Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.

- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL’s, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.

- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft [Open Specification Promise](#) or the [Community Promise](#). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).

- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights.** All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.
## Revision Summary

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/06/2009</td>
<td>0.1</td>
<td>Major</td>
<td>Initial Availability</td>
</tr>
<tr>
<td>02/19/2010</td>
<td>1.0</td>
<td>Minor</td>
<td>Updated the technical content</td>
</tr>
<tr>
<td>03/31/2010</td>
<td>1.01</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>04/30/2010</td>
<td>1.02</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>06/07/2010</td>
<td>1.03</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>06/29/2010</td>
<td>1.04</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content.</td>
</tr>
<tr>
<td>07/23/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>09/27/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>11/15/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>12/17/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>03/18/2011</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>06/10/2011</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>01/20/2012</td>
<td>1.5</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>04/11/2012</td>
<td>1.5</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>07/16/2012</td>
<td>1.5</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
</tbody>
</table>
# Table of Contents

1 **Introduction** ........................................................................................................................................ 5  
   1.1 Glossary ........................................................................................................................................... 5  
   1.2 References ....................................................................................................................................... 5  
      1.2.1 Normative References ............................................................................................................ 5  
      1.2.2 Informative References .......................................................................................................... 5  
   1.3 Structure Overview (Synopsis) ........................................................................................................ 6  
   1.4 Relationship to Protocols and Other Structures ............................................................................. 6  
   1.5 Applicability Statement .................................................................................................................. 6  
   1.6 Versioning and Localization ........................................................................................................... 7  
   1.7 Vendor-Extensible Fields ................................................................................................................. 7  

2 **Structures** ............................................................................................................................................. 8  
   2.1 XML Automaton Files ....................................................................................................................... 8  
      2.1.1 Part Enumerations .................................................................................................................... 8  
      2.1.2 Extensions .................................................................................................................................. 8  
      2.1.3 Global Elements ....................................................................................................................... 8  
         2.1.3.1 fsa Element ......................................................................................................................... 8  
      2.1.4 Global Attributes ..................................................................................................................... 8  
      2.1.5 Complex Types ......................................................................................................................... 8  
         2.1.5.1 CT_states ............................................................................................................................ 8  
            2.1.5.1.1 CT_state ......................................................................................................................... 9  
            2.1.5.1.1.1 CT_transition ........................................................................................................... 9  
         2.1.5.2 CT_values ............................................................................................................................ 10  
            2.1.5.2.1 CT_value ....................................................................................................................... 10  
            2.1.5.3 CT_meta ........................................................................................................................... 10  
      2.1.6 Simple Types ............................................................................................................................. 11  
   2.2 XML Dictionary Files ....................................................................................................................... 11  
      2.2.1 Part Enumerations .................................................................................................................... 11  
      2.2.2 Extensions .................................................................................................................................. 11  
      2.2.3 Global Elements ....................................................................................................................... 11  
         2.2.3.1 dictionary Element ............................................................................................................. 11  
      2.2.4 Global Attributes ..................................................................................................................... 11  
      2.2.5 Complex Types ......................................................................................................................... 11  
         2.2.5.1 CT_entry ............................................................................................................................. 11  
      2.2.6 Simple Types ............................................................................................................................ 12  

3 **Structure Examples** ............................................................................................................................. 13  
   3.1 XML Dictionary File ......................................................................................................................... 13  
   3.2 XML Automaton File ....................................................................................................................... 13  
   3.3 XML Empty Automaton File ........................................................................................................... 14  

4 **Security Considerations** ..................................................................................................................... 15  

5 **Appendix A: XML Schemas** ............................................................................................................. 16  
   5.1 Automata Files XML Schema ......................................................................................................... 16  
   5.2 Dictionary Files XML Schema ......................................................................................................... 17  

6 **Appendix B: Product Behavior** ......................................................................................................... 18  

7 **Change Tracking** ............................................................................................................................... 19  

---

[MS-FSLRDS] — v20120630  
Linguistic Resource Data Structure  
Copyright © 2012 Microsoft Corporation.  
Release: July 16, 2012
1 Introduction

The Linguistic Resource Data Structure provides file formats that enable linguistic dictionaries to be represented as XML files. A typical scenario for using these structures is to store linguistic data that is employed by different applications.

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

XML

The following terms are defined in [MS-OFCGLOS]:

automaton
dictionary
property extraction

The following terms are specific to this document:

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.


1.2.2 Informative References


[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
1.3 Structure Overview (Synopsis)

This specification documents two file formats: An automaton file format, and a plain text dictionary file format. Both file formats use XML files to represent linguistic information.

The automaton file format uses XML files to represent a deterministic finite state machine that contains the following:

- A finite set of states.
- A finite set of transitions that are annotated with transition labels.
- A finite set of output values.

The dictionary file format uses XML files to contain keys, and optional values for these keys, in plain text format.

A finite state machine (FSM) is a mechanism that reads an input string and optionally writes an output string. Starting at a specially designated state called the start state, the FSM reads one character of its input string and, if there is a transition to another state that is labeled with that character, the FSM performs this transition. There is a special set of states called the final states, which are states without transitions to other states.

A deterministic finite state machine (DFSM) is an FSM that, for every state and every input character, has only one transition to the state that is labeled with that character. The automata that these structures represent are always deterministic.

For a more comprehensive description of deterministic finite state machines and the related algorithms, see [HopcroftUllman].

Deterministic finite state machines can be employed in the encoding of linguistic data, for example, into dictionaries that are used for spell checking or property extraction.

1.4 Relationship to Protocols and Other Structures

The structures that are described in this document can be used as linguistic dictionaries that are uploaded to and downloaded from the Resource Store, as described in [MS-FSRS].

The XML files that are described in this document can optionally be compressed with the "gzip" tool, as described in [RFC1952]. In that case, the files require the extension ".xml.gz"; otherwise, they require the ".xml" extension.

1.5 Applicability Statement

The structures described in this document are used to store linguistic data that is shared between applications. Typical examples are applications that create and use spell checking data, or applications that create and use property extraction data.

The dictionary files, which are ideal for smaller amounts of data, can be edited manually. The automaton files are not usually edited manually; instead, they are automatically created or updated from other dictionary sources by using the algorithms that are described in [HopcroftUllman].
1.6  Versioning and Localization

None.

1.7  Vendor-Extensible Fields

None.
2 Structures

2.1 XML Automaton Files

The terminology (such as "state", "transition") in this section is used in the same technical sense as in section 1.3.

2.1.1 Part Enumerations

None.

2.1.2 Extensions

None.

2.1.3 Global Elements

2.1.3.1 fsa Element

This element MUST be the root element for the file. In this sequence, the fsa element contains one states element, one values element, and one meta element, as follows:

```xml
<xs:element name="fsa">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="states" type="CT_states" />
      <xs:element minOccurs="1" maxOccurs="1" name="values" type="CT_values" />
      <xs:element minOccurs="1" maxOccurs="1" name="meta" type="CT_meta" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

states: A CT_states element.

values: A CT_values element.

meta: A CT_meta element.

2.1.4 Global Attributes

None.

2.1.5 Complex Types

2.1.5.1 CT_states

This complex type is referenced by the fsa element, has no attributes, and contains a list of states, as follows.

```xml
<xs:complexType name="CT_states">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="unbounded" name="s" type="CT_state" />
  </xs:sequence>
</xs:complexType>
```
2.1.5.1.1   CT_state

This complex type referenced by the CT_states element represents a state of the automaton. The CT_state element can optionally contain transitions to other states represented by a sequence of t elements. If the CT_state element contains such transitions, the state is called a non-final state, otherwise the state is called a final state, as follows.

```xml
<xs:complexType name="CT_state">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="t" type="CT_transition" />
  </xs:sequence>
  <xs:attribute name="i" type="xs:unsignedInt" use="required" />
  <xs:attribute name="v" type="xs:unsignedInt" use="optional" />
</xs:complexType>
```

The attributes of the CT_state element are represented in the following table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>The identifier of this state, an unsigned integer.</td>
</tr>
<tr>
<td>v</td>
<td>The value of this state, an unsigned integer. This is a reference to a v element in the same automaton, that is, the integer MUST correspond to the value of an i attribute of a v element in the same automaton.</td>
</tr>
</tbody>
</table>

2.1.5.1.1.1   CT_transition

This complex type is referenced by the CT_state element, has no child elements, and represents a transition of the automaton from the state that contains this element, to the target state that is denoted by the attribute t. This process reads a single character, that is, the label of the transition, the attribute l, as follows.

```xml
<xs:complexType name="CT_transition">
  <xs:attribute name="l" type="xs:unsignedByte" use="required"/>
  <xs:attribute name="t" type="xs:unsignedInt" use="required"/>
</xs:complexType>
```

The attributes of the CT_transition element are represented in the following table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>The label of the transition, consisting of an unsigned integer which represents the byte value of a single character. For example the character &quot;e&quot; is represented by its ASCII value &quot;101&quot;. It is the responsibility of the application reading the automaton to interpret this integer value as a character in a given encoding, which can be different from the encoding of the XML file.</td>
</tr>
</tbody>
</table>
| t    | The target state of the transition, an unsigned integer. This is a reference to an s element in the
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>The identifier of this value, an unsigned integer.</td>
</tr>
<tr>
<td>v</td>
<td>The output value, a string in the encoding denoted by the &quot;encoding&quot; attribute of the file.</td>
</tr>
</tbody>
</table>

2.1.5.2 CT_values

This complex type is referenced by the fsa element, has no attributes, and contains a list of output values represented by a non-empty sequence of v elements, as follows.

```xml
<xs:complexType name="CT_values">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="unbounded" name="v" type="CT_value" />
  </xs:sequence>
</xs:complexType>
```

v: A CT_value element.

2.1.5.2.1 CT_value

This complex type is referenced by the CT_values element, has no child elements, and represents the output value of a state of the automaton, which is referenced by the v attribute of an s element, as follows.

```xml
<xs:complexType name="CT_value">
  <xs:attribute name="i" type="xs:unsignedInt" use="required" />
  <xs:attribute name="v" type="xs:string" use="required" />
</xs:complexType>
```

The attributes of the CT_value element are represented in the following table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>The identifier of this value, an unsigned integer.</td>
</tr>
<tr>
<td>v</td>
<td>The output value, a string in the encoding denoted by the &quot;encoding&quot; attribute of the file.</td>
</tr>
</tbody>
</table>

2.1.5.3 CT_meta

This complex type is referenced by the fsa element, has no child elements, and contains information about the start state of the automaton, as follows.

```xml
<xs:complexType name="CT_meta">
  <xs:attribute name="startstate" type="xs:unsignedInt" use="required" />
</xs:complexType>
```

The attributes of the CT_meta element are represented in the following table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startstate</td>
<td>This attribute represents the start state of the automaton as an unsigned integer identifier.</td>
</tr>
</tbody>
</table>
2.1.6 Simple Types
None.

2.2 XML Dictionary Files

2.2.1 Part Enumerations
None.

2.2.2 Extensions
None.

2.2.3 Global Elements

2.2.3.1 dictionary Element
This element, which has no attributes, MUST be the root element for the file. This element optionally contains a sequence of entry elements as follows.

```xml
<xs:element name="dictionary">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="entry" type="CT_entry" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

entry: A CT_entry element.

2.2.4 Global Attributes
None.

2.2.5 Complex Types

2.2.5.1 CT_entry
This complex type is referenced by the dictionary element, has no child elements, and represents an entry in the dictionary containing a key and, optionally, a value, as follows.

```xml
<xs:complexType name="CT_entry">
  <xs:attribute name="key" type="xs:string" use="required" />
  <xs:attribute name="value" type="xs:string" use="optional" />
</xs:complexType>
```
The attributes of the CT_entry element are represented in the following table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>This attribute denotes the key of the dictionary entry, that is, the entry string itself. The value of the attribute is a string in the encoding of the XML file.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>value</td>
<td>This optional attribute denotes a value that is associated with a dictionary entry. The value of the attribute is a string in the encoding of the XML file.</td>
</tr>
</tbody>
</table>

### 2.2.6 Simple Types

None.


3 Structure Examples

3.1 XML Dictionary File

The following example file represents a dictionary in the key-value format. The key "test" has the integer value 25, the key "others" has the value "other", and the key "none" is not associated with a value.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
  <entry key="test" value="25"/>
  <entry key="none"/>
  <entry key="others" value="other"/>
</dictionary>
```

3.2 XML Automaton File

This section describes an automaton based on the example shown in section 3.1 in the automaton format. The following figure represents the states of the automaton with circles; arrows represent the transitions between states.

![Finite state machine represented by an automaton](image)

Figure 1: Finite state machine represented by an automaton

The start state is the state numbered with 1. The final states, that is, the states that have optional output values are represented by double circles in the figure. The output values of the final states are shown in quotation marks. The transitions are identified by the byte values of their labels; for example, "e" is represented as 101.

This finite state machine is represented by the following example of an automaton XML file. The XML information in this example represents the same information that is described the example shown in section 3.1 in the automaton format.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<fsa>
  <states>
    <s i="5" v="1"/>
    <s i="4">
      <t l="101" t="5"/>
    </t>
    <s i="3">
      <t l="110" t="4"/>
    </t>
    <s i="2">
      <t l="111" t="3"/>
    </t>
  </states>
</fsa>
```
3.3 XML Empty Automaton File

The following example file represents an empty automaton. It contains one state that is both an initial state and a final state, and which has an empty output value.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<fsa>
  <states>
    <s i="1" v="1"/>
  </states>
  <values>
    <v i="1" v=""/>
  </values>
  <meta startstate="1"/>
</fsa>
```
4 Security Considerations

None.
5 Appendix A: XML Schemas

For ease of implementation, the XML Schemas are provided in the following sections.

5.1 Automata Files XML Schema

XML automaton files adhere to the following XML schema.

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- **************** Main element structure ****************** -->
  <xs:element name="fsa">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="states" type="CT_states" />
        <xs:element minOccurs="1" maxOccurs="1" name="values" type="CT_values" />
        <xs:element minOccurs="1" maxOccurs="1" name="meta" type="CT_meta" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- **************** Complex types ****************** -->
  <xs:complexType name="CT_states">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="unbounded" name="s" type="CT_state" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="CT_values">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="unbounded" name="v" type="CT_value" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="CT_meta">
    <xs:attribute name="startstate" type="xs:unsignedInt" use="required" />
  </xs:complexType>

  <xs:complexType name="CT_state">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="t" type="CT_transition" />
    </xs:sequence>
    <xs:attribute name="i" type="xs:unsignedInt" use="required" />
    <xs:attribute name="v" type="xs:string" use="optional" />
  </xs:complexType>

  <xs:complexType name="CT_value">
    <xs:attribute name="i" type="xs:unsignedInt" use="required" />
  </xs:complexType>

  <xs:complexType name="CT_transition">
    <xs:attribute name="i" type="xs:unsignedInt" use="required" />
    <xs:attribute name="v" type="xs:string" use="required" />
  </xs:complexType>
</xs:schema>
```
5.2 Dictionary Files XML Schema

XML dictionary files adhere to the following XML schema.

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- *************** Main element structure ****************-->  
  <xs:element name="dictionary">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="entry" type="CT_entry"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- *************** Complex types ****************-->  
  <xs:complexType name="CT_entry">
    <xs:attribute name="key" type="xs:string" use="required"/>
    <xs:attribute name="value" type="xs:string" use="optional"/>
  </xs:complexType>
</xs:schema>
```
6 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® FAST™ Search Server 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.
7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.
Index

A

Applicability 6
Automaton
  XML files 8
  XML schema 16

C

Change tracking 19
Complex types (section 2.1.5 8, section 2.2.5 11)
  CT_entry 11
  CT_meta 10
  CT_state 9
  CT_states 8
  CT_transition 9
  CT_value 10
  CT_values 10
  CT_entry complex type 11
  CT_meta complex type 10
  CT_state complex type 9
  CT_states complex type 8
  CT_transition complex type 9
  CT_value complex type 10
  CT_values complex type 10

D

Details
  Automaton - XML files
    complex types 8
    extensions 8
    global attributes 8
    global elements 8
    part enumerations 8
    simple types 11
  Automaton - XML files 8
  CT_entry complex type 11
  CT_meta complex type 10
  CT_state complex type 9
  CT_states complex type 8
  CT_transition complex type 9
  CT_value complex type 10
  CT_values complex type 10
  Dictionary - XML files 11
    complex types 11
    extensions 11
    global attributes 11
    global elements 11
    part enumerations 11
    simple types 12
  Dictionary - XML schema 17
    dictionary element 11

E

Elements
  dictionary 11
  fsa 8

Elements - global
  dictionary element 11
  fsa element 8
Examples
  XML Automaton File 13
  XML Dictionary File 13
  XML Empty Automaton File 14
Extensions (section 2.1.2 8, section 2.2.2 11)

F

Fields - vendor-extensible 7
  fsa element 8
  Full XML schema 16

G

Global attributes (section 2.1.4 8, section 2.2.4 11)
Global elements (section 2.1.3 8, section 2.2.3 11)
  dictionary element 11
  fsa element 8
Glossary 5

I

Implementer - security considerations 15
Informative references 5
Introduction 5

L

Localization 7

N

Normative references 5

O

Overview (synopsis) 6

P

Part enumerations (section 2.1.1 8, section 2.2.1 11)
Product behavior 18

R

References 5
  informative 5
  normative 5
Relationship to protocols and other structures 6

S

Security - implementer considerations 15
Simple types (section 2.1.6 11, section 2.2.6 12)
Structures
Automaton - XML files
- complex types 8
- extensions 8
- global attributes 8
- global elements 8
- part enumerations 8
- simple types 11

Dictionary - XML files 11
- complex types 11
- extensions 11
- global attributes 11
- global elements 11
- part enumerations 11
- simple types 12

**T**

Tracking changes 19

Types - complex (section 2.1.5 8, section 2.2.5 11)
- CT_entry 11
- CT_meta 10
- CT_state 9
- CT_states 8
- CT_transition 9
- CT_value 10
- CT_values 10

Types - simple (section 2.1.6 11, section 2.2.6 12)

**V**

Vendor-extensible fields 7

Versioning 7

**X**

XML
- dictionary files 11
- XML Automaton File example 13
- XML Dictionary File example 13
- XML Empty Automaton File example 14

XML files
- Automaton 8
- Dictionary 11
- XML schema 16
- automaton 16
- dictionary 17