. A change to documentation will not result in a version increment.

4.1.9.1 IRS XML Schema Versioning Naming and Design Rules

Rule #	Rule
VER1	Every major version release of an IRS schema or schema module MUST have its unique namespace end with: major-number.0 (where the zero represents the minor version which is always set to zero when a major revision is incremented)
VER2	Every schema and schema module major version number MUST be sequentially assigned, incremental number greater than zero
VER3	The first minor version release of an IRS schema or schema module MUST have its unique namespace end with: major-number.non-zero
VER4	For minor version changes, the name of the version construct MUST NOT change (short name not qualified name), unless the intent of the change is to rename the construct.
VER5	Every schema and schema module minor version number MUST be a sequentially assigned, incremental non-negative integer.
VER6	A minor version document schema MUST import its immediately preceding minor version document schema.
VER7	Schema and schema module minor version changes MUST be limited to the use of xsd:extension or xsd:restriction to alter existing types or add new constructs.
VER8	Schema and schema module minor version changes MUST not break semantic compatibility with prior versions.

4.1.9.2 IRS XML Component Versioning Naming and Design Rules

Rule #	Rule
CVR1	Every component major version number MUST be sequentially assigned, incremental number greater than zero
CVR2	The first minor version release of an IRS component MUST be numbered: major-number.non-zero

CVR3	For minor version changes, the name of the component MUST NOT change, unless the intent of the change is to rename the component.
CVR4	Every component version number MUST be a sequentially assigned, incremental non-negative integer.
CVR5	Component minor version changes MUST not break semantic compatibility with prior versions.

4.1.10 XML Instance Documents

4.1.10.1 IRS XML Instance Document Naming and Design Rules

IRS developed XML instance documents must adhere to the following set of standards:

Rule #	Rule
IND1	All instance documents MUST validate to a corresponding schema.
IND2	All IRS XML SHOULD be expressed using UTF-8.
IND3	All instance documents MUST always identify their character encoding with the XML declaration.
IND4	All instance documents MUST contain the following namespace declaration in the root element: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
IND5	Instance documents SHOULD NOT contain an element devoid of content or null values.
IND6	The absence of a construct or data in a instance document MUST NOT carry meaning.

4.1.11 XML Code Lists

Code lists capture an enumeration of valid values that represent abstract or concrete concepts in a compact form (typically an abbreviation of some kind).Code lists can promote interoperability by constraining the meaning of enumerated values to mutually agreed concepts between trading partners. Examples of standardized code lists are country codes and currency codes.

4.1.11.1 IRS XML Code List Naming and Design Rules

Code list schemas must adhere to the following set of standards:

Rule #	Rule
CDL1	All IRS Codes MUST be part of an IRS or externally maintained Code List.
CDL2	The IRS SHOULD identify and use external standardized code lists rather than develop its own code lists.

CDL3	The IRS MAY design and use an internal code list where an existing external code list needs to be extended, or where no suitable external code list exists.
CDL4	The IRS maintained code lists MUST be enumerated using the standardized IRS Code List schema format.
CDL5	Each code list schema MUST have its own unique namespace.
CDL6	An xsd:Import element MUST be declared for every code list required in an IRS schema.

345

346 ### 4.1.12 XML Artifact Documentation

347 #### 4.1.12.1 IRS XML Artifact Documentation Naming and Design Rules

348 XML schemas must adhere to the following set of standards:

349

Rule #	Rule
DOC1	Use of the IRS standard documentation parameters is REQUIRED. These are defined in the schema "DocumentationParameters-[major].[minor].xsd".
DOC2	The standard prefix for the IRS standard documentation tags MUST be "irsdoc".
DOC3	Use of comment (<!-- -->) tags MUST NOT be used for schema documentation. All documentation MUST be put inside <xsd:annotation><xsd:documentation> tags.
DOC4	Schema level documentation MUST immediately follow the <xsd:schema> tag.
DOC5	Type documentation MUST immediately follow the type's opening tag. It MUST precede the content definition portion of the type definition.
DOC6	Global elements documentation is OPTIONAL; but documentation is required at the type level.
DOC7	The documentation element for each major version MUST only reflect the documentation for the current version.
DOC8	Each minor version MUST be documented. The documentation MUST proceed and include all previous minor version documentation.

350

351 #### 4.1.12.2 IRS XML Artifact Documentation Parameter Element Definitions

352 XML artifacts require certain tags to facilitate reuse, interoperability, and management of XML
353 in the IRS. The table below contains rules for each type of documentation parameter element.
354 The Status column defines if the metadata element is optional or mandatory, along with other
355 information pertinent to the element.

356

Element	Purpose	DataType	Status
Dictionary Entry Nm	The name of the Basic Component	Text	Mandatory

Element	Purpose	DataType	Status
Description Txt	A text string explaining what the Basic Component is rather than how it does something.	Text	Mandatory
Major Version Num	Identifies the major version number of the artifact	Numeric	Mandatory
Minor Version Num	Identifies the minor version number of the artifact.	Numeric	Mandatory
Version Number Description Txt	A text string that describes the reason for the version number change.	Text	Mandatory
Version Effective Begin Dt	The date the version for the artifact is effective.	Date	Mandatory
Version Effective End Dt	The date the version for the artifact is no longer effective.	Date	Optional

The DocumentationParameters schema defines the following documentation parameters:

```
<!-- ===== Element Declarations ===== -->
<xsd:element name="Component" type="irsdoc:ComponentType"/>
<!-- ===== Type Definitions ===== -->
<xsd:complexType name="ComponentType">
      <xsd:sequence>
            <xsd:element ref="DictionaryEntryNm"/>
            <xsd:element ref="MajorVersionNum"/>
            <xsd:element ref=”MinorVersionNum”/>
            <xsd:element ref="VersionEffectiveBeginDt"/>
            <xsd:element ref="VersionEffectiveEndDt" minOccurs="0"/>
            <xsd:element ref="VersionDescriptionTxt"/>
            <xsd:element ref="DescriptionTxt"/>
      </xsd:sequence>
</xsd:complexType>
<!-- ===== Element Declarations ===== -->
<xsd:element name="DictionaryEntryNm" type="xsd:normalizedString"/>
<xsd:element name="MajorVersionNum" type="xsd:normalizedString"/>
<xsd:element name="MinorVersionNum" type="xsd:normalizedString"/>
<xsd:element name="VersionEffectiveBeginDt" type="irssdt:DateType"/>
<xsd:element name="VersionEffectiveEndDt" type="irssdt:DateType"/>
<xsd:element name="VersionDescriptionTxt" type="xsd:string"/>
<xsd:element name="DescriptionTxt" type="xsd:string"/>
```

Sample implementations of the documentation rules follow below:

For schema level documentation:

```
<xsd:annotation>
      <xsd:documentation>
            <irsdoc:Component>
                  <irsdoc:DictionaryEntryNm>
                        Common Aggregate Components Schema
                  </irsdoc:DictionaryEntryNm>
                  <irsdoc:MajorVersionNum>1
```

```
                    </irsdoc:MajorVersionNum>
                    <irsdoc:MinorVersionNum>0
                    </irsdoc:MinorVersionNum>
                       <irsdoc:VersionEffectiveBeginDt>2000-07-01
                       </irsdoc:VersionEffectiveBeginDt>
                    <irsdoc:VersionEffectiveEndDt>2006-12-22
                    </irsdoc:VersionEffectiveEndDt>
                    <irsdoc:VersionDescriptionTxt>
                          Original Definition
                    </irsdoc:VersionDescriptionTxt>
                    <irsdoc:DescriptionTxt>
                          Aggregate types based on ELDM version 2.0
                    </irsdoc:DescriptionTxt>
              </irsdoc:Component>
        </xsd:documentation>
   </xsd:annotation>
```

For simple types (including those in the SpecializedDataTypes.xsd):

```
   <xsd:simpleType name="CityType">
        <xsd:annotation>
              <xsd:documentation>
                    <irsdoc:Component>
                          <irsdoc:DictionaryEntryName>City
                          </irsdoc:DictionaryEntryName>
                          <irsdoc:MajorVersionNum>1
                          </irsdoc:MajorVersionNum>
                          <irsdoc:MinorVersionNum>0
                          </irsdoc:MinorVersionNum>
                          <irsdoc:VersionEffectiveBeginDt>2000-07-01
                          </irsdoc:VersionEffectiveBeginDt>
                          <irsdoc:VersionEffectiveEndDt>2006-12-22
                          </irsdoc:VersionEffectiveEndDt>
                          <irsdoc:VersionNumDescriptionTxt>
                               Original Definition
                          </irsdoc:VersionNumDescriptionTxt>
                          <irsdoc:DecsriptionTxt>
                                Used for a city. Legal Characters: A-Z,
                                a-z, and single space.  Illegal
                                Character: leading space,
                                trailing space, adjacent spaces,
                                 and symbols.
                          </irsdoc:DescriptionTxt>
                    </irsdoc:Component>
              </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
              <xsd:maxLength value="22"/>
              <xsd:pattern value="([A-Za-z] ?)*[A-Za-z]"/>
        </xsd:restriction>
   </xsd:simpleType>
```

For complex types of simple content (including those in the CommonBasicComponents.xsd):

```
   <xsd:complexType name="CityType">
```

```
            <xsd:annotation>
                  <xsd:documentation>
                  <irsdoc:Component>
                        <irsdoc:DictionaryEntryName>
                              City Type
                        </irsdoc:DictionaryEntryName>
                        <irsdoc:MajorVersionNum>1
                        </irsdoc:MajorVersionNum>
                        <irsdoc:MinorVersionNum>0
                           </irsdoc:MinorVersionNum>
                        <irsdoc:VersionEffectiveBeginDt>2000-07-01
                        </irsdoc:VersionEffectiveBeginDt>
                        <irsdoc:VersionEffectiveEndDt>2006-12-22
                        </irsdoc:VersionEffectiveEndDt>
                        <irsdoc:VersionDesscriptionTxt>
                              Original Definition
                        </irsdoc:VersionDescriptionTxt>
                        <irsdoc: DescriptionTxt>
                              A City name
                        </irsdoc:DescriptionTxt>
                  </irsdoc:Component>
                  </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                  <xsd:extension base="irssdt:CityType"/>
            </xsd:simpleContent>
      </xsd:complexType>
```

For complex types with sequence elements (including those in CommonAggregateComponents.xsd):

```
      <xsd:complexType name="USAddressType">
            <xsd:annotation>
                  <xsd:documentation>
                  <irsdoc:Component>
                        <irsdoc:DictionaryEntryName>
                                    US Address Type
                        </irsdoc:DictionaryEntryName>
                        <irsdoc: MajorVersionNum>2
                        </irsdoc:MajorVersionNum>
                        <irsdoc: MniorVersionNum>0
                        </irsdoc:MinorVersionNum>
                        <irsdoc:VersionEffectiveBeginDt>2000-07-01
                        </irsdoc:VersionEfectiveBeginDt>
                        <irsdoc:VersionEffectiveEndDt>2006-12-22
                        </irsdoc:VersionEffectiveEndDt>
                        <irsdoc:VersionDescriptionTxt>Update of version 1.0
                  to add a third street address line
                        </irsdoc:VersionDescriptionTxt>
                        <irsdoc:DescriptionTxt>
                                    Used for a formatted US Address
                        </irsdoc:DescriptionTxt>
                  </irsdoc:Component>
                  </xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
```

```
                <xsd:element ref="irscbc:AddressLine1"/>
                <xsd:element ref="irscbc:AddressLine2" minOccurs="0"/>
                <xsd:element ref="irscbc:City"/>
                <xsd:element name="State" type="irsState:StateCodeType"/>
                <xsd:element ref="ZipCd"/>
            </xsd:sequence>
        </xsd:complexType>
```

4.1.13 XML Stylesheets

4.1.13.1 Stylesheet Lexicon

Any discussion of XML component lexicon would not be complete without also addressing an XML stylesheet. This final XML component completes the functionality started by an XML schema.

Like an object class, an XML schema describes the content of a set of XML instances, or documents. An XML schema does not describe how these XML instances, or documents, are presented on screens. To fill this presentation void, an XML schema requires a corresponding XML stylesheet. An XML stylesheet formally describes how XML content should be styled, laid out, and paginated on presentation medium such as a Web browser window.

An Extensible Stylesheet Language (XSL) stylesheet transforms a class of XML instances, or documents, into HTML documents on a Web server without adding new HTML tags. An XSL stylesheet, identified by xsl:, is another XML component that may also be found in an XML namespace and is considered to be an information resource object in an XML registry,

4.1.13.2 IRS XML Stylesheet Naming and Design Rules

To be developed.

Appendix A. IRS XML NDR Rule-UBL Rule Mapping

The following table contains a mapping of the IRS XML NDR rules to the UBL XML NDR rules. It can be used as a reference for determining the origination of an IRS XML NDR rule.

Ledgend:

- IRS Rule is the IRS XML NDR Rule identifier
- External Ref identifies the corresponding UBL XML NDR rule(s) from which the IRS rule was derived. N/A indicates the IRS XML NDR rule is either new, or it was derived from a prior IRS XML Standards & Guidelines rule.

IRS Rule	UBL Rule
ATD1	UBL ATD2
ATD2	UBL ATD7
ATD3	UBL ATD8 UBL GTD2
ATD4	UBL CTD13
ATD5	N/A
CDL1	UBL CDL1
CDL2	UBL CDL2
CDL3	UBL CDL3
CDL4	UBL CDL5
CDL5	UBL CDL6
CDL6	UBL CDL7
COM1	UBL CTD1 UBL ELD3
COM2	UBL CTD2 UBL GXS9
COM3	UBL CTD5
COM4	UBL CTD9
CVR1	UBL VER6
CVR2	UBL VER4
CVR3	UBL VER5
CVR4	UBL VER7
CVR5	UBL VER11
DOC1	N/A
DOC2	N/A
DOC3	N/A
DOC4	N/A
DOC5	N/A
DOC6	N/A
DOC7	N/A
DOC8	N/A
ELD1	UBL ELD7

IRS Rule	UBL Rule
ELD2	UBL ELD8 UBL GTD2
FRM1	N/A
FRM2	N/A
FRM3	N/A
FRM4	N/A
FRM5	N/A
FRM6	N/A
FRM7	N/A
FRM8	N/A
FRM9	N/A
FRM10	N/A
FRM11	N/A
FRM12	N/A
GNR1	N/A
GNR2	N/A
GNR3	UBL ATN1
GNR4	UBL ATN1
GNR5	UBL CTN1
GNR6	UBL ELN1
GNR7	UBL CTN5 UBL GNR3
GNR8	UBL CTN5 UBL GNR3
GNR9	UBL GNR3 UBL GNR8
GNR10	UBL GNR3 UBL GNR9
GNR11	UBL GNR4 UBL GNR5 UBL GNR6
GNR12	UBL GNR7
GNR13	UBL GTD1
GNR14	N/A
GXS1	UBL SSM8
GXS2	UBL GXS5
GXS3	UBL GXS6
GXS4	UBL GXS7
GXS5	UBL GXS8
GXS6	UBL GXS10
GXS7	UBL GXS11
GXS8	UBL GXS12
GXS9	UBL GXS13
GXS10	N/A
IMP1	UBL ELD6
IMP2	UBL CDL7

IRS Rule	UBL Rule
IMP3	UBL ATD5
IMP4	N/A
IMP5	N/A
IND1	UBL IND1
IND2	UBL IND2
IND3	UBL IND3
IND4	UBL IND4
IND5	UBL IND5
IND6	UBL IND6
NMP1	N/A
NMP2	UBL NMS8
NMP3	UBL NMS10
NMP4	UBL NMS17
NMS1	UBL NMS2
NMS2	N/A
NMS3	UBL CDLXX
NMS4	UBL NMS3
NMS5	UBL NMS2
NMS6	UBL CDLX
NMS7	UBL NMS6
NMS8	N/A
NMS9	UBL GXS4
NMS10	UBL NMS1
NMS11	UBL NMS7
SCF1	N/A
SCF2	UBL NMS19 UBL CDL6
SCF3	UBL SSM7
SCH1	UBL GXS1
SIM1	UBL ELD5
SIM2	UBL ATD4
SIM3	UBL GXS3
SIM4	UBL GXS3
SSM1	UBL SSM2
SSM2	UBL SSM6
SSM3	UBL SSM5
STA1	UBL STA1
STA2	UBL STA2
STA3	N/A
STA4	N/A
VER1	UBL VER2
VER2	UBL VER6
VER3	UBL VER4
VER4	UBL VER5
VER5	UBL VER7

IRS Rule	UBL Rule
VER6	UBL VER9
VER7	UBL VER10
VER8	UBL VER11

Appendix B. Glossary

Approved Status: A data model class or attribute in the EDD considered to be the most authoritative data source for enterprise-wide use. It must be used in its present format if it will fully satisfy new data requirements. If not, a modified version of this data element must be submitted to the IRS Enterprise Data Management Project Office (EDMPO).

Archived Status: A data model class or attribute maintained in the EDD purely as an historical reference of a data element that satisfied earlier data requirements. It has been replaced by another Approved data element, as a result of the continual data improvement process, and should not be recreated at this time.

Candidate Status: Compared to Approved status, a data model class or attribute in the EDD in Candidate status is considered to be the second most authoritative data source. It can be used in its present format for enterprise-wide use with a high degree of confidence.

Collection: A DB2-unique term for an identified group of packages. When binding a package, the collection to which the package belongs is specified.

Column: A set of data values of the same type collected and stored in the rows of a table.

Context Data: Existing Approved data model classes and attributes appearing on a new conceptual or logical data model because they satisfy the new data requirements of this data model.

Current Production Environment (CPE) Data: Data elements from the As Is, or legacy, Internal Revenue Service (IRS) information systems.

Database: A set of table spaces and index spaces.

Database Request Module (DBRM): A DB2-unique term for a collection of SQL statements extracted from the source program. This module is created during application program compilation and communicated to DB2 during the bind process.

Deprecated Generic Element/Class Word: A legacy system IRS class word that is permanently grandfathered, but only for its current use in existing attributes. This avoids the need for funding a change request to make retroactive changes. A preferred approved class word should instead be used for all new attributes.

Development Status: A data model class or attribute in the EDD in the initial stage of development. It may be used to satisfy a new data requirement but with the understanding that the data element is not yet mature.

Any data model class, or attribute, is first developed before being given a status code. The normal status code progression cycle is Developmental, Candidate, Approved, and Archived.

Document Schema (XML): The overarching schema within a namespace that conveys the business document functionality of that namespace. The document schema declares a target namespace and is likely to pull in by including internal schema modules or importing external schema modules. Each namespace will have one, and only one, document schema.

Document Type Definition (DTD): The defined rules, or permitted markup, for a particular Extensible Markup Language (XML) application. A DTD is required for a valid XML document.

Element: A particular unit of character data surrounded by markup, or tags, describing the character data in an Extensible Markup Language (XML) document.

Engineering Data: The To Be data elements found in the IRS modernization systems data dictionaries and data models.

Enterprise Conceptual Data Model (ECDM): The first enterprise level data model developed. It is intentionally small, showing major classes, key attributes, and primary relationships. On the ECDM, it is not necessary to show all attributes or to fully describe the attributes that are shown.

Engineering Data Dictionary (EDD): One of the components of the Enterprise Metadata Repository (EMR) along with the enterprise conceptual, logical, and physical data models and other information. The EDD holds metadata, a complete description of the classes and attributes in these data models, at a sufficient level of detail to ensure that they are discrete and clearly understood. The EDD is the definitive source for data meaning, representation, and historical data structure changes by projects assigned development responsibility.

Enterprise Logical Data Model (ELDM): The second enterprise level data model developed. It is the result of merging a Project Logical Data Model (PLDM), such as CADE Milestone 3, into the existing Enterprise Conceptual Data Model (ECDM). The ELDM extends the ECDM level of detail.

Electronic Business using eXtensible Business Markup Language (ebXML): Sponsored by UN/CEFACT and OASIS, is a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet.

Extensible Markup Language (XML): A meta-markup language for describing data elements that is designed to be extended because it does not have a fixed set of tags and elements.

Extensible Stylesheet Language (XSL): A standard from the W3C for describing a style sheet for XML documents.

Enterprise Data Management Office (EDMO): The IRS organization whose mission is to manage data as a corporate asset and centrally oversees its design, architecture, implementation and usage through the development of Enterprise Data Models and Enterprise Data Architecture, and through the publication and enforcement of data management policies, standards and procedures at both the Enterprise and project levels.

Enterprise Data Standards and Guidelines (EDSG): The name of this document which is Attachment 1 to the Enterprise Standards Profile of the IRS Enterprise Architecture.

Index: The only DBMS directory listing maintained by DB2. This consists of an ordered set of pointers to rapidly locate table row records based on the values of data in one or more key columns.

Internal Schema Module (XML): A schema that is part of a schema set within a specific namespace.

Location: A DB2-unique term for an instance of a DBMS at a particular place.

Objective Data Dictionary Tool: A follow-on to the interim configuration of the Engineering Data Dictionary (EDD) established in April 2001. Compared to the interim EDD that satisfies

two-thirds of the data dictionary requirements, the objective data dictionary tool should satisfy nearly all of these requirements.

Package: A DB2-unique term for a database program consisting of one or more procedures. A package is produced during program preparation and contains control structures used to execute SQL statements.

Plan: A DB2-unique term, synonymous with application plan, for a high-level strategy specifying processing options for executing SQL commands. It contains a list of package names and/or the bound form of SQL statements from one or more DBRMs. Every DB2 application requires an application plan.

Procedure: A database application program executed by DB2 in response to a single call statement. A procedure can either be contained within a package or can stand alone.

Project Logical Data Model (PLDM): Created by a project by adding attributes to the classes in its assigned Enterprise Conceptual Data Model (ECDM) subject area. If required, the project also adds descendant classes and attributes to the PLDM.

Project Physical Data Model (PPDM): Created by project from Project Logical Data Model (PLDM) for physical implementation.

Resource Access Control Facility (RACF) Group: A DB2-unique term, typically used in conjunction with an authorization ID, for a team that is the owner or qualifier for a table. This team has specific responsibilities, authorities, and privileges. Database access securities are verified by a user ID and password.

Rejected Status: A data model class or attribute in the EDD that should not be considered because it never adequately satisfied new data requirements. Reasons for rejection may be duplicate, purely physical, application-unique, derived, or obsolete. This data element is retained in the EDD so that it will not accidentally be re-created.

Schema (DB2): A composite object that is a collection of other discretely named objects assigned to a schema when created by a database user. The objects comprising a schema may include distinct types, functions, stored procedures, and triggers.

Schema (XML): A definition, written in Extensible Markup Language (XML) syntax, of constraints for the content type and data type of XML tags.

Schema Module (XML): A schema document containing type definitions and element declarations intended to be reused in multiple schemas.

Schema Set (XML): A collection of schema instances that together comprise the names in a specific UBL namespace.

Storage Group: A DB2-unique term for a list of Direct Access Storage Devices (DASDs) on which DB2 can allocate data sets for associated storage structures. This association between storage structure and storage group is explicitly or implicitly defined.

Table: A set of related columns and rows in a relational database. A table is a logical structure maintained by DB2.

Table Space: A portion of a database reserved for where a table will go. Table structure is the mapping of tables into table spaces.

Tag: The markup portion of an Extensible Markup Language (XML) element surrounding the character data. The name of the tag reflects the content inside the XML element.

Transitional Data: Data elements between those found in the Current Production Environment (CPE), or legacy, IRS information systems and the engineering data elements.

Trigger: A set of actions that are executed when a delete, insert, or update operation occurs on a specified table. The trigger is activated when the SQL operation is executed.

Uniform Resource Identifier (URI): The addressing technology for identifying resources on the Internet or a private intranet.

Uniform Resource Locator (URL): The address that defines the route to a file on a Web server.

Uniform Resource Name (URN): A name that identifies a resource on the Internet. Unlike URLs, which use network addresses (domain, directory path, file name), URNs use regular words that are protocol and location independent.

Universal Business Language (UBL): An XML specification that defines a set of rules (XML schemas) for drawing up electronic messages. It is closely related to the ebXML standard,

Valid: A well-formed Extensible Markup Language (XML) document that also matches the Document Type Definition (DTD).

Version: An identifier that is assigned to a package when the package is created. This identifier is a parameter of the DB2 pre-compilation and is associated with the program being bound.

View: An alternative way of looking at the data in one or more tables. A view is a named query in the database that can be used as if it were a table. A view changes the outward appearance of a table without actually changing the data in the table.

Well-formed: An Extensible Markup Language (XML) document that has sufficiently specific grammar to be read and understood by an XML parser.

World Wide Web Consortium (W3C): An international industry consortium founded in 1994 to develop standards for the Web. The W3C has standardized many of the fundamental technologies of the Web, including HTML and XML, URLs and URIs, the SOAP protocol and the P3P privacy description