



SURFACE VEHICLE STANDARD

J2630™

JUL2019

Issued 2003-12
Cancelled 2019-07

Superseded by ITU-T Rec. X.693 |
ISO/IEC 8825-4

Converting ATIS Message Standards from ASN.1 to XML

RATIONALE

SAE J2630 provided a means for converting advanced traveler information systems (ATIS) messages from abstract syntax notation (ASN) into extensible markup language (XML). Since its publication, SAE J2630 has been totally superseded by ITU-T Rec. X.693 | ISO/IEC 8825-4, a global standard for encoding ASN-defined objects with XML or Extended XML Encoding Rules (XER and EXER encodings, respectively). Future SAE standards should use XER or EXER encoding in lieu of SAE J2630.

CANCELLATION NOTICE

This Technical Report has been declared "CANCELLED" as of July 2019 and has been superseded by ITU-T Rec. X.693 | ISO/IEC 8825-4. By this action, this document will remain listed in the respective index, if applicable. Cancelled Technical Reports are available from SAE.

SAE Technical Standards Board Rules provide that: "This report is published by SAE to advance the state of technical and engineering sciences. The use of this report is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom, is the sole responsibility of the user."

SAE reviews each technical report at least every five years at which time it may be revised, reaffirmed, stabilized, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2019 SAE International

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

TO PLACE A DOCUMENT ORDER: Tel: 877-606-7323 (inside USA and Canada)
Tel: +1 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
<http://www.sae.org>

SAE WEB ADDRESS:

Full document details: [SAE J2630_201907](http://standards.sae.org/J2630_201907)

SAE values your input. To provide feedback on this Technical Report, please visit http://standards.sae.org/J2630_201907

Documents
Document Sharing Hub

Foreword—This document provides a set of conversion rules and recommendations which can be applied to ASN.1 to produce XML schema outputs. These rules were developed primarily for use with the ITS message sets developed by SAE and other standards organizations. However, these rules may also apply to other ITS message sets developed by SAE and other standards organizations. The results of applying these rules (a set of XML schema encodings) is published elsewhere (typically with the source standard). In the case of SAE standards the results of applying these rules can be found in the revised edition of SAE J2354.

TABLE OF CONTENTS

1.	Scope	2
2.	References	3
3.	Definitions.....	4
4.	Rules for Converting an ASN.1 Message Set Definition to an XML Schema.....	5
4.1	Defining a Simple Type in Terms of a Basic Type.....	6
4.2	Defining Basic INTEGER Types	6
4.3	Defining INTEGER Types with Minimum and Maximum Values.....	7
4.4	Defining String Types of Minimum and Maximum Length	7
4.5	Defining String Types of Fixed Length.....	8
4.6	Defining ENUMERATED Types	9
4.7	Defining a Type Definition Embedded in an Element Definition	11
4.8	Defining a Complex Type as a SEQUENCE of Elements.....	11
4.9	Defining a Complex Type as a CHOICE of Elements.....	12
4.10	Defining a Complex Type as a SEQUENCE OF Named Types.....	12
4.11	Defining a Complex Type as a SEQUENCE OF a Named Type with Minimum and Maximum Number of Elements.....	13
4.12	Defining a Complex Type as a SEQUENCE OF an Unnamed Type.....	13
4.13	Defining Elements as Instances of Existing Basic Types	14
4.14	Defining Elements as Instances of User-Defined Types.....	14
4.15	Defining Elements as Instances of In-Line Defined Types	15
4.16	Defining BIT STRING Elements	16
4.17	Defining OCTET STRING Elements.....	17
4.18	NULL Elements	19
4.19	Object Identifier (OID) Elements	19

4.20	Optional Elements	19
4.21	Handling Comments	20
4.22	Commonly Repeated Element	20
4.23	Defining an Element of a Type Defined in a Different Schema	21
4.24	Defining Top-Level Elements	21
4.25	Beginning and Ending the Schema	22
5.	Discussion of Selected Conversion Rules	22
5.1	Methods of Defining ENUMERATED Elements	22
5.2	Alternate Methods of Defining Mixtures of Text Strings and ENUMERATED Elements	24
5.3	Alternate Method of Defining BIT STRINGs	26
5.4	Treatment of IA5String Contents	27
5.5	Treatment of OCTET STRINGs	27
5.6	Treatment of Manual ASN.1 Tag	27
5.7	Treatment of Long ASN.1 Names	28
5.8	Treatment of ASN.1 Comments	28
5.9	Use of XML Namespaces to Point to Types Defined in Other Standards	29
5.10	Expressing the Resulting Schema in a Standard	31
6.	New Types Defined for the SAE ATIS J2353/2354 Schema	31
6.1	Binary Type	32
6.2	OctetString Type	32
6.3	NumericString Type	32
6.4	BoundedString Type	33
6.5	UndefinedElement Type	33
7.	A Complete Example of the Rules in Use	34
8.	BIBLIOGRAPHY	40
TABLE 1	XML TRANSLATION BASIC TYPES	6
TABLE 2	XML TRANSLATION RECOMMENDATIONS	6
TABLE 3	RANGES FOR XML INTEGER TYPES	6

- 1. Scope**—This SAE Standard presents a set of rules for transforming an Abstract Syntax Notation (ASN.1) message set definition into an eXtensible Markup Language (XML) schema. The result is intended to be a stand-alone XML Schema that is fully consistent with an existing ASN.1 information model. This is a different goal from other related work by other standards bodies developing a set of XML encoding rules for ASN.1 or ASN.1 encodings for XML Schema. These rules were initially developed in order to produce an XML schema for the SAE ATIS standard. While other standards may also choose to use these rules, the rules may not be applicable for all environments.

The goal for these transformation rules is twofold. The first goal is to provide a uniform set of such rules that all interested parties can use. The second goal is to use such rules to define an adopted schema for traveler information that reflects the preferred translation of ASN.1 message sets to XML for use by ITS system implementers. The first goal is met by this document. The second goal is met by employing this document to produce XML information as part of the periodic re-balloting of the SAE ATIS standard. This is a parallel standards effort with this document.¹

1. At this time a number of technical data formatting issues are being resolved as part of the re-issue of SAE J2354 (drafts available at the committee forum site <http://www.SAE.org/>). The XML productions will be included in this parallel effort. SAE J2353 and the message set contents of other ITS standards are being integrated within this single standard to produce a composite data dictionary of all SAE ITS messages.

These rules were developed as part of the process to draft an XML version of the ATIS data element and message set standards. The original effort focused upon the needs found in the currently adopted October 2000 SAE J2354 standard for ATIS message sets. SAE J2354 makes extensive use of elements from the ITETMDD work and from ITE-TCIP work. In addition, the draft “Event Report Message” (ERM) portion of the Message Sets for External Traffic Management Center Communication (MS/ETMCC) standard was also examined and translated. Message and data elements from the IEEE Incident Management standard (IEEE 1512-2000) were also examined. By this effort, every major message set of ITS was considered to some degree to ensure that the resulting translations could be successfully employed by others toward a common result. It is intended that the resulting ATIS schema will be voted on as an SAE ATIS standard. Other standards-developing organizations may also choose to use these rules; however, it is recognized that translations may not be necessary in some environments and that these rules may not be applicable to all environments within ITS. It is left to each standards-development organization to make such determinations for their specific environments.

Section 4 is the complete set of conversion rules. An example of use for each rule is given, showing an original ASN.1 definition, the resulting XML schema definition, and a sample XML document element where applicable.

Section 5 of this document discusses the background of some non-obvious conversion rules presented in Section 4.² These include methods of handling the XML representation of enumerations, octet strings, and bit strings. Section 5 also describes how XML namespaces can be used to point to types defined in the Traffic Management Data Dictionary (TMDD), the Transit Communications Interface Profiles (TCIP) standard, the Location Reference Message Set (LRMS), or the International Traveler Information Systems (ITIS) phrase lists. It also discusses additional translation refinements proposed for the conversion of the ATIS message standard.

Some omissions in the SAE standards required new types to be defined in order to have a valid XML schema. In addition, some new types must be defined to implement the representation of octet strings and bit strings. These changes are documented in Section 6. In the currently adopted revision of the standard the use of inline definitions rather than the more proper use of formal Type Definitions is also a challenge for conversion.

Section 6 also documents a number of common types that were defined globally at the start of the schema rather than being redefined identically, in-line multiple times in the schema.

Section 7 presents a sizable sample ASN.1 definition and the corresponding XML schema. The sample illustrates a large number of the conversion rules.

2. References

2.1 Applicable Publications—The following publications form a part of this specification to the extent specified herein.

2.1.1 SAE PUBLICATIONS—Available from SAE, 400 Commonwealth Drive, Warrendale, PA 15096-0001.

SAE J2353—Data Dictionary for Advanced Traveler Information Systems (ATIS)

SAE J2354—Message Sets for Advanced Traveler Information Systems (ATIS)

Readers are advised that these two documents are now being updated to reflect current data registry formats and current ITS industry practices. J2354 will also include an XML schema derived from the ASN.1 found in the revised documents meeting the translation rules of this standard.

2. This document deals only with the final set of recommended rules. An earlier set of preliminary rules, refined by group experience, is documented in several informal reports. This earlier draft is available at the committee forum site at <http://www.SAE.org>.

- 2.1.2 ITE PUBLICATIONS—Available from Institute for Transportation Engineers (ITE), 1099 14th Street, NW, Suite 300 West, Washington, DC 20005-3438 Telephone: 202-289-0222 ite_staff@ite.org Transit Communications Interface Profiles (TCIP) Framework Standard (NTCIP 1400 Draft), December 2000.

Transit Communications Interface Profiles (TCIP) Scheduling Business Area Standard Framework Standard (NTCIP 1403 Draft), December 2000.

Transit Communications Interface Profiles (TCIP) Passenger Information Business Area Standard Framework Standard (NTCIP 1403 Draft), December 2000.

Standards for Functional Level Traffic Management Data Dictionary (TMDD), Standard No. TM1.01.4.

Message Sets for External Traffic Management Center Communication (MS/ETMCC) ITE-AASHTO-TM2.01 August 2000.

- 2.1.3 PUBLICATION—XML Definitions Available on the Internet from the URLs below.

World-Wide-Web Consortium Definition of XML: <http://www.w3.org/XML/>

World-Wide-Web Consortium Definition of XML Schemas: <http://www.w3.org/XML/Schema> Consisting of Parts Two and Three. [Part One is informative and is listed in the Bibliography]

3. Definitions

- 3.1 **Application Program**—In the context of this document, any process beyond the XML validation, which may or may not be done by the top most application. For example it is common to employ intermediate levels to further process the XML source and manipulate it based on knowledge of the message structures or <xsd:appinfo> tag data.
- 3.2 **ATIS**—Advanced Traveler Information Systems, specifically the SAE J2353 and J2354 standards of ITS data elements and message sets. As a slang term, this has become the common term to refer to all the SAE efforts in developing ITS messages.
- 3.3 **ERM**—Event Reporting Message. A portion of the TMDD message set family which was develop first by rural states and has now been added to the most current draft of the ITE committee message set work.
- 3.4 **False**—In the context of this standard and XML, the Boolean value 0 is also defined as False.
- 3.5 **IEEE**—The Institute of Electrical and Electronics Engineers, a New York not-for-profit corporation engaged in developing standards. One of the standards developing organizations. A party to the cooperative development agreement of the FHWA to develop ITS standards.
- 3.6 **IM**—Incident Management, a slang term for the IEEE 1512-2000 and related family of standards that deal with exchanges of message sets between centers involved in an incident.
- 3.7 **ITE**—The Institute of Traffic Engineers, a not-for-profit corporation engaged in developing standards. One of the standards developing organizations. A party to the cooperative development agreement of the FHWA to develop ITS standards.
- 3.8 **LRMS**—Location Referencing Message Set, a portion of ITS message structures developed under the FHWA cooperative agreements and now undergoing standardization by the SAE in its second major revision. The LRMS handles all of the spatial portions of ITS messages and is therefore widely re-used by other standards.
- 3.9 **TCIP**—Transit Communications Interface Profiles

- 3.10 TMDD**—Traffic Management Data Dictionary. The data dictionary developed by the ITE as part of its work on messages used between and within traffic management centers. Also often used as a slang term for the both the data elements and the messages developed in that effort. The SAE ATIS message set reuses these data elements as well.
- 3.11 True**—In the context of this standard and XML, the Boolean value 1 is also defined as True.
- 3.12 XML**—eXtensible Marked Language
- 3.13 XSL**—eXtensible Stylesheet Language
- 4. Rules for Converting an ASN.1 Message Set Definition to an XML Schema**—The following is a list of rules organized as pseudo-code instructions for converting an ASN.1 definition to an XML schema. It is far from complete for anything that may appear in ASN.1, but it is believed complete for everything that appeared in the ATIS message sets standard as well as several other standards found in ITS.

There are many ways such a conversion could be performed; this approach attempts to be as direct as possible. For example, almost no user-defined attributes are used, and all data structures are preserved. The information contained in the transmitted document is identical, and the data structures are clearly equivalent.

One goal is that a program with knowledge of an ASN.1 definition set and the XML schema produced from it by these rules should be able to convert a conforming XML document into a conforming ASN.1 encoding and vice versa.³ A second goal, however, is that developers of XML-based interfaces and the software they develop will not require knowledge or use of ASN.1.

Note: In ASN.1, formally defined type definitions begin with a capital letter and identifiers for those types begin with a lower case letter. This conversion to XML preserves this convention, although there is no rule in XML requiring this convention. In fact, ebXML guidelines recommend that the names of all types and elements should begin with a capital letter, while the names of attributes should begin with a lower case letter. The SAE ATIS committee should consider whether to retain the lower case starting letter convention. In any event, shortening of some ASN.1 names when used as tags may be adopted.

The examples are taken from fragments of SAE ATIS, ITE TMDD, and IEEE IM standards, but the precise structures have been modified at times to serve as an example more clearly. Explanatory comments from the ASN.1 have been removed to conserve space. Consult the governing standard for correct structures and use commentary. The indentation shown in this document is purely for readability; no recommendation in this regard is given and any style desired may be adopted. Note that the convention is maintained that all defined type names start with an upper-case letter and all defined element names start with a lower-case letter.

In general, XML schemas define types and elements. Elements are instances of types. Basic types are predefined, such as integer, decimal, date, or string. User-defined types are defined as restrictions, expansions, or unions of other types. A simple type does not contain any internal structure. A complex type is a structure containing other elements or structures, defined as a SEQUENCE or SET. It is expected that the order of elements found in a complex type structure will be preserved (e.g., arbitrary re-ordering shall not be allowed).

Type names do not appear in XML documents; only the names of elements appear as tags in an XML document. Type definitions and global element definitions may be imported and exported among schemas using namespaces, but non-global element definitions may not.

3. There is some debate on the complete ability to convert back from XML to the original ASN.1. This is a laudable goal and any differences would be expected to be minor. There is a small number of XML expressions which have no equivalent in ASN.1 and hte definitive translations of these would be problematic to ASN.1 users.

Sections 4.1 through 4.7 present rules for converting ASN.1 simple-type definitions to the corresponding XML simple type definitions. Sections 4.8 through 4.12 present rules for converting ASN.1 complex type definitions to the corresponding XML complex type definitions. Sections 4.13 through 4.18 present rules for converting ASN.1 element definitions to the corresponding XML element definitions. Sections 4.19 through 4.25 present other rules to follow in creating an XML schema from an ASN.1 definition.

- 4.1 Defining a Simple Type in Terms of a Basic Type**—To define a new simple type in terms of an existing basic type, replace the ASN.1 definition *Abc ::= basictype*

```
<xsd:simpleType name="Abc">
  <xsd:restriction base="xsd:basictype"/>
</xsd:simpleType>
```

where basic type is chosen from the following table:

TABLE 1—XML TRANSLATION BASIC TYPES

ASN.1 basic type	XML basic type
Any date	xsd:date
Any date + time	xsd:dateTime
Any time	xsd:time
BOOLEAN	xsd:boolean
IA5STRING	xsd:string
INTEGER	xsd:integer -- See Section 4.2 for further rules
NumericString	xsd:string
PrintableString	xsd:string
REAL	xsd:decimal or xsd:float
UTF8String	xsd:string
VisibleString	xsd:string

The following types are not considered basic XML types and are handled in the sections that follow.

TABLE 2—XML TRANSLATION RECOMMENDATIONS

ASN.1 basic type	XML Recommendations
ASN.1 basic type	XML Recommendations
ASN.1 basic type	XML Recommendations
ASN.1 basic type	XML Recommendations
ASN.1 basic type	XML Recommendations
ASN.1 basic type	XML Recommendations

- 4.2 Defining Basic INTEGER Types**—For definitions of integers, derived XML classes may be used if the defined values fall within the minimum and maximum (inclusive) values specified in the table below. XML basic integer types are defined in the following table.

TABLE 3—RANGES FOR XML INTEGER TYPES

Minimum value	Maximum value	XML type
–2147483648	2147483647	xsd:int
–32768	32767	xsd:short
–128	127	xsd:byte
0	4294967295	xsd:unsignedInt
0	65535	xsd:unsignedShort
0	255	xsd:unsignedByte

When no further information on the value range is available, the type `xsd:integer` shall be used.

For example, the ASN.1 type definitions:

```
DatabaseNumber ::= INTEGER (0..255)
```

The resulting schema fragment would be

```
<xsd:simpleType name="DatabaseNumber">  
  <xsd:restriction base="xsd:unsignedByte"/>  
</xsd:simpleType>
```

4.3 Defining INTEGER Types with Minimum and Maximum Values—For an ASN.1 type defined as an integer with a minimum and maximum value, replace `Abc ::= INTEGER (a .. b)` by:

```
<xsd:simpleType name="Abc">  
  <xsd:restriction base="xsd:integer">  
    <xsd:minInclusive value="a"/>  
    <xsd:maxInclusive value="b"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

`xsd:integer` may be replaced by one of the derived integer types listed in Table. If the lower bound is zero and the basic type is an unsigned type, the lower bound statement may be omitted since zero is already the lower bound of the unsigned type.

For example, given the ASN.1 fragments of:

```
Scaling      ::= INTEGER (0..15)  
Start-time   ::= INTEGER (0..4095)  
Location-long ::= INTEGER (–1800000000..1800000000)
```

The resulting schema fragments would be (applying this rule and the prior one for integer ranges):

```
<xsd:simpleType name="Scaling">
  <xsd:restriction base="xsd:unsignedByte">
    <xsd:maxInclusive value="15"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="Start-time">
  <xsd:restriction base="xsd:unsignedShort">
    <xsd:maxInclusive value="4095"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="Location-long">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value="-1800000000"/>
    <xsd:maxInclusive value="1800000000"/>
  </xsd:restriction>
</xsd:simpleType>
```

4.4 Defining String Types of Minimum and Maximum Length—For a type defined as a character string (typically an IA5String) of minimum and maximum length, replace

```
Abc ::= IA5String      (SIZE(a .. b)) or
Abc ::= NumericString (SIZE(a .. b)) or
Abc ::= PrintableString(SIZE(a .. b)) or
Abc ::= VisibleString (SIZE(a .. b)) by:
```

```
<xsd:simpleType name="Abc">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="a"/>
    <xsd:maxLength value="b"/>
  </xsd:restriction>
</xsd:simpleType>
```

For example, given the ASN.1 fragment of:

```
Traveler-LastName    ::= IA5String (SIZE(1..25))
Traveler-Email       ::= IA5String (SIZE(1..40))
Route-Description     ::= IA5String (SIZE(1..300))
```

The resulting schema fragment would be:

```
<xsd:simpleType name="Traveler-LastName">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="25"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="Traveler-Email">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="40"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="Route-Description">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="300"/>
  </xsd:restriction>
</xsd:simpleType>
```

4.5 Defining String Types of Fixed Length—For a type defined as a string (typically an IA5String or other derived ASN.1 string type) of fixed length, replace `Abc ::= IA5String (SIZE(a))` by:

```
<xsd:simpleType name="Abc">
  <xsd:restriction base="xsd:string">
    <xsd:length value="a"/>
  </xsd:restriction>
</xsd:simpleType>
```

For example, given the ASN.1 fragment of:

```
Flight-DestinationAirport ::= IA5String (SIZE(3))
ATIS-TimeOffset           ::= IA5String (SIZE(4))
```

The resulting schema fragment would be:

```
<xsd:simpleType name="Flight-DestinationAirport">
  <xsd:restriction base="xsd:string">
    <xsd:length value="3"/>
  </xsd:restriction>

</xsd:simpleType>
<xsd:simpleType name="ATIS-TimeOffset ">
  <xsd:restriction base="xsd:string">
    <xsd:length value="4"/>
  </xsd:restriction>
</xsd:simpleType>
```

4.6 Defining ENUMERATED Types—For a type defined as an ENUMERATED type, replace

Abc ::= ENUMERATED { . . . } by:

```
<xsd:simpleType name="Abc">
  <xsd:union>
    <xsd:simpleType>
      <xsd:restriction base="xsd:unsignedInt">
        . . .
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value=". . ."/>
        <xsd:enumeration value=". . ."/>
      </xsd:restriction>
    </xsd:simpleType >
  </xsd:union>
  <xsd:annotation>
    <xsd:appinfo>
      . . .
    </xsd:appinfo>
  </xsd:annotation>
</xsd:simpleType>
```

The base type of the integer index is set as per prior rules for the integer range of the index list. The possible values of the integer index are either enumerated (if there are only a few permissible values) or specified with minInclusive and maxInclusive statements. The permissible text values are enumerated. The union of the integer index with the enumerated list of text strings permits an element of the defined type to contain either an index or one of the permissible text strings. The optional <appinfo> section inside <annotation>tags permits index values to be linked with text values by an application program. Section 5.1 discusses this construction in greater detail.

If the original intention of the enumeration was to be as a bit string, then the numerical indexes should be omitted and only the textual enumerations shall be used.⁴

4. The *Free-Flow-Rate* data element adopted in the original version of SAE J2369 would be an example of this type of definition of an enumeration with binary overtones. List items include lines formatted as: under-80-mph (13) , -- '1101 ' B.

For example, given the ASN.1 definition:

```

ATIS-SearchOperator ::= ENUMERATED {
    mustContain      (0),
    shouldContain    (1),
    shouldNotContain (2),
    mustNotContain   (3),
    ...

```

The resulting schema would be:

```

<xsd:simpleType name="ATIS-SearchOperator">
  <xsd:union>
    <xsd:simpleType>
      <xsd:restriction base="xsd:unsignedByte">
        <xsd:maxInclusive value="3"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="mustContain"/>
        <xsd:enumeration value="shouldContain"/>
        <xsd:enumeration value="shouldNotContain"/>
        <xsd:enumeration value="mustNotContain"/>
      </xsd:restriction>
    </xsd:simpleType >
  </xsd:union>
  <xsd:annotation>
    <xsd:appinfo>
      mustContain (0)
      shouldContain (1)
      shouldNotContain (2)
      mustNotContain (3)
    </xsd:appinfo>
  </xsd:annotation>
</xsd:simpleType>

```

If `atis-SearchOperator` is defined to be an element of type `ATIS-SearchOperator`, then an example of content using this definition would be

```
<atis-SearchOperator>mustContain</atis-SearchOperator>
```

An equally valid expression with the same meaning would be

```
<atis-SearchOperator>0</atis-SearchOperator>
```

If one or more of the index values in a range is undefined, then the missing elements shall be added to create a continuous range. If large gaps in range are present, a union for each smaller continuous range may be used.

If one or more of the integer index values is defined as a “local value,” then the following construct should be added to the union:

```
<xsd:simpleType>
  <xsd:restriction base="xsd:string"/>
    <xsd:pattern value = "\[.*\].*" />
  </xsd:restriction>
</xsd:simpleType>
```

This permits a value such as [StateDOT]SpecialCode to appear as the value of the element. Such a definition (1) tells anyone receiving the message whose special code it is (StateDOT's), (2) enables StateDOT to write code to look for its SpecialCode value and handle it accordingly, (3) enables other parties who don't have StateDOT's list of special codes to validate the document without needing to know what the special code means.

Currently there is no rule for what to do with index values labeled as “reserved for future use.” If new integer values and corresponding text values are added to the standard, the standard will be changed by expanding the range of index values permitted and adding the new text value to the enumerated list of permissible text values.

- 4.7 Defining a Type Definition Embedded in an Element Definition**—Sometimes an ASN.1 definition contains an embedded type definition. In these cases, the ASN.1 should be restructured so that the type is defined globally rather than within the scope of the data element, such as in the case below.

```
Direction ::= BIT STRING (SIZE(1))
  one-Direction    Direction ::= '0'B
  both-Directions  Direction ::= '1'B
```

In such a case the ASN.1 may be re-defined for the purpose of converting to XML as an in-line definition of an enumerated type, such as:

```
Direction ::= BIT STRING (SIZE(1)) {
  one-Direction    (0),
  both-Directions  (1)
}
```

Once converted, the resulting enumeration is translated using the other rules in this document; refer to 4.6.

- 4.8 Defining a Complex Type as a SEQUENCE of Elements**—Replace a complex type definition
 Abc ::= SEQUENCE { ... } or Abc ::= SET { ... } by:

```
<xsd:complexType name="Abc">
  <xsd:sequence>
    ...
  </xsd:sequence>
</xsd:complexType>
```

For example, the ASN.1 fragment

```
DirectoryRequest ::= SEQUENCE {  
    coreRequest DirectoryCoreRequest,  
    start DateTimePair OPTIONAL,  
    end DateTimePair OPTIONAL  
}
```

is converted to

```
<xsd:complexType name="DirectoryRequest">  
  <xsd:sequence>  
    <xsd:element name="coreRequest" type="DirectoryCoreRequest"/>  
    <xsd:element name="start" type="xsd:dateTime" minOccurs="0"/>  
    <xsd:element name="end" type="xsd:dateTime" minOccurs="0"/>  
  </xsd:sequence>  
  
</xsd:complexType>
```

Refer to Section 4.20 for rules about optional elements.

4.9 Defining a Complex Type as a CHOICE of Elements—Replace a complex type definition Abc ::= CHOICE { ... } by:

```
<xsd:complexType name="Abc">  
  <xsd:choice>  
    ...  
  </xsd:choice>  
</xsd:complexType>
```

For example, the ASN.1 fragment

```
LinkOrNodeAdditionalInformation ::= CHOICE {  
    link LinkAdditionalInformation,  
    node NodeAdditionalInformation  
}
```

is converted to:

```
<xsd:complexType name="LinkOrNodeAdditionalInformation">  
  <xsd:choice>  
    <xsd:element name="link" type="LinkAdditionalInformation"/>  
    <xsd:element name="node" type="NodeAdditionalInformation"/>  
  </xsd:choice>  
</xsd:complexType>
```

4.10 Defining a Complex Type as a SEQUENCE OF Named Types—A type defined as a sequence of multiple occurrences of a named type translates to a sequence of multiple occurrences of elements of that named type. For these definitions, replace: `Abc ::= SEQUENCE OF Def` or `Abc [number] SEQUENCE OF Def` by:

```
<xsd:complexType name="Abc">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="def" type="Def"/>
  </xsd:sequence>
</xsd:complexType>
```

For example, given the ASN.1 definition of:

```
TrafficInformation ::= SEQUENCE OF LinkOrNode
```

The resulting schema definition would be:

```
<xsd:complexType name="TrafficInformation">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="linkOrNode" type="LinkOrNode"/>
  </xsd:sequence>
</xsd:complexType>
```

Note that the type name (beginning with a capital letter) has been used with a lower case initial to name an element of that type. This new element in the sequence is not required by ASN.1 but is required in XML.

An example of content using this would be

```
<TrafficInformation>
  <linkOrNode> complex contents of type LinkOrNode here </linkOrNode>
  <linkOrNode> complex contents of type LinkOrNode here </linkOrNode>
  <linkOrNode> complex contents of type LinkOrNode here </linkOrNode>
</TrafficInformation>
```

4.11 Defining a Complex Type as a SEQUENCE OF a Named Type with Minimum and Maximum Number of Elements—For a type defined as a sequence of a named type with a minimum and/or maximum number of elements, replace:

`Abc ::= SEQUENCE (a .. b) of Def` or `Abc [number] SEQUENCE (a .. b) of Def`

```
<xsd:complexType name="Abc">
  <xsd:sequence minOccurs="a" maxOccurs="b">
    <xsd:element name="def" type="Def"/>
  </xsd:sequence>
</xsd:complexType>
```