Emergency Data Exchange Language (EDXL) Common Types (CT) Version 1.0

Working Draft 03

03 May 2011

Abstract:

This Common Types describes components and component types that can be reused across the suite of Emergency Data Exchange Language (EDXL) standards. These common components and types are intended for internal use by the Emergency Management Technical Committee and its subcommittees as they develop specific standards utilizing these types.

Status:

This Working Draft (WD) is a preliminary version of a Work Product produced by one or more TC Members that has not yet been voted on by the TC to be approved as a Committee Specification Draft (CSD). The OASIS document Approval Process begins officially with a TC vote to approve a WD as a CSD, which is the target state for this Work Product. The TC may at any stage during development of a Work Product approve a Working Draft as a CSD; approval of a draft at CSD level requires a Full Majority Vote of the TC. The TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Specification Draft.

Table of Contents

1 Introduction

[All text is normative unless otherwise labeled]

This document describes common components and component types that can be reused across the suite of Emergency Data Exchange Language (EDXL) standards. This document is intended for internal use by the Emergency Management Technical Committee and its subcommittees as they develop specific standards utilizing these types. The goal is to enable reuse of components which are commonly used in specifications and which have been designed based on lessons learned from the development of the Common Alert Protocol 1.1, the Distribution Element 1.0, Hospital Availability and Resource Messaging. The first use of these common components is intended to be in Situation Reports 1.0 and the Distribution Element 2.0. The components will be used and expanded as needed for future EDXL specifications.

1.1 Terminology

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 **[RFC2119]**.

1.2 Normative References

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[RFC2046] N. Freed, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, http://www.ietf.org/rfc/rfc2046.txt, IETF RFC 2046, November 1996.

[RFC3066] H. Alvestrand, Tags for the Identification of Languages, http://www.ietf.org/rfc/rfc3066.txt, IETF RFC 3066, January 2001.

[WGS 84] National Geospatial Intelligence Agency, Department of Defense World Geodetic System 1984, http://earth-info.nga.mil/GandG/tr8350_2.html, NGA Technical Report TR8350.2, January 2000.

[XML 1.0] T. Bray, *Extensible Markup Language (XML) 1.0 (Third Edition)*, http://www.w3.org/TR/REC-xml/, W3C REC-XML-20040204, February 2004.

[namespaces] T. Bray, *Namespaces in XML*, http://www.w3.org/TR/REC-xml-names/, W3C REC-xml-names-19990114, January 1999.

[dateTime] N. Freed, *XML Schema Part 2: Datatypes Second Edition*, http://www.w3.org/TR/xmlschema-2/#dateTime, W3C REC-xmlschema-2, October 2004.

1.3 Non-Normative References

[EDXL GFR] *EDXL General Functional Requirements*, http://www.oasis-open.org/committees/document.php?document_id=10031&wg_abbrev=emergency, November 2004.

[EDXL-DE IG] *EDXL Distribution Element Implementer's Guide*, http://www.oasis-open.org/committees/document.php?document_id=14120&wg_abbrev=emergency, August 2005.

2 Design Principles & Concepts (non-normative)

2.1 Design Philosophy

Below are some of the guiding principles of the EDXL CIQ Profile:

1. Provide a method to capture and reuse xml types and elements for representing persons and organizations which are commonly needed across multiple EDXL standards.

2. Provide flexible mechanisms to update the EDXL CIQ Profile efficiently, without slowing down the EDXL standards development process.

3. Allow for easy updates to capture fixes or improvements.

4. Ease the reuse and understanding of the basic CIQ person and organization elements and types important for emergency management.

5. Speed the development of EDXL Standards through reuse of common components and thereby improve information sharing and data exchange across the local, state, tribal, national and non-governmental organizations of different professions that provide emergency response and management services.

6. Support the integration of data elements from profiles which enables efficient and effective reuse of other important open standards.

2.2 Structural Summary

About multiplicity notation: “[l..h]” designates range from lower bound l to higher bound h, with both l and h Natural numbers, 0 ≤ l ≤ h, plus option “*” (unbounded) for h.

2.2.1 Simple Types

	Type name	depends on
–	EDXLDateTimeType	xs:dateTime
–	EDXLStringType	xs:token
–	PercentageType	xs:float
–	ValueListURIType	xs:anyURI
–	ValueType	xs:string
–	CurrencyType	xs:string
–	DegreesCType	xs:float [-100 .. 70]
–	DegreesCircleType	xs:float [0 .. 360]

- WeatherQualifierType xs:string [*enumeration*]
- WeatherDescriptorType xs:string [*enumeration*]
- WeatherPrecipitationType xs:string [*enumeration*]
- WeatherObscurationType xs:string [*enumeration*]
- WeatherAddlPhenomType xs:string [*enumeration*]
- SkyConditionType xs:string [*enumeration*]

2.2.2 Complex Types

Type name		depends on
– ValueListType		ct:ValueListURI, ct:Value
– ValueKeyType		ct:ValueListURI, ct:Value
– ValueKeyStringPairType		ct:ValueKeyType, xs:string
– ValueKeyIntPairType		ct:ValueKeyType, xs:int
– TimePeriodType		ct:EDXLDateTimeType
– PersonTimePairType		ct:PersonDetailsType, ct:EDXLDateTimeType
– OrganizationInformationType		xpil:OrganizationDetailsType
– PersonDetailsType		xpil:PersonDetailsType
– METARType		*sequence of elements*
– StationID	[1,1]	xs:string *restricted*
– ObservationTime	[1,1]	ct:EDXLDateTimeType
– TempC	[0,1]	ct:DegreesCType
– DewPointC	[0,1]	ct:DegreesCType
– WindDirDegrees	[0,1]	ct:DegreesCircleType
– WindSpeedkt	[0,1]	xs:int [0 .. 300]
– WindGustkt	[0,1]	xs:int [0 .. 300]
– VisibilityStatuteMI	[0,1]	xs:float [0.0 .. 10.0]
– AltimeterHP	[0,1]	xs:int [800 .. 1200]
– SeaLevelPressuremb	[0,1]	xs:int [800 .. 1200]
– WeatherPhenomenaReport	[0,1]	*sequence of elements*
– Qualifier	[0,1]	ct:WeatherQualifierType
– Descriptor	[0,1]	ct:WeatherDescriptorType
– Precipitation	[0,1]	ct:WeatherPrecipitationType

- Obscuration [0,1] ct:WeatherObscurationType
- Additional [0,1] ct:WeatherAddlPhenomType
- SkyCondition [0,1] ct:SkyConditionType
- Precip1HrIn [0,1] xs:float *restricted*
- Precip3HrIn [0,1] xs:float *restricted*
- Precip6HrIn [0,1] xs:float *restricted*
- Precip24HrIn [0,1] xs:float *restricted*

- WeatherInfoType *sequence of elements*
  - METARString [0,1] xs:string
  - METARReadings [0,1] ct:METARType
  - WeatherRemarks [0,1] xs:string
  - WeatherConcerns [0,1] xs:string

2.2.3 Top Level Elements

Element name	depends on
ValueListURI	ct:ValueListURIType
Value	ct:ValueType
WeatherInfo	ct:WeatherInfoType

3 EDXL Common Types Structure (normative)

3.1 Data Dictionary

Namespaces and prefixes used below include:

xs="http://www.w3.org/2001/XMLSchema"
ct="urn:oasis:names:tc:emergency:edxl:ct:1.0"
xpil="urn:oasis:names:tc:emergency:edxl:ciq:1.0:xpil"
xal="urn:oasis:names:tc:emergency:edxl:ciq:1.0:xal"
xnl="urn:oasis:names:tc:emergency:edxl:ciq:1.0:xnl"

3.1.1 EDXL Common Simple Types

Type	**EDXLDateTimeType**
BaseType	Restricted xs:dateTime
Restriction	Pattern "\d\d\d\d-\d\d-\d\dT\d\d:\d\d:\d\d[-,+]\d\d:\d\d"
Usage	Use wherever you would otherwise use xs:dateTime
Definition	A restricted form of dateTime which requires the use of a timezone offset and thereby prohibits the use of "Z" without an offset. Also prohibited is the use fractional seconds.
Comments	1. The uniform requirement for a timezone offset provides greater reliability and robustness for emergency systems.
Schema Component	<xs:simpleType name="EDXLDateTimeType"> <xs:restriction base="xs:dateTime"> <xs:pattern value="\d\d\d\d-\d\d-\d\dT\d\d:\d\d:\d\d[-,+]\d\d:\d\d"/> </xs:restriction> </xs:simpleType>
Used In	Top level type
Example	<DateTimeSent>2009-11-15T16:53:00-05:00</DateTimeSent>

Type	**EDXLStringType**
BaseType	Restricted xs:token
Restriction	minLength = 1, maxLength = 1023
Usage	Use wherever you would otherwise use xs:string
Definition	A restricted form of string which is limited to 1023 characters (and must be at least 1 character) in

	length
Comments	1. This common type provides a string type which is of long but limited length. Emergency systems shouldn't be required to manage indefinitely long string lengths. `maxLength` counts the maximum number of characters in the string. 2. This type does not exclude the use of the more general `xs:string`, but should be applied whenever length limitation is technically indicated, e.g. - to prevent circumvention of REQUIRED usage by supplying an empty string (length 0), or - for coding or transmission efficiency.

Schema Component	`<xs:simpleType name="EDXLStringType">`
	`  <xs:restriction base="xs:token">`
	`  <xs:maxLength value="1023"/>`
	`  <xs:minLength value="1"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

Used In	Top level type
Example	`<SenderID>mary.thompson@myagency.gov</SenderID>`

Type	**`PercentageType`**
BaseType	Restricted `xs:float`
Restriction	`minInclusive=0, maxInclusive=100.0`
Usage	Use wherever you need to use a percentage
Definition	A restricted form of unsigned floating number ranging from 0.0 to 100.0 inclusive intended to represent a percentage
Comments	1. Percentages are often used in emergency messages so this Percentage type facilitates a standardized format as opposed to ad hoc percentage formats.

Schema Component	`<xs:simpleType name="PercentageType">`
	`  <xs:restriction base="xs:float">`
	`  <xs:minInclusive value="0"/>`
	`  <xs:maxInclusive value="100.0"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

Used In	Top level type
Example	`<Percentage>100<Percentage>`

Type	**`ValueListURIType`**
BaseType	Restricted `xs:anyURI`
Restriction	None.
Usage	Used to denote the URI of a `ValueList` and related types
Definition	A URI referencing an externally-managed list of values.
Comments	1. A lesson learned from early EDXL specification development was the need to support lists of

	values that may vary depending on the jurisdiction or community. The `ValueListType` and related structures are based on the concept that the "list" of values can be maintained externally and referenced in the EDXL standards. The reference is handled by structures which include a `ValueListURI` providing a unique identifier for the external "list" and then followed by a value or values from that list. The reason "list" is quoted is because the external structure may be an ontology or other structure adopted by the jurisdiction or community rather than just a simple list.

Schema Component	`<xs:simpleType name="ValueListURIType">`
	`  <xs:restriction base="xs:anyURI"/>`
	`</xs:simpleType>`

Used In	Top level type
Examples	

Type	**`ValueType`**
BaseType	Restricted `xs:string`
Restriction	None.
Usage	Used to denote the value(s) of a `ValueList` and related types
Definition	A string value from an externally-managed list of values referenced by a `ValueListURI`.
Comments	1. A lesson learned from early EDXL specification development was the need to support lists of values that may vary depending on the jurisdiction or community. The `ValueListType` and related structures are based on the concept that the "list" of values can be maintained externally and referenced in the EDXL standards. The reference is handled by structures which include a `ValueListURI` providing a unique identifier for the external "list" and then followed by a value or values from that "list". The reason "list" is quoted is because the external structure may be an ontology or other structure adopted by the jurisdiction or community rather than just a simple list.

Schema Component	`<xs:simpleType name="Value">`
	`  <xs:restriction base="xs:string"/>`
	`</xs:simpleType>`

Used In	Top level type
Examples	

Type	**`CurrencyType`**
BaseType	`xs:string`
Restriction	Pattern `"([0-9])+[.][0-9][0-9] [A-Z][A-Z][A-Z]"`
Usage	Use wherever currency is used in a specification.
Definition	A `CurrencyType` is at least one number followed by 0 or more numbers, followed by an optional fractional part, and followed by three capital letters designating the currency code (ISO 4217).
Comments	

Schema Component	<xs:simpleType name="CurrencyType"> <xs:restriction base="xs:string"> <xs:pattern value="([0-9])+[.][0-9][0-9] [A-Z][A-Z][A-Z]"/> </xs:restriction> </xs:simpleType>
Used In	Top level type
Examples	<ct:Currency>45USD</ct:Currency> <ct:Currency xsi:schemaLocation="urn:oasis:names:tc:emergency:edxl:ct:1.0: EDXL_Com- mon_Types_ver05.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema- instance"xmlns:ct="urn:oasis:names:tc:emergency:EDXL:CT:1.0">099999999999 999999.00 AAA</ct:Currency>

Type	**DegreesCType**
BaseType	Restricted xs:float
Restriction	minInclusive=-100.0, maxInclusive=70.0
Usage	Use wherever degree Celsius is used in a specification.
Definition	A restricted form of floating number ranging from -100.0 to 70.0 inclusive, intended to represent a temperature in degrees centigrade
Comments	
Schema Component	<xs:simpleType name="DegreesCType"> <xs:restriction base="xs:float"> <xs:minInclusive value="-100.0"/> <xs:maxInclusive value="70.0"/> </xs:restriction> </xs:simpleType>
Used In	METARType.TempC
Examples	<TempC>37.2</TempC>

Type	**DegreesCircleType**
BaseType	Restricted xs:float
Restriction	minInclusive= 0.0, maxInclusive=360.0
Usage	Use wherever an angle measurement in degrees is used in a specification.
Definition	A restricted form of unsigned floating number ranging from 0.0 to 360.0 inclusive, intended to represent an angle measurement in degrees.
Comments	
Schema Component	<xs:simpleType name="DegreesCircleType"> <xs:restriction base="xs:float"> <xs:minInclusive value="0.0"/> <xs:maxInclusive value="360.0"/> </xs:restriction>

	</xs:simpleType>
Used In	METARType.WindDirDegrees
Examples	<WindDirDegrees>32.3</WindDirDegrees>

3.1.2 EDXL Enumerated Types

Type	**WeatherQualifierType**
BaseType	Enumeration
Values	"Light", "Moderate", "Heavy"
Usage	
Definition	A selection of qualifiers to categorize types of weather.
Comments	
Schema Component	<xs:simpleType name="WeatherQualifierType"> <xs:restriction base="xs:string"> <xs:enumeration value="Light"/> <xs:enumeration value="Moderate"/> <xs:enumeration value="Heavy"/> </xs:restriction> </xs:simpleType>
Used In	METARType.WeatherPhenomenaReport.Qualifier
Examples	<Qualifier>Light</Qualifier>

Type	**WeatherDescriptorType**
BaseType	Enumeration
Values	"Shallow", "Blowing", "Patches", "Showers", "Partial", "Drifting", "Thunderstorm", "Freezing"
Usage	
Definition	A selection of weather characteristics.
Comments	
Schema Component	<xs:simpleType name="WeatherDescriptorType"> <xs:restriction base="xs:string"> <xs:enumeration value="Shallow"/> <xs:enumeration value="Blowing"/> <xs:enumeration value="Patches"/> <xs:enumeration value="Showers"/> <xs:enumeration value="Partial"/> <xs:enumeration value="Drifting"/> <xs:enumeration value="Thunderstorm"/> <xs:enumeration value="Freezing"/> </xs:restriction> </xs:simpleType>
Used In	METARType.WeatherPhenomenaReport.Descriptor

Examples	<Descriptor>Showers</Descriptor>

Type	**WeatherPrecipitationType**
BaseType	Enumeration
Values	"Drizzle", "Ice Crystals", "Unknown", "Rain", "Ice Pellets", "Snow", "Hail", "Snow Grains", "Snow Hail"
Usage	
Definition	A selection of precipitation characteristics.
Comments	
Schema Component	<xs:simpleType name="WeatherPrecipitationType"> <xs:restriction base="xs:string"> <xs:enumeration value="Drizzle"/> <xs:enumeration value="Ice Crystals"/> <xs:enumeration value="Unknown"/> <xs:enumeration value="Rain"/> <xs:enumeration value="Ice Pellets"/> <xs:enumeration value="Snow"/> <xs:enumeration value="Hail"/> <xs:enumeration value="Snow Grains"/> <xs:enumeration value="Snow Hail"/> </xs:restriction> </xs:simpleType>
Used In	METARType.WeatherPhenomenaReport.Precipitation
Examples	<Precipitation>Drizzle</Precipitation>

Type	**WeatherObscurationType**
BaseType	Enumeration
Values	"Mist", "Sand", "Smoke", "Haze", "Volcanic Ash", "Spray", "Widespread Dust", "Other"
Usage	
Definition	A selection of qualifiers to categorize types of obscuration.
Comments	
Schema Component	<xs:simpleType name="WeatherObscurationType"> <xs:restriction base="xs:string"> <xs:enumeration value="Mist"/> <xs:enumeration value="Sand"/> <xs:enumeration value="Smoke"/> <xs:enumeration value="Haze"/> <xs:enumeration value="Volcanic Ash"/> <xs:enumeration value="Spray"/> <xs:enumeration value="Widespread Dust"/> <xs:enumeration value="Other"/> </xs:restriction> </xs:simpleType>

Used In	METARType.WeatherPhenomenaReport.Obscuration
Examples	<Obscuration>Other</Obscuration>

Type	**WeatherAddlPhenomType**
BaseType	Enumeration
Values	"Squall", "Funnel Cloud", "Sandstorm", "Tornado", "Waterspout", "Dust-storm", "Dust Whirls"
Usage	
Definition	A selection of qualifiers for weather phenomena.
Comments	
Schema Component	<xs:simpleType name="WeatherAddlPhenomType"> <xs:restriction base="xs:string"> <xs:enumeration value="Squall"/> <xs:enumeration value="Funnel Cloud"/> <xs:enumeration value="Sandstorm"/> <xs:enumeration value="Tornado"/> <xs:enumeration value="Waterspout"/> <xs:enumeration value="Duststorm"/> <xs:enumeration value="Dust Whirls"/> </xs:restriction> </xs:simpleType>
Used In	METARType.WeatherPhenomenaReport.Additional
Examples	<Additional>Dust Whirls</Additional>

Type	**SkyConditionType**
BaseType	Enumeration
Values	"Sky Clear", "Few", "Scattered", "Broken", "Overcast"
Usage	
Definition	A selection of qualifiers for sky conditions.
Comments	
Schema Component	<xs:simpleType name="SkyConditionType"> <xs:restriction base="xs:string"> <xs:enumeration value="Sky Clear"/> <xs:enumeration value="Few"/> <xs:enumeration value="Scattered"/> <xs:enumeration value="Broken"/> <xs:enumeration value="Overcast"/> </xs:restriction> </xs:simpleType>
Used In	METARType.SkyCondition
Examples	<SkyCondition>Overcast</SkyCondition>

3.1.3 EDXL Common Complex Types

Type	**ValueListType**
BaseType	xs:complexType
Restriction	None.
Usage	Use wherever a specification needs values from an externally managed list.
Definition	A ValueListType includes one ValueListURI element and one or more Value elements.
Comments	1. A lesson learned from early EDXL specification development was the need to support lists of values that may vary depending on the jurisdiction or community. The ValueListType and related structures are based on the concept that the "list" of values can be maintained externally and referenced in the EDXL standards. The reference is handled by structures which include a ValueListURI providing a unique identifier for the external "list" and then followed by a value or values from that "list". The reason "list" is quoted is because the external structure may be an ontology or other structure adopted by the jurisdiction or community rather than just a simple list. 2. A lesson learned is that enumerated types provide a brittle, hard-to-change solution to a list of types which needs to satisfy the needs of many jurisdictions.

Schema Component	`<xs:complexType name="ValueListType">`
	`  <xs:sequence>`
	`  <xs:element ref="ct:ValueListURI" minOccurs="1" maxOccurs="1"/>`
	`  <xs:element ref="ct:Value" minOccurs="1" maxOccurs="unbounded"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

Used In	Top level type

Examples	`<Keyword>`
	`  <ct:ValueListURI>urn:myagency:gov:sensors:keywords</ct:ValueListURI>`
	`  <ct:Value>SNM Detection</ct:Value>`
	`  <ct:Value>XYZ</ct:Value>`
	`</Keyword>`

Type	**ValueKeyType**
BaseType	xs:complexType
Restriction	None.
Usage	Use wherever a specification needs one single value from an externally managed list.
Definition	A ValueKeyType includes one ValueListURI element and one and only one Value element.
Comments	1. A lesson learned from early EDXL specification development was the need to support lists of values that may vary depending on the jurisdiction or community. The ValueKeyType and related structures are based on the concept that the "list" of values can be maintained externally and referenced in the EDXL standards. The reference is handled by structures which include a ValueListURI providing a unique identifier for the external "list" and then followed

	by a value from that “list”. The reason “list” is quoted is because the external structure may be an ontology or other structure adopted by the jurisdiction or community rather than just a simple list. 2. A lesson learned is that enumerated types provide a brittle, hard-to-change solution to a list of types which needs to satisfy the needs of many jurisdictions. 3. A lesson learned is that from some kinds of lists only one value is appropriate and multiple values would be an error. In this case, use `ValueKeyType` instead of `ValueListType`.

Schema Component	`<xs:complexType name="ValueKeyType">`
	`  <xs:sequence>`
	`  <xs:element ref="ct:ValueListURI" minOccurs="1" maxOccurs="1"/>`
	`  <xs:element ref="ct:Value" minOccurs="1" maxOccurs="1"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

Used In	Top level type

Examples	`<DistributionDefType>`
	`  <ct:ValueListURI>`
	`  urn:oasis:names:tc:emergency:EDXL:DE:2.0:Defaults:DistributionType`
	`  </ct:ValueListURI>`
	`  <ct:Value>Report</ct:Value>`
	`</DistributionDefType>`

Type	**ValueKeyStringPairType**
BaseType	`xs:complexType`
Usage	Use wherever a specification needs one single value from an externally managed list paired with a string.
Definition	A `ValueKeyStringPairType` includes one `ValueKeyURI` (of type `ValueKeyType` containing a `ValueListURI` and one single `Value`) followed by a `PairValue` of type `xs:string`.
Comments	

Schema Component	`<xs:complexType name="ValueKeyStringPairType">`
	`  <xs:sequence>`
	`  <xs:element name="ValueKeyURI" type="ct:ValueKeyType"`
	`  minOccurs="1" maxOccurs="1"/>`
	`  <xs:element name="PairValue" type="xs:string"`
	`  minOccurs="1" maxOccurs="1"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

Used In	Top level type

Examples	`<AValueKeyStringPair>`
	`  <ct:ValueKeyURI>`
	`  <ct:ValueListURI>http://example.com/lists/mylist</ct:ValueListURI>`
	`  <ct:Value>OneSingleValue</ct:Value>`
	`  </ct:ValueKeyURI>`
	`  <ct:PairValue>A Paired String</ct:PairValue>`
	`</AValueKeyStringPair>`

Type	**ValueKeyIntPairType**

BaseType	`xs:complexType`
Usage	Use wherever a specification needs one single value from an externally managed list paired with an integer.
Definition	A `ValueKeyIntPairType` includes one `ValueKeyURI` (of type `ValueKeyType` containing a `ValueListURI` and one single `Value`) followed by a `PairValue` of type `xs:int`.
Comments	

Schema Component	`<xs:complexType name="ValueKeyIntPairType">`
	`  <xs:sequence>`
	`  <xs:element name="ValueKeyURI" type="ct:ValueKeyType"`
	`  minOccurs="1" maxOccurs="1"/>`
	`  <xs:element name="PairValue" type="xs:int"`
	`  minOccurs="1" maxOccurs="1"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

Used In	Top level type

Examples	`<AValueKeyIntPair>`
	`  <ct:ValueKeyURI>`
	`  <ct:ValueListURI>`
	`  http://example.com/lists/mylist`
	`  </ct:ValueListURI>`
	`  <ct:Value>OneSingleValue</ct:Value>`
	`  </ct:ValueKeyURI>`
	`  <ct:PairValue>37</ct:PairValue>`
	`</AValueKeyIntPair>`

Type	**`TimePeriodType`**
BaseType	`xs:complexType`
Usage	Use wherever a specification needs to represent a period of time.
Definition	A `TimePeriodType` includes one and only one `FromDateTime` of type `ct:EDXLDateTime-Type` and one and only one `ToDateTime` element of type `ct:EDXLDateTimeType` .
Comments	1. Time periods are commonly needed in emergency standards. This type provides a simple and useful representation of a time period which can be used for uniformity throughout the EDXL specifications.

Schema Component	`<xs:complexType name="TimePeriodType">`
	`  <xs:sequence>`
	`  <xs:element name="FromDateTime" type="ct:EDXLDateTimeType"`
	`  minOccurs="1" maxOccurs="1"/>`
	`  <xs:element name="ToDateTime" type="ct:EDXLDateTimeType"`
	`  minOccurs="1" maxOccurs="1"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

Used In	Top level element

Examples	`<ATimePeriod>`
	`  <ct:FromDateTime>2009-11-15T17:53:00-05:00</ct:FromDateTime>`
	`  <ct:ToDateTime>2009-11-15T16:53:00-05:00</ct:ToDateTime>`
	`</ATimePeriod>`

Type	**PersonTimePairType**
BaseType	xs:complexType
Usage	Use wherever a specification needs to represent a person paired with a time.
Definition	A PersonTimePairType includes one and only one PersonDetails element of type ct:PersonDetailsType and one and only one TimeValue of type ct:EDXLDateTimeType.
Comments	
Schema Component	<xs:complexType name="PersonTimePairType"> <xs:sequence> <xs:element name="PersonDetails" type="ct:PersonDetailsType" minOccurs="1" maxOccurs="1"/> <xs:element name="TimeValue" type="ct:EDXLDateTimeType" minOccurs="1" maxOccurs="1"/> </xs:sequence> </xs:complexType>
Used In	Top level type
Examples	<APersonTimePair> <ct:PersonDetails> <xnl:PersonName> <xnl:NameElement>Mary Smith</n:NameElement> </xnl:PersonName> </ct:PersonDetails> <ct:TimeValue>2009-11-15T17:53:00-05:00</ct:TimeValue> </APersonTimePair>

Type	**METARType**
BaseType	xs:complexType
Usage	
Definition	A subset of the METAR weather data set.
Comments	This is a verbose form for reporting METAR weather information
Sub-elements	• StationID [1..1] of type xs:string restricted • ObservationTime [1..1] of type ct:EDXLDateTimeType • TempC [0..1] of type ct:DegreesCType • DewPointC [0..1] of type ct:DegreesCType • WindDirDegrees [0..1] of type ct:DegreesCircleType • WindSpeedkt [0..1] of type xs:int restricted • WindGustkt [0..1] of type xs:int restricted • VisibilityStatuteMI [0..1] of type xs:float restricted • AltimeterHP [0..1] of type xs:int restricted • SeaLevelPressuremb [0..1] of type xs:float restricted • WeatherPhenomenaReport [0..1] of type xs:complexType • SkyCondition [0..1] of type ct:SkyConditionType • Precip1HrIn [0..1] of type xs:float restricted • Precip3HrIn [0..1] of type xs:float restricted

	• `Precip6HrIn` [0..1] of type `xs:float` restricted • `Precip24HrIn` [0..1] of type `xs:float` restricted

Schema Component	See schema
	`<xs:complexType name="METARType">`
	`..`
	`</xs:complexType>`

Used In	Top level type

Examples	`<myMETAR>`
	`<ct:StationID>KEYF</ct:StationID>`
	`<ct:ObservationTime>2011-04-23T01:41:00+00:00</ct:ObservationTime>`
	`<ct:TempC>37.2</ct:TempC>`
	`<ct:DewpointC>10.0</ct:DewpointC>`
	`<ct:WindDirDegrees>32.3</ct:WindDirDegrees>`
	`<ct:WindSpeedkt>20</ct:WindSpeedkt>`
	`<ct:WindGustkt>50</ct:WindGustkt>`
	`<ct:VisibilityStatuteMI>1.0</ct:VisibilityStatuteMI>`
	`<ct:AltimeterHP>800</ct:AltimeterHP>`
	`<ct:SeaLevelPressuremb>800</ct:SeaLevelPressuremb>`
	`<ct:WeatherPhenomenaReport>`
	`...`
	`</ct:WeatherPhenomenaReport>`
	`<ct:SkyCondition>Overcast</ct:SkyCondition>`
	`<ct:Precip1HrIn>00.01</ct:Precip1HrIn>`
	`<ct:Precip3HrIn>01.00</ct:Precip3HrIn>`
	`<ct:Precip6HrIn>01.23</ct:Precip6HrIn>`
	`<ct:Precip24HrIn>02.25</ct:Precip24HrIn>`
	`</myMETAR>`

Sub-Element	`[METARType.]`**`StationID`**
Type	`xs:string` restricted
Restriction	Pattern "[A-Z]{4}"
Usage	[1..1]
Definition	Identifies the reporting station
Comments	Four-character ICAO Location Indicator

Schema Component	`<xs:element name="StationID" minOccurs="1">`
	`<xs:simpleType>`
	`<xs:restriction base="xs:string">`
	`<xs:pattern value="[A-Z]{4}"/>`
	`</xs:restriction>`
	`</xs:simpleType>`
	`</xs:element>`

Used In	`METARType`
Examples	`<ct:StationID>KEYF</ct:StationID>`

Sub-Element	`[METARType.]`**`WindSpeedkt`**

Type	`xs:int` restricted
Restriction	Range [0 .. 300]
Usage	[0..1]
Definition	Wind speed in knots
Comments	

Schema Component	`<xs:element name="WindSpeedkt" minOccurs="0">`
	`  <xs:simpleType>`
	`  <xs:restriction base="xs:int">`
	`  <xs:minInclusive value="0"/>`
	`  <xs:maxInclusive value="300"/>`
	`  </xs:restriction>`
	`  </xs:simpleType>`
	`</xs:element>`

Used In	`METARType`
Examples	`<ct:WindSpeedkt>20</ct:WindSpeedkt>`

Sub-Element	`[METARType.]`**`WindGustkt`**
Type	`xs:int` restricted
Restriction	Range [0 .. 300]
Usage	[0..1]
Definition	Speed of wind gusts in knots
Comments	

Schema Component	`<xs:element name="WindGustkt" minOccurs="0">`
	`  <xs:simpleType>`
	`  <xs:restriction base="xs:int">`
	`  <xs:minInclusive value="0"/>`
	`  <xs:maxInclusive value="300"/>`
	`  </xs:restriction>`
	`  </xs:simpleType>`
	`</xs:element>`

Used In	`METARType`
Examples	`<ct:WindGustkt>50</ct:WindGustkt>`

Sub-Element	`[METARType.]`**`VisibilityStatuteMI`**
Type	`xs:float` restricted
Restriction	Range [0 .. 10.0]
Usage	[0..1]
Definition	Visibility in Statute Miles
Comments	

Schema Component	<xs:element name="VisibilityStatuteMI" minOccurs="0"> <xs:simpleType> <xs:restriction base="xs:float"> <xs:minInclusive value="0.0"/> <xs:maxInclusive value="10.0"/> </xs:restriction> </xs:simpleType> </xs:element>
Used In	METARType
Examples	<ct:VisibilityStatuteMI>1.0</ct:VisibilityStatuteMI>

Sub-Element	[METARType.]**AltimeterHP**
Type	xs:int restricted
Restriction	Range [800 .. 1200]
Usage	[0..1]
Definition	Altimeter measurement in hectopascal
Comments	
Schema Component	<xs:element name="AltimeterHP" minOccurs="0"> <xs:simpleType> <xs:restriction base="xs:int"> <xs:minInclusive value="800"/> <xs:maxInclusive value="1200"/> </xs:restriction> </xs:simpleType> </xs:element>
Used In	METARType
Examples	<ct:AltimeterHP>800</ct:AltimeterHP>

Sub-Element	[METARType.]**SeaLevelPressuremb**
Type	xs:int restricted
Restriction	Range [800 .. 1200]
Usage	[0..1]
Definition	Atmospheric pressure at sea level in millibar
Comments	1 mb = 1 hPa
Schema Component	<xs:element name="SeaLevelPressuremb" minOccurs="0"> <xs:simpleType> <xs:restriction base="xs:int"> <xs:minInclusive value="800"/> <xs:maxInclusive value="1200"/> </xs:restriction> </xs:simpleType> </xs:element>

Used In	METARType
Examples	<ct:SeaLevelPressuremb>800</ct:SeaLevelPressuremb>

Sub-Element	[METARType.]**WeatherPhenomenaReport**
Type	xs:complexType
Usage	[0..1]
Definition	
Comments	
Sub-elements	• Qualifier [0..1] of type ct:WeatherQualifierType • Descriptor [0..1] of type ct:WeatherDescriptorType • Precipitation [0..1] of type ct:WeatherPrecipitationType • Obscuration [0..1] of type ct:WeatherObscurationType • Additional [0..1] of type ct:WeatherAddlPhenomType
Schema Component	<xs:element name="SeaLevelPressuremb" minOccurs="0"> <xs:simpleType> <xs:restriction base="xs:int"> <xs:minInclusive value="800"/> <xs:maxInclusive value="1200"/> </xs:restriction> </xs:simpleType> </xs:element>
Used In	METARType
Examples	<ct:WeatherPhenomenaReport> <ct:Qualifier>Light</ct:Qualifier> <ct:Descriptor>Showers</ct:Descriptor> <ct:Precipitation>Drizzle</ct:Precipitation> <ct:Obscuration>Other</ct:Obscuration> <ct:Additional>Dust Whirls</ct:Additional> </ct:WeatherPhenomenaReport>

Sub-Element	[METARType.]**Precip1HrIn**
Type	xs:float restricted
Restriction	Pattern "[0-9][0-9].[0-9][0-9]"
Usage	[0..1]
Definition	Amount of rain fall in 1 h, in inches
Comments	
Schema Component	<xs:element name="Precip1HrIn" minOccurs="0"> <xs:simpleType> <xs:restriction base="xs:float"> <xs:pattern value="[0-9][0-9].[0-9][0-9]"/> </xs:restriction> </xs:simpleType>

	`</xs:element>`
Used In	`METARType`
Examples	`<ct:Precip1HrIn>00.01</ct:Precip1HrIn>`

Sub-Element	`[METARType.]`**`Precip3HrIn`**
Type	`xs:float` restricted
Restriction	Pattern "[0-9][0-9].[0-9][0-9]"
Usage	[0..1]
Definition	Amount of rain fall in 3 h, in inches
Comments	

Schema Component	`<xs:element name="Precip3HrIn" minOccurs="0">`
	`  <xs:simpleType>`
	`  <xs:restriction base="xs:float">`
	`  <xs:pattern value="[0-9][0-9].[0-9][0-9]"/>`
	`  </xs:restriction>`
	`  </xs:simpleType>`
	`</xs:element>`

Used In	`METARType`
Examples	`<ct:Precip3HrIn>01.00</ct:Precip3HrIn>`

Sub-Element	`[METARType.]`**`Precip6HrIn`**
Type	`xs:float` restricted
Restriction	Pattern "[0-9][0-9].[0-9][0-9]"
Usage	[0..1]
Definition	Amount of rain fall in 6 h, in inches
Comments	

Schema Component	`<xs:element name="Precip6HrIn" minOccurs="0">`
	`  <xs:simpleType>`
	`  <xs:restriction base="xs:float">`
	`  <xs:pattern value="[0-9][0-9].[0-9][0-9]"/>`
	`  </xs:restriction>`
	`  </xs:simpleType>`
	`</xs:element>`

Used In	`METARType`
Examples	`<ct:Precip6HrIn>01.23</ct:Precip6HrIn>`

Sub-Element	`[METARType.]`**`Precip24HrIn`**

Type	`xs:float` restricted
Restriction	Pattern "[0-9][0-9].[0-9][0-9]"
Usage	[0..1]
Definition	Amount of rain fall in 24 h, in inches
Comments	

Schema Component	`<xs:element name="Precip24HrIn" minOccurs="0">`
	`  <xs:simpleType>`
	`  <xs:restriction base="xs:float">`
	`  <xs:pattern value="[0-9][0-9].[0-9][0-9]"/>`
	`  </xs:restriction>`
	`  </xs:simpleType>`
	`</xs:element>`

Used In	`METARType`
Examples	`<ct:Precip24HrIn>02.25</ct:Precip24HrIn>`

Type	**`WeatherInfoType`**
BaseType	`xs:complexType`
Usage	Use wherever weather info is needed in a specification.
Definition	A container to transmit predefined weather info with free format remarks and concerns
Comments	1. METAR string: raw METAR data, "the most popular format in the world for the transmission of weather data. It is highly standardized through International Civil Aviation Organization (ICAO), which allows it to be understood throughout most of the world" [Wikipedia] 2. METAR readings: a more verbose formatted set of weather data
Sub-elements	• METARString [0..1] of type xs:string • METARReadings [0..1] of type ct:METARType • WeatherRemarks [0..1] of type xs:string • WeatherConcerns [0..1] of type xs:string

Schema Component	`<xs:complexType name="WeatherInfoType">`
	`  <xs:sequence>`
	`  <xs:element name="METARString" type="xs:string" minOccurs="0"/>`
	`  <xs:element name="METARReadings" type="ct:METARType" minOccurs="0"/>`
	`  <xs:element name="WeatherRemarks" type="xs:string" minOccurs="0"/>`
	`  <xs:element name="WeatherConcerns" type="xs:string" minOccurs="0"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

Used In	Top level type

Examples	`<ct:WeatherInfo`
	`xsi:schemaLocation="urn:oasis:names:tc:emergency:edxl:ct:1.0`
	`EDXL_Common_Types_wd02_dpm.xsd" xmlns:xsi="http://www.w3.org/2001/XMLS-`
	`chema-instance" xmlns:ct="urn:oasis:names:tc:emergency:edxl:ct:1.0">`
	`  <ct:METARString>KEYF 222355Z AUTO 00000KT 4SM BR 17/17 A3022 RMK AO2`
	`T01700170</ct:METARString>`
	`  <ct:METARReadings>`
	`  <ct:StationID>KEYF</ct:StationID>`
	`  <ct:ObservationTime>2011-04-23T01:41:00+00:00</ct:ObservationTime>`
	`  <ct:TempC>37.2</ct:TempC>`
	`  <ct:DewpointC>10.0</ct:DewpointC>`

	<ct:WindDirDegrees>32.3</ct:WindDirDegrees> <ct:WindSpeedkt>20</ct:WindSpeedkt> <ct:WindGustkt>50</ct:WindGustkt> <ct:VisibilityStatuteMI>1.0</ct:VisibilityStatuteMI> <ct:AltimeterHP>800</ct:AltimeterHP> <ct:SeaLevelPressuremb>800</ct:SeaLevelPressuremb> <ct:WeatherPhenomenaReport> <ct:Qualifier>Light</ct:Qualifier> <ct:Descriptor>Showers</ct:Descriptor> <ct:Precipitation>Drizzle</ct:Precipitation> <ct:Obscuration>Other</ct:Obscuration> <ct:Additional>Dust Whirls</ct:Additional> </ct:WeatherPhenomenaReport> <ct:SkyCondition>Overcast</ct:SkyCondition> <ct:Precip1HrIn>00.01</ct:Precip1HrIn> <ct:Precip3HrIn>01.00</ct:Precip3HrIn> <ct:Precip6HrIn>01.23</ct:Precip6HrIn> <ct:Precip24HrIn>02.25</ct:Precip24HrIn> </ct:METARReadings> <ct:WeatherRemarks>This is weather</ct:WeatherRemarks> <ct:WeatherConcerns> I am concerned it may change, and that scares me... </ct:WeatherConcerns> </ct:WeatherInfo>

Type	**OrganizationInformationType**
BaseType	Extends xpil:OrganisationDetailsType
Usage	Use wherever a specification needs to specify information about an organization.
Definition	The container type for organization information elements. The OrganizationInformationType includes at least one xnl:OrganisationName and optionally Addresses, ContactNumbers, ElectronicAddressIdentifiers and OrganisationInfo. See the OASIS EM CIQ Profile for details.
Comments	1. Note that some elements use the American spelling “Organization” and some the English spelling “Organisation”.
Schema Component	<xs:complexType name="OrganizationInformationType"> <xs:complexContent> <xs:extension base="xpil:OrganisationDetailsType"/> </xs:complexContent> </xs:complexType>
Used In	Top level type
Examples	<AnOrganizationInformation> <xnl:OrganisationName> <xnl:NameElement>Corporation XYZ</xnl:NameElement> </xnl:OrganisationName> </AnOrganizationInformation>

Type	**PersonDetailsType**
Type	xpil:PersonDetailsType

Usage	Used in the `PersonTimePairType`.
Definition	`A container for defining the unique characteristics of a person only.` `PersonDetailsType` is an extension of `xpil:PersonDetailsType` which is defined in the OASIS EM TC CIQ profile xpil schema to include at least one PersonName, and optionally one `Addresses`, `ContactNumbers`, `ElectronicAddressIdentifiers` and `Identifers`. For more information, see the OASIS EM TC CIQ profile.
Comments	1. See the EM-TC CIQ Profile

Schema Component	`<xs:complexType name="PersonDetailsType">`
	`  <xs:complexContent>`
	`  <xs:extension base="xpil:PersonDetailsType"/>`
	`  </xs:complexContent>`
	`</xs:complexType>`

Used In	`PersonTimePairType`

Examples	`<APersonDetails>`
	`  <ct:PersonDetails>`
	`  <xnl:PersonName>`
	`  <xnl:NameElement>Mary Smith</n:NameElement>`
	`  </xnl:PersonName>`
	`  </ct:PersonDetails>`
	`</APersonDetails>`

3.1.4 EDXL Common Top Level Elements

Element	**`ValueListURI`**
Type	`ct:ValueListURIType`
Usage	Used to denote the URI of a `ValueListType` and related types
Definition	A URI referencing an externally-managed list of values.
Comments	1. A lesson learned from early EDXL specification development was the need to support lists of values that may vary depending on the jurisdiction or community. The `ValueListType` and related structures are based on the concept that the "list" of values can be maintained externally and referenced in the EDXL standards. The reference is handled by structures which include a `ValueListURI` providing a unique identifier for the external "list" and then followed by a value or values from that list. The reason "list" is quoted is because the external structure may be an ontology or other structure adopted by the jurisdiction or community rather than just a simple list.
Schema Component	`<xs:element name="ValueListURI" type="ValueListURIType"/>`

Used In	`ValueListType`
	`ValueKeyType`
	`ValueKeyStringPairType`
	`ValueKeyIntPairType`

Examples	`<ct:ValueListURI>`http://example.com/mylist`</ct:ValueListURI>`
	
	`<ct:ValueListURI>`
	`  urn:oasis:names:tc:emergency:edxl:de:2.0:Defaults:DistributionType`
	`</ct:ValueListURI>`

Element	**Value**
Type	ct:ValueType
Usage	Used to denote the value(s) of a ValueListType and related types
Definition	A string value from an externally-managed list of values referenced by a ValueListURI.
Comments	1. A lesson learned from early EDXL specification development was the need to support lists of values that may vary depending on the jurisdiction or community. The ValueListType and related structures are based on the concept that the "list" of values can be maintained externally and referenced in the EDXL standards. The reference is handled by structures which include a ValueListURI providing a unique identifier for the external "list" and then followed by a value or values from that "list". The reason "list" is quoted is because the external structure may be an ontology or other structure adopted by the jurisdiction or community rather than just a simple list.
Schema Component	<xs:element name="Value" type="ValueType"/>
Used In	ValueListType ValueKeyType ValueKeyStringPairType ValueKeyIntPairType
Examples	<ct:Value>SomeValue</ct:Value>

4 Conformance

The last numbered section in the specification must be the Conformance section. Conformance Statements/Clauses go here.

TBD

A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Don McGarry, MITRE Corp., Member

Jeff Waters, DoD, Member

Werner Joerg, IEM, Inc., Member

B. Non-Normative Text

C. Revision History

Revision	Date	Editor	Changes Made
WD01	03/02/2011	Jeff Waters	Initial Setup
WD02	04/21/2011	Werner Joerg	Adaptation to new schema; ready for TC review
WD03	05/02/2011	Werner Joerg	Expanded WeatherInfo; ready for TC review