

# **CPSIN XML Guide**

## **Version 2-0-0**

**DRAFT**

**2005-09-29**

# Contents

CPSIN XML .....	3
Dictionary Data Model .....	5
CPSIN XML Model .....	8
CPSIN XML Data Exchange Standard .....	16
CPSIN XML Global Schema Structure .....	18
CPSIN Transaction Schemas .....	22
Codes .....	23
Multilingualism .....	26
CPSIN XML Viewer .....	27
CPSIN Browser.....	28
CPSIN XML Download .....	30
Using CPSIN XML .....	31
CPSIN XML Schema Design Standard .....	41

# CPSIN XML

*"Smart data structures and dumb code works a lot better than the other way around"*

*-- Eric S. Raymond, "The Cathedral and the Bazaar"*

This document provides a detailed, technical presentation of the CPSIN XML data exchange strategy developed by the Data Standards Secretariat. It is an essential companion to the CPSIN XML Global Schema modules which implement the CPSIN Data Dictionary as an XML data exchange standard. This document is aimed at a technical audience and full comprehension of some content requires extensive knowledge of XML Schema and related technologies.

## Introduction

As part of the implementation of the Canada Public Safety Information Network (CPSIN), the Data Standards Secretariat (DSS) is facilitating the development of a data dictionary with a Working Group representing the CPSIN Charter signatories, namely RCMP, JC, CSC, NPB, CIC, CBSA, and CCJS. The CPSIN Data Dictionary contains a common set of public safety definitions and terminology intended to provide the vocabulary for standardized, electronic data exchange among CPSIN partners. It currently contains over four hundred data elements and one hundred code value sets. The core CPSIN Data Dictionary aims to meet the data exchange needs of the federal public safety community, with a longer term objective of expanding its scope to the provincial and municipal tiers, which is already under way.

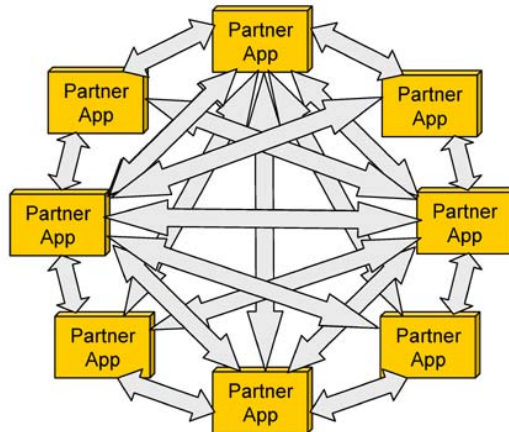
XML (eXtensible Markup Language) has become the de facto standard for data exchange over the internet and is the technology that CPSIN partners are focusing on for the exchange of information between their applications. Therefore, the DSS has implemented the CPSIN Data Dictionary in an XML Schema format that can be directly referenced in the design of CPSIN XML transactions and the validation of CPSIN XML files.

This document explains the principles of the CPSIN Data Dictionary Data Model and how they have been implemented in XML Schema. It also provides detailed documentation of the CPSIN XML Schema modules, covering the design principles and standards used in their development and how they are intended to be used and extended. In addition, it introduces the XML support tools available, namely the CPSIN XML Viewer and the CPSIN Browser.

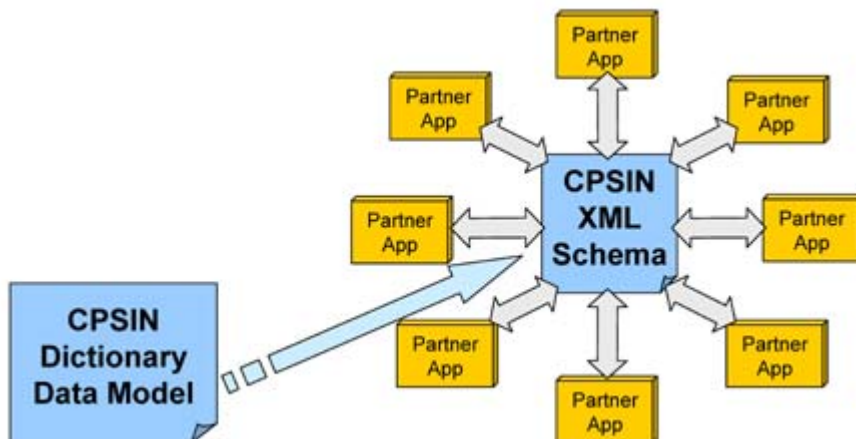
## The Goal of CPSIN XML

The goal of CPSIN XML is to give the CPSIN community the ability to move away from the traditional world of custom bilateral interfaces, where every application defines its own interface from scratch and requires every interfacing application to accommodate it.

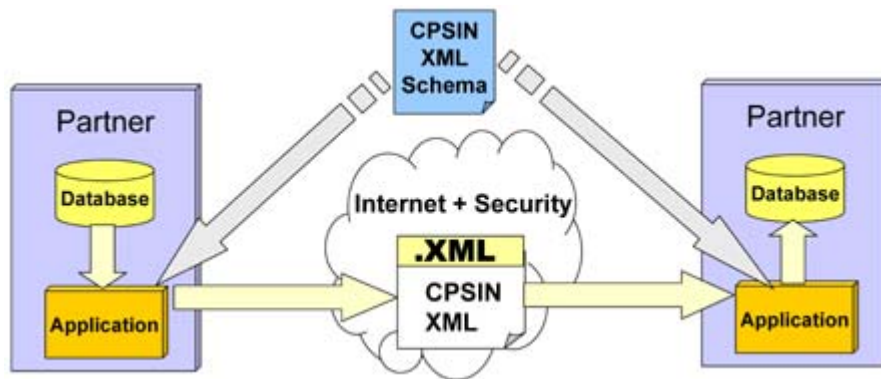
[Enlarge Image](#)



The lack of a common philosophy and common components means that these solutions are inflexible and offer no reuse. This places a huge burden of redundant analysis, development and maintenance on CPSIN partners and makes information sharing unnecessarily difficult, expensive and slow to implement.



CPSIN XML offers partner organizations the ability to exchange a common set of data elements and reusable data structures defined in the CPSIN XML Global Schema. The authoritative source of this metadata is the CPSIN Data Dictionary and its Data Model, which will also drive any future implementations in other technologies. This means CPSIN partners only have to learn one language for data exchange, instead of learning every dialect in use by CPSIN applications.

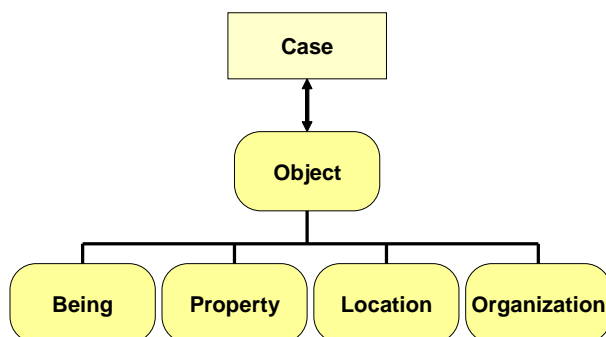


The CPSIN XML Global Schema can then be used to design the XML transactions for data exchange between CPSIN applications, regardless of the application architecture being implemented. At runtime, CPSIN XML schemas can also be used to validate the structure and content of the XML documents exchanged.

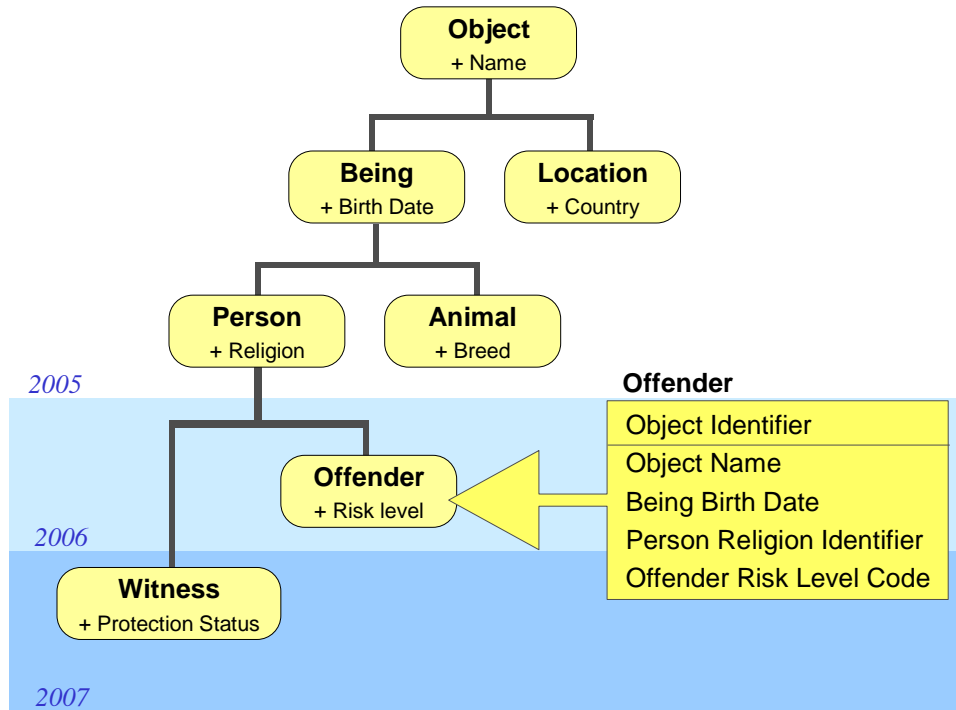
## Dictionary Data Model

### Taxonomy

The CPSIN Data Dictionary Working Group organized their work in a series of topics, namely Person, Vehicle, Case, Property, Location, and Organization. To keep the task manageable and to control the potential for redundancy, an object taxonomy was developed. Objects are broken down into a multi-level hierarchy where each subtype inherits the common data characteristics defined at higher levels in its branch of the taxonomy. This brings discipline to the process of selecting truly essential data elements so that in the end every data concept only appears once. In the process of refining the taxonomy, two significant changes were made. First, the Person topic was renamed to Being to accommodate the common data characteristics of both people and animals. Secondly, Vehicle was absorbed under Property. This resulted in the current top level Object subtypes of Being, Property, Location, and Organization, each of which breaks down into further levels of subtype.



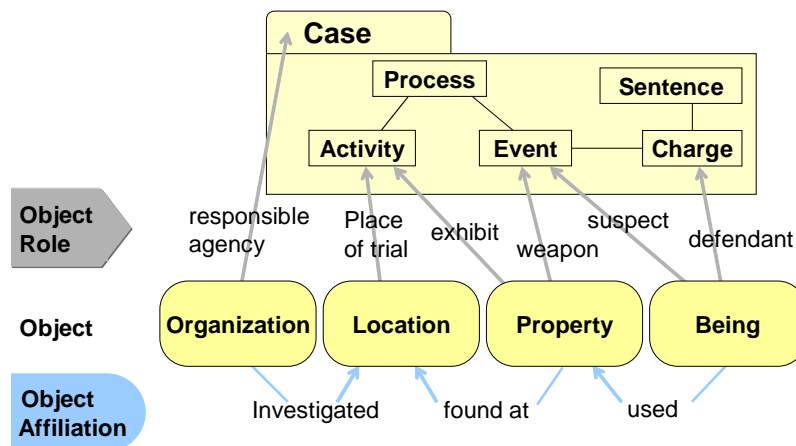
The taxonomy provides a stable conceptual framework that should allow extensibility without significant modification to existing content. It is inevitably a work in progress and further levels will need be defined over time, as more requirements are addressed and the CPSIN partnership expands.



The object taxonomy is incorporated into the CPSIN Data Dictionary Data Model, which is an entity-relationship model maintained in ERwin and available in the CPSIN Data Model section. Note that CPSIN data element names follow the ISO/IEC 11179 3-part naming convention and are preserved intact, in camel case, within the XML environment.

## Case

The Data Model supports a generic paradigm of public safety casework, applicable to all CPSIN partners. The Case topic supports the administrative backbone of a case, consisting of Events that trigger Processes and their Activities, using the basic concepts and vocabulary of the Workflow Management Coalition (WfMC). Each partner can contribute a specific set of values for the Events, Processes and Activities for their domain. The Case topic also provides a high-level view of key justice sector outcomes, such as Events, Charges, and Sentences.



## Object Role

Objects can be given multiple associations with a Case and its Events, Activities, and Charges, etc. by assigning the Object an Object Role for each context. For instance a Person (a subtype of Being) may have an Object Role of “suspect” in an Event and an Object Role of “defendant” in a Charge. A Vehicle (a subtype of Property) may have an Object Role of “weapon” in an Event and an Object Role of “exhibit” in an Activity.

## Object Affiliation

Similarly, Objects can be directly associated with one another and multi-link chains can be formed where required. For instance, a Person may have an Object Affiliation of “owner of” with Property, which may in turn have an Object Affiliation of “recovered at” with a Location.

It is the ability to assign Object Roles and Object Affiliations to any Object subtype that gives the Data Model its flexibility. It permits a limited number of context-free data elements to be named and defined and then reused in many different business contexts. There will be rapid growth in the types of Object Role and Object Affiliation required over time. However, this will not require changes to be made to the structure of the Data Model or the resulting XML schemas.

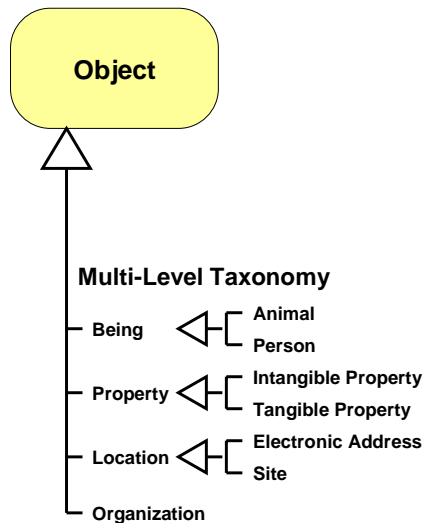
See the CPSIN Data Dictionary publication for a fuller explanation of these concepts and documentation on each Data Model topic.

# CPSIN XML Model

This section explains how the concepts of the CPSIN Data Model have been implemented as a global XML schema whose basic constructs provide a grammar for building XML data exchange transactions for the public safety sector.

## Object

The Object construct is a class hierarchy with multiple levels of inheritance that implements the CPSIN Data Model Object taxonomy.

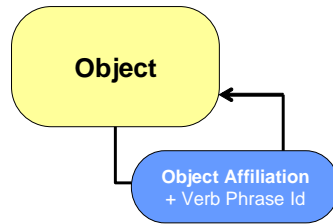


The `Object` element itself does not appear in XML instance files. Instead, an appropriate `Object` subtype is used which inherits all the properties relevant to its branch and level in the class hierarchy. The selected subtype does not necessarily have to be at the leaf level.

This is implemented by means of a substitution group with the abstract `Object` as its head element. Every `Object` subtype is a member of this substitution group and has a named complex type derived by extension from others to provide the necessary inheritance, with `ObjectSuperType` as their common origin.

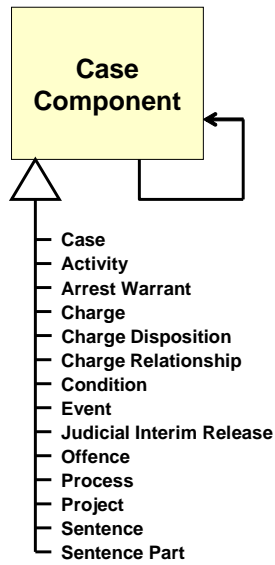
## Object Affiliation

This is the construct that allows Objects to be chained together in multi-link relationships. `ObjectAffiliation` is a sub-element of `Object`, which itself contains an `Object` element, plus a `verbPhraseId` attribute to explain the nature of the relationship between the parent and child Objects.



## Case Component

The `CaseComponent` element itself does not appear in XML instance files. Instead, any of its subtypes listed below can be specified. These are the major entities from the Case Topic of the CPSIN Data Model. These elements can be directly nested within each other to provide complete flexibility in the building of Case data structures for XML transactions.

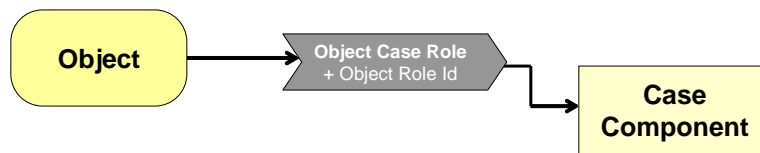


This is implemented by means of a substitution group with the abstract `CaseComponent` as its head element. Every subtype is a member of this substitution group and has a named complex type derived by extension from `CaseComponentType` to provide the inheritance of their common properties.

## Object Role

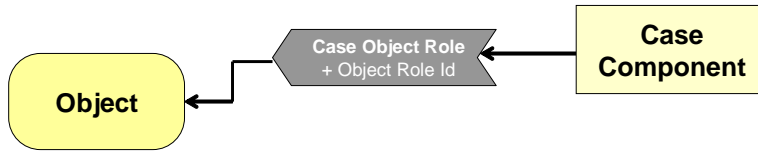
This is the construct that allows an Object to be associated with a Case Component with a specific context. Because XML data is structured hierarchically for navigation in only one direction, the CPSIN Data Model concept of Object Role has been implemented as two XML elements. Both carry the `objectRoleId` attribute that explains why the Object instance is associated with the CaseComponent. The choice of element depends on whether the parent element is the Object or the CaseComponent:

ObjectCaseRole



This is a sub-element of `Object` that may contain any kind of `CaseComponent`.

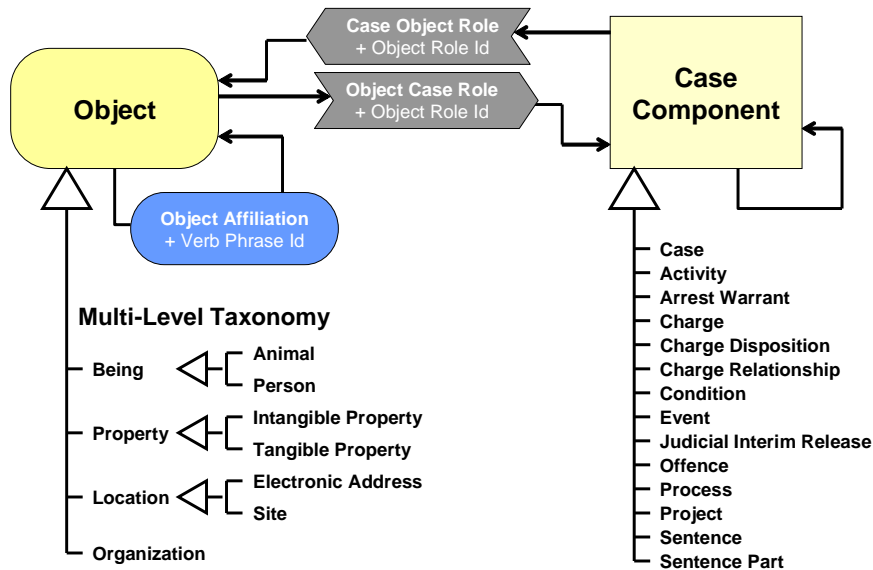
## CaseObjectRole



This is a sub-element of **CaseComponent** that may contain any kind of **Object**.

## CPSIN XML Grammar Model

When these constructs are combined, as illustrated in this model, they provide a consistent grammar for building data structures from CPSIN Objects and Case Components that can meet any public safety data exchange requirement.



Note again that the CPSIN XML Global Schema defines both `Object` and `CaseComponent` as abstract elements. Therefore, they do not appear in instance files because they are always substituted by an appropriate subtype.

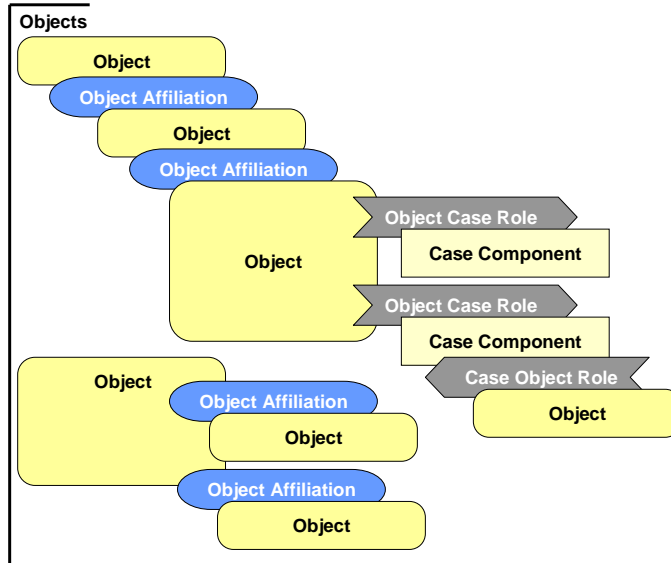
## Roots and Containers

Every XML document must have a single root element which forms the starting point of its hierarchical structure. The root element of a CPSIN XML document may be any kind of Case Component (such as a Case or Event) or any kind of Object (such as a Person or Vehicle).

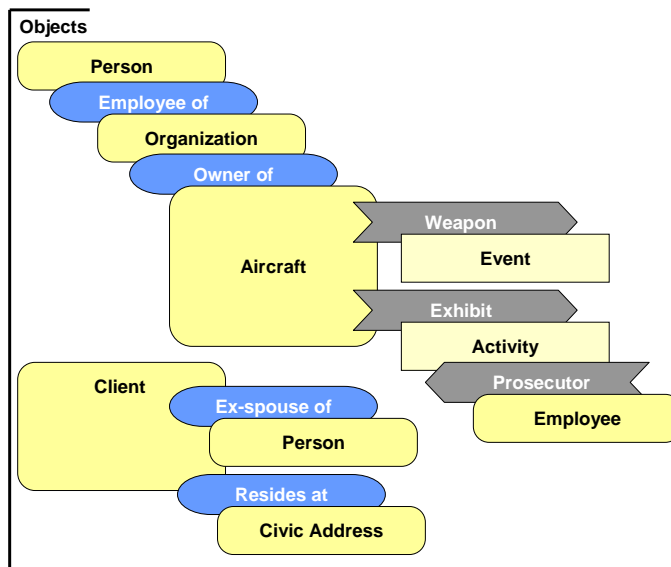
To document a list of things in a single XML document requires a container element to act as their common root. CPSIN provides two such container elements, `Cases` and `Objects`.

## Object-centric Example

This CPSIN XML document is structured as a list of Objects. Objects can contain multiple Object Affiliations to other Objects and multiple Object Case Roles to Case Components. The term Object-centric only refers to the root or container element because the contained elements can include any combinations of Case Components and Objects that are valid in CPSIN grammar, nested to any depth.

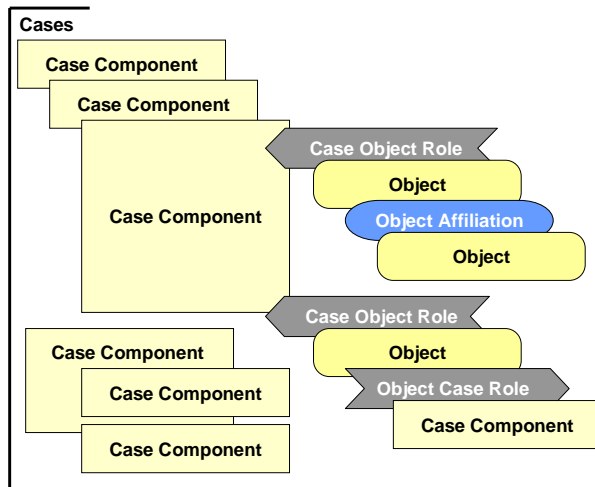


Using typical values, this example could document the following:

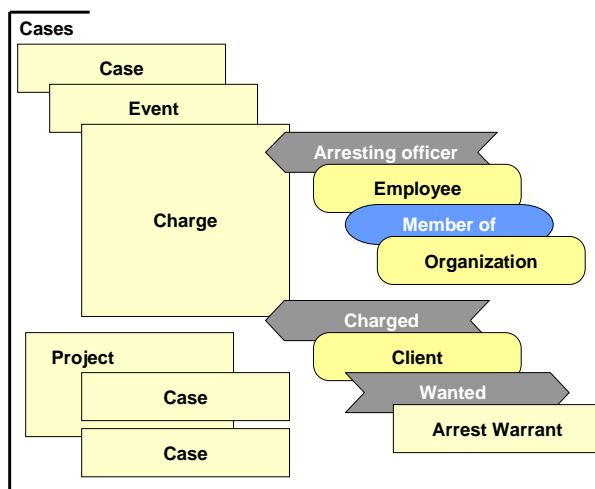


## Case-centric Example

This CPSIN XML document is structured as a list of Cases. Case Components can directly contain other Case Components and multiple Case Object Roles to Objects. The term Case-centric only refers to the root or container element because the contained elements can include any combinations of Case Components and Objects that are valid in CPSIN grammar, nested to any depth.



Using typical values, this example could document the following:



# CPSIN XML Data Exchange Standard

## CPSIN XML Manifesto

The CPSIN XML Exchange Standard will provide the CPSIN community with a standard vocabulary that:

1. Meets the common data exchange requirements of CPSIN partners.
2. Maximizes the flexibility of data exchange.
3. Facilitates clean and easily maintained partner extensions.
4. Provides an acceptable level of performance with current generation technology.
5. Can be upgraded in a timely fashion using a consultative and open process.
6. Continues to support previous versions as required.

These are the objectives that the Data Standards Secretariat will always strive to meet in the evolution of the XML data exchange environment for CPSIN.

## XML Schema Levels

Three levels of XML schema will exist in a fully-fledged CPSIN XML environment:

1. The CPSIN XML Global Schema
2. CPSIN XML transaction Schemas
3. CPSIN XML partner reuse and extension schemas

## The CPSIN XML Global Schema

The CPSIN XML Global Schema implements the full scope of the CPSIN Data Dictionary and Data Model in a set of XML schema modules. This provides a general-purpose data exchange vocabulary that CPSIN partners can immediately begin using in their applications. It can be used at any level of granularity, from complete topics, to Object subtypes such as a Person or Civic Address, smaller schema fragments, or even individual data elements. Obviously, the larger the schema fragments used, the greater the resulting standardization benefits and productivity. In particular, adherence to the CPSIN XML grammar when constructing XML transactions from the Global Schema will ensure that they:

- are immediately comprehensible to all partners
- can be reused in multiple contexts
- can automatically use CPSIN tools without customization, e.g. the CPSIN Browser and class generators.

The CPSIN Global Schema is relaxed, meaning that it does not specify validation constraints beyond the most fundamental cardinality restrictions for the data structures. It is not possible or even desirable to attempt to validate all the business rules of the CPSIN community at this global level.

## CPSIN Transaction Schemas

CPSIN Transaction Schemas will be developed and officially adopted by the CPSIN partner community for specific XML queries, responses or workflow transactions.

Since they will typically be single-purpose and business context-specific, they may feature more detailed validation constraints.

## Partner Re-Use and Extension Schemas

These are XML schemas developed by CPSIN partner organizations that incorporate CPSIN XML schema content. This encompasses both simple reuse of schema fragments at any level of granularity as well as additions to CPSIN schema content for a partner environment.

Such extensions to the standard may be temporary while new content goes through the CPSIN approval process, or may remain permanently outside the official CPSIN scope. The official extension techniques are presented later in this guide. In any event, the DSS team should be consulted before any form of extension is considered to ensure that existing features are being fully exploited and to make the team aware of additional needs in the community.

Simply adhering to the CPSIN XML grammar of the Global Schema for constructing XML transactions will yield the highest long term productivity. Optimizing data structures for a specific application context may be perceived as beneficial, e.g. bandwidth savings. However, this will typically be greatly outweighed by the development and maintenance costs of customization for each partner involved, including the inability to use CPSIN tools straight out of the box.

Similarly, using custom schemas to perform complete data validation is not recommended. W3C XML Schema derivation by restriction is cumbersome, maintenance-prone, and impractical for deeply nested structures. Its features for co-occurrence constraints are limited, and totally absent for co-occurrence by value. Therefore, schema validation will rarely meet all the validation requirements of the receiving application and consequently not eliminate the need for validation code. Where full validation is required prior to application processing, the DSS recommends the rule-based ISO Schematron [<http://www.schematron.com>] as the best complement to W3C XML Schema structures.

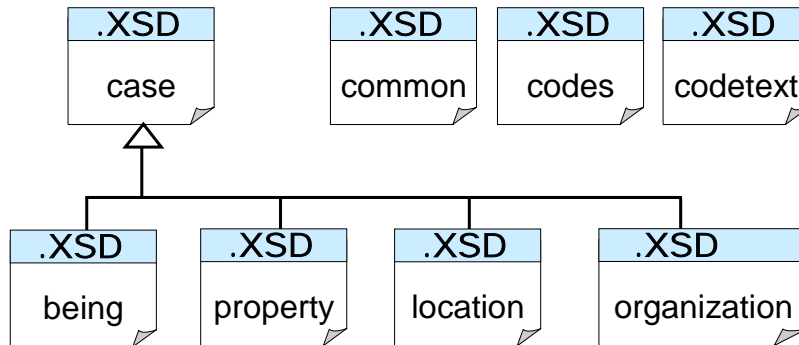
## Standard Compliance

A much welcomed feature of the W3C XML Schema 1.0 Recommendation is the ability to make references to multiple namespaces. This provides a technique for cleanly identifying imported components and avoiding the error-prone processes of cloning and update synchronization. Therefore, referencing CPSIN schemas and schema fragments from CPSIN namespaces, and using the official versions of those schemas, is the only recognized technique for compliance with the CPSIN XML Data Exchange Standard.

# CPSIN XML Global Schema Structure

## Modules

The CPSIN XML Global Schema has a modular structure designed to localize the impact of upgrades and to minimize runtime memory consumption.



The first five modules reflect the content of the CPSIN Data Model topics and the last three have specific global purposes:

File Name	Description
case.xsd	The administrative backbone of a case, implemented as Case Components, such as Projects, Cases, Events, Processes, and Activities, plus key justice sector outcomes, such as Offences, Charges, Dispositions, and Sentences. This module also includes the abstract Object, which provides the common data characteristics inherited throughout the Object taxonomy such as the Name and Unit of Measure constructs, Object Roles and Object Affiliations.
being.xsd	The common data characteristics of people and animals and their subtypes.
property.xsd	Any article of financial or informational value. Includes body parts.
location.xsd	Places and addresses, both physical and electronic.
organization.xsd	Any organized group or non-individual legal entity.
codes.xsd	XML attributes for CPSIN language-neutral numeric codes and XML elements for terse codes from other sources.
codetext.xsd	XML elements for the language-specific text version of codes.
common.xsd	Named simple and complex types referenced throughout the schema which are the XML implementation of the CPSIN Data Dictionary data types and domains.

## Schema Version

CPSIN XML schema files (.xsd extension) for both Global Schema modules and Transaction Schemas, are identified by the following path and name convention which can be used to download them from the DSS Website:

[http://\[Domain\]/\[Schema\]/\[Major\]-\[Minor\]-\[Incremental\]](http://[Domain]/[Schema]/[Major]-[Minor]-[Incremental])

Name Component	Definition
Domain	<a href="http://tools.dss-snd.gc.ca/cpsin/ns/">http://tools.dss-snd.gc.ca/cpsin/ns/</a>
Schema	gl (for Global)
Major	A version number incremented for each release of the schema featuring extensive new functionality or extensive changes to content or conventions. Typically not backward compatible.
Minor	A version number incremented for each release of the schema within a major version where XML elements or attributes have been added or modified, beyond changes to code value sets.
Incremental	A version number incremented for each release of the schema within a minor version when code value sets have been updated.

Example: <http://tools.dss-snd.gc.ca/cpsin/ns/gl/1-2-3>

## XML Schema Filenames

CPSIN XML schema files (.xsd extension) for both Global Schema modules and Transaction Schemas, are identified by the following path and name convention which can be used to download them from the DSS Website:

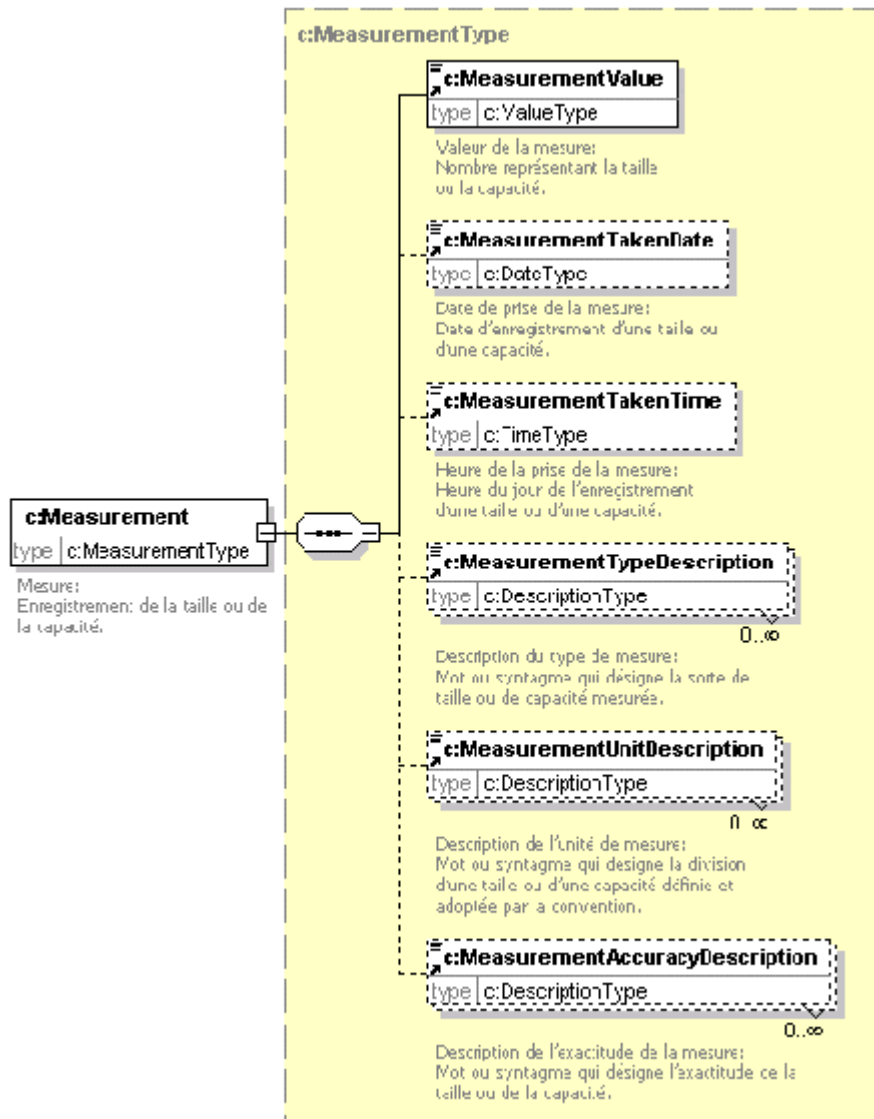
[http://\[Schema Version\]/\[Format\]/\[Name\].xsd](http://[Schema Version]/[Format]/[Name].xsd)

Name Component	Definition
Schema Version	Defined above
Format	sx = Syntax only, recommended for production use  fr = Annotated with French Dictionary names and definitions  en = Annotated with English Dictionary names and definitions
Name	All lower-case filename

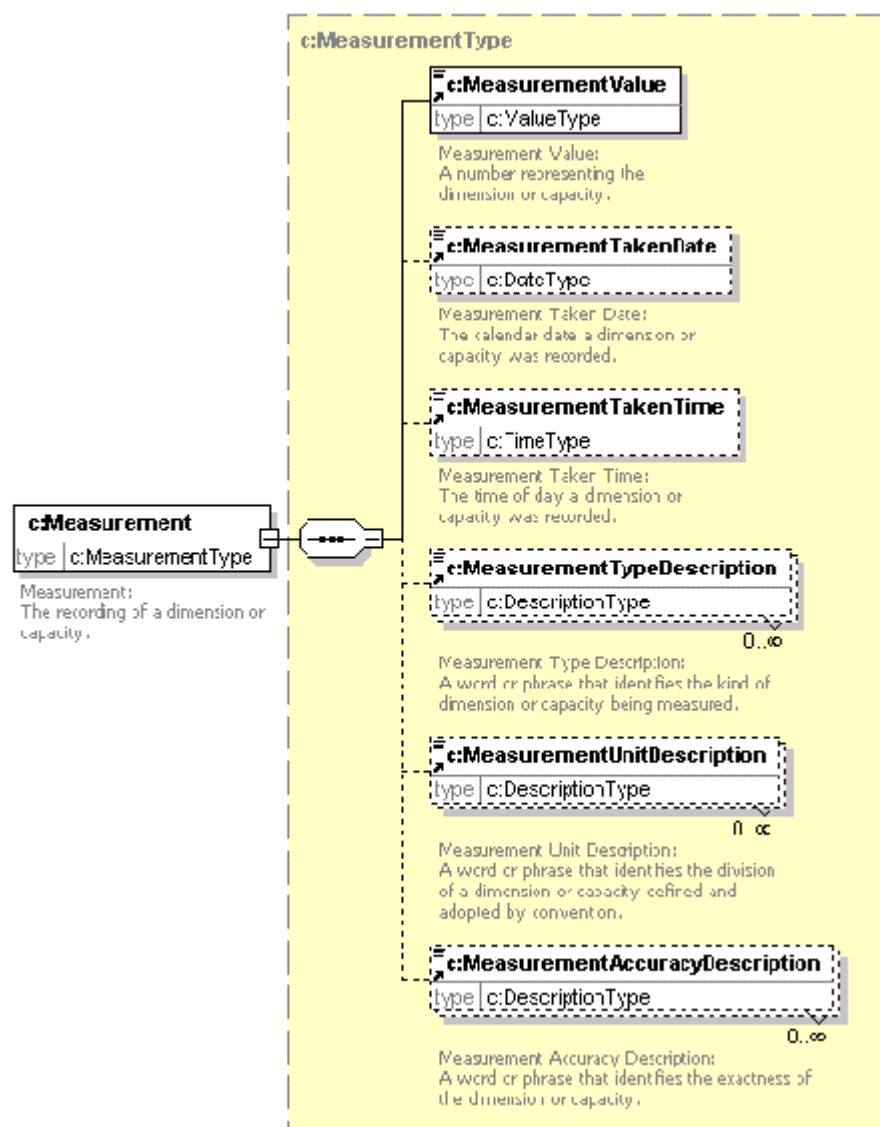
Example: <http://tools.dss-snd.gc.ca/cpsin/ns/gl/1-2-3/sx/case.xsd>

The examples that follow are French and English annotated schemas in the XML Spy design view.

# French Annotated Schema



# English Annotated Schema



# CPSIN Transaction Schemas

## CPSIN Community Transactions

One CPSIN Transaction Schema is currently available. This is the Common Offence Library Download Schema which is included in the CPSIN XML Download and carries the namespace prefix of COL1.

# Codes

The CPSIN Data Dictionary addresses coded data by the use of matching pairs of data elements. One defines the terse coded values while the other defines their language-specific plain text equivalents. Where the value set is maintained by CPSIN, the terse codes are language-independent numeric values and the data element has a representation term of Identifier. For codes maintained by other authorities the original coded values are respected and the elements have a representation term of Code, e.g. ISO Country Code, Country Subdivision Code, and Language Code and NAICS Code. The data element for their plain text equivalents has a representation term of Description, Name or Abbreviation. For instance, the CPSIN XML Global Schema contains:

Identifying elements	Descriptive elements
sexId	SexDescription
streetTypeId	StreetTypeAbbreviation
CountryCode	CountryShortName

Note that the language-independent numeric value identifiers are implemented as XML attributes. Hence the lower camel case.

The CPSIN XML Global Schema supports the exchange of coded data in two ways:

- Coded - “Just-in-Time Multilingualism”
- Decoded - Language-specific plain text

## Coded Method

This is the primary, recommended method and the only method supported by the CPSIN Browser. It is the most efficient method in terms of bandwidth consumption and application processing for database-to-database exchange or any scenario where output must be presented based on a language preference controlled by the end-user. Hence the term “Just-in-Time Multilingualism”. In addition to operational concerns, it also offers more fundamental benefits to the CPSIN community:

- All partners share official, standard translations maintained by CPSIN.
- Prevents the erosion of data quality and bilingualism through the use of unofficial free-form text values
- Tried and true technique that has served the Federal IT community well in the development of table-driven bilingual systems

In this approach:

- CPSIN-administered language-independent numeric code values are exchanged in XML attributes.
- Code values administered by other authorities are exchanged in XML elements.
- CPSIN and other authority codes are decoded using locally-held tables which provide French or English plain texts and are available as XML files in the CPSIN download.
- Decoding can be performed by XSLT templates provided with the CPSIN Browser or by partner applications.

Sample of coded data:

```
<c:Person c:sexId="1" c:transgenderId="3">  
  <c:ObjectFullName />  
</c:Person>
```

## Decoded Method

This is not the standard, recommended method and it is not supported by the CPSIN Browser. It is supported by the CPSIN XML Global Schema to accommodate scenarios where secure encryption, audit trail, and court-strength non-repudiation concerns dictate that exchanged information cannot be dependent on after-the-fact table lookups. For instance, encrypted eForm integrity may require knowing with absolute certainty what users actually saw when they provided an authorizing signature without tampering with the envelope. This may require the exchange of plain text code values in one or both official languages.

In this approach:

- Language-specific, plain text values are exchanged in XML elements, which are permitted multiple occurrences to accommodate more than one language.
- Each XML element instance must specify or explicitly inherit the language used by means of the `xml:lang` attribute.
- Plain text values can be validated, translated or encoded by partner applications using locally-held tables that contain the French or English texts with their terse equivalents and are available as XML files in the CPSIN download.

Sample of decoded data:

```
<c:Person xml:lang="fr-CA">
  <c:SexDescription>femelle</c:SexDescription>
  <c:TransgenderDescription xml:lang="en-CA">transvestite
</c:TransgenderDescription>
  <c:ObjectFullName />
</c:Person>
```

# Multilingualism

The language used in all instances of CPSIN XML elements that contain language-specific, translatable text must be identifiable to parsing software. Therefore, all XML elements of type Abbreviation, Description, Name, Narrative, Text and Title must either specify or explicitly inherit a value of the `xml:lang` attribute from any parent element. Consequently, all CPSIN data exchange documents must carry at least one instance of the `xml:lang` attribute, typically in the root element.

Use of the `xml:lang` attribute is recommended by internationalization/localization standards (visit <http://www.opentag.com/xmli18nfaq.htm>). It will be increasingly used for Web browsers, audio page readers, translation software, spell checkers, etc., and public safety interoperability initiatives will become increasingly multilingual. It is also essential to CPSIN support for coded data. Text generated by CPSIN partners should use the values:

- en-CA (Canadian English)
- fr-CA (Canadian French)

Future values might include, among others:

- en-GB (British English)
- en-US (American English)
- es-MX (Spanish)
- iu (Inuktitut)

In this example, the first `CaseNoteNarrative` explicitly inherits the value of `en-CA` from a parent element, in this case the root element, `Person`. In the second `CaseNoteNarrative`, this is overridden by specifying a value of `fr-CA`.

```
<c:Person xml:lang="en-CA">
  <c:CaseNote>
    <c:CaseNoteNarrative>
      In my opinion, this person ...
    </c:CaseNoteNarrative>
  </c:CaseNote>
  <c:CaseNote>
    <c:CaseNoteNarrative xml:lang="fr-CA">
      À mon avis, cette personne ...
    </c:CaseNoteNarrative>
  </c:CaseNote>
</c:Person>
```

# CPSIN XML Viewer

The CPSIN XML Viewer gives XML transaction developers an enhanced view of raw XML data in a Web browser (Microsoft Internet Explorer 5.5+ only) by providing mouse-controlled:

- CPSIN data element name and definition hints
- translation of CPSIN code values in both official languages
- expandable/collapsible tag outlining.

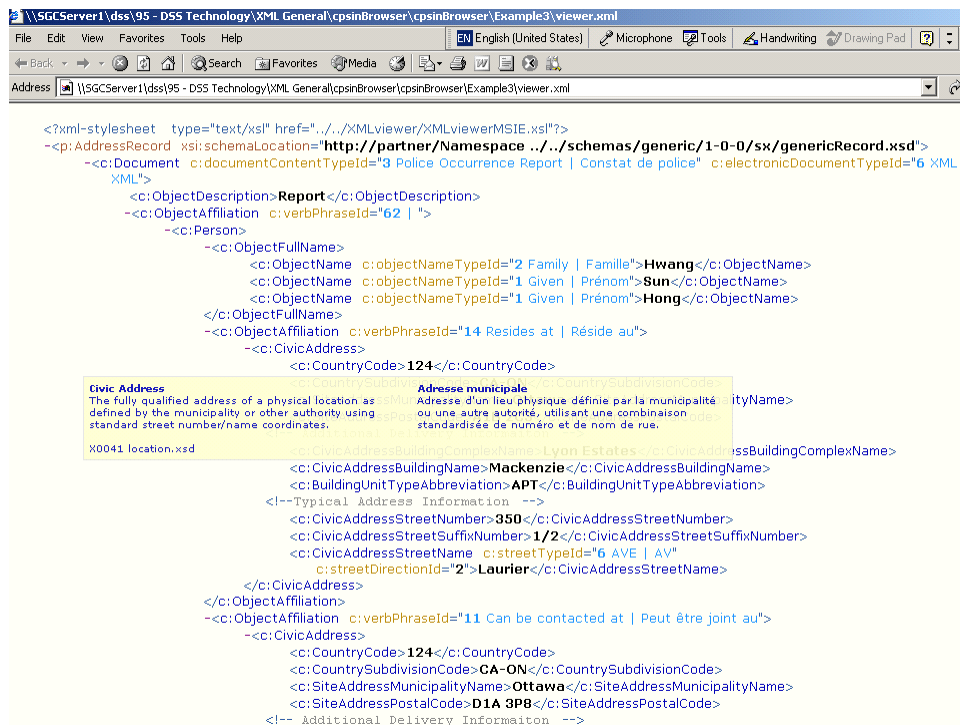
It aims to enhance productivity in the design, development and debugging of CPSIN XML transactions. Written in XSLT, it is invoked by inserting a processing instruction in the XML file for MSIE, such as:

```
<?xml-stylesheet type="text/xsl" href="../../../schemas/cpsin/gl/2-0-0/viewer/XMLviewerMSIE.xsl"?>
```

It is included in the CPSIN Download and example files can be found in the folder:

cpsindownload\data\cpsintools

A typical screen shot:



# CPSIN Browser

The CPSIN Browser styles CPSIN XML for direct human consumption in any modern Web browser. This allows partner staff to view any fully-compliant CPSIN XML file as an interactive, bilingual Web page, regardless of local technology investments. It features:

- an easy-to-read generic format for CPSIN grammar constructs
- expandable/collapsible outlining
- official language switching
- cross-browser and cross-platform deployment
- additional support for inline base64 binary xml images in Mozilla Firefox.

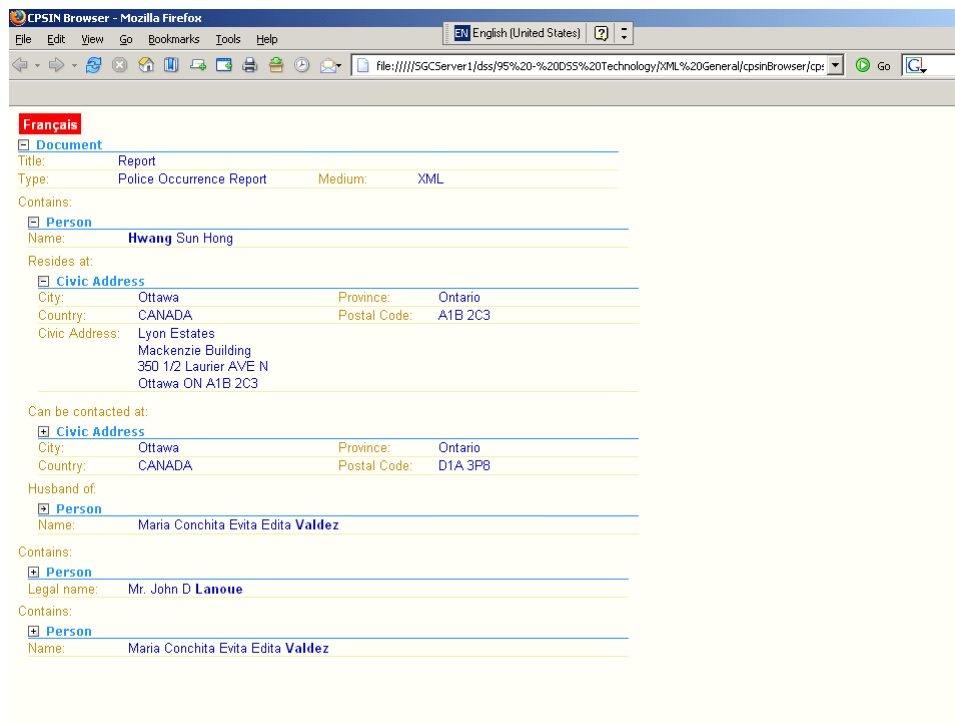
Written in XSLT, it is invoked by inserting a processing instruction in the XML file for the Web browser, such as:

```
<?xml-stylesheet type="text/xsl"
href="../../../schemas/cpsin/gl/2-0-0/browser/browser.xsl" ?>
```

It is included in the CPSIN Download and example files can be found in the folder:

```
cpsindownload\data\cpsintools
```

A typical English and French screen shot:



[English](#)

[Document](#)

Titre: Report  
 Type: Constat de police      Format: XML

Contient:

[Personne](#)  
 Nom: **Hwang Sun Hong**

Réside au:

[Adresse municipale](#)  
 Ville: Ottawa      Province: Ontario  
 Pays: CANADA      Code postal: A1B 2C3  
 Adresse municipale: Lyon Estates  
 Mackenzie Building  
 350 1/2 Laurier AV N  
 Ottawa ON A1B 2C3

Peut être joint au:

[Adresse municipale](#)  
 Ville: Ottawa      Province: Ontario  
 Pays: CANADA      Code postal: D1A 3P8

Mari (conjoint, époux) de:

[Personne](#)  
 Nom: Maria Conchita Evíta Edita **Valdez**

Contient:

[Personne](#)  
 Nom légal: M. John D **Lanoué**

Contient:

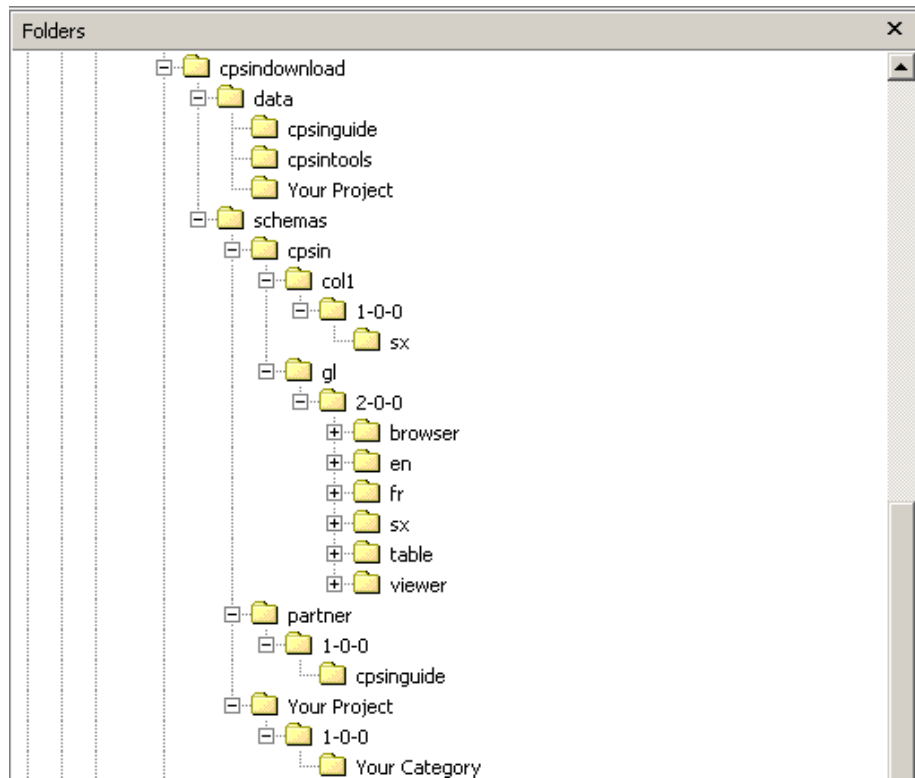
[Personne](#)  
 Nom: Maria Conchita Evíta Edita **Valdez**

# CPSIN XML Download

The CPSIN XML Download provides, in a compressed format, the current version of:

- CPSIN XML Global Schema and release notes
- CPSIN XML Transaction Schemas
- CPSIN XML Guide with example files and extension schemas
- CPSIN XML Viewer and Browser with example files

It is organized in the recommended folder structure for partner projects to facilitate version management and the use of CPSIN schemas and supporting tools. Please note that if a project modifies the folder structure, changes will be required in the paths specified in schema and style sheet references. However, just renaming the cpsindownload folder locally will have no impact.



After unzipping the download and studying the Using CPSIN XML section of this guide, the best way to get started is to:

1. Try out the CPSIN Browser and XML Viewer by opening files in data\cpsintools with Internet Explorer.
2. Review the Guide example files in data\cpsinguide with Internet Explorer and, if necessary, the example schema extensions in schemas\partner\1-0-0\cpsinguide.
3. Use files from these folders as starter templates and copy them into data\Your Project or schemas\Your project\1-0-0\Your Category.
4. Rename the Your Project folders appropriately and start editing in your favourite XML toolset (depending on Notepad is not recommended).

# Using CPSIN XML

The XML instance files used in the following examples are included in the CPSIN Download folder:

```
cpsindownload\data\cpsinguide
```

and can be reviewed in the CPSIN XML Viewer by opening them in Microsoft Internet Explorer.

The partner extension schemas used in these examples can be found in the folder:

```
cpsindownload\schemas\partner\1-0-0\cpsinguide
```

and are best viewed in graphical IDEs such as oXygen, Stylus Studio or XML Spy, but can also be opened in a Web browser.

Throughout the following examples, note that:

- `c:` is used as the CPSIN namespace prefix. This is recommended for brevity and consistency, but also because it is required in all contexts by the CPSIN XML Viewer and Browser.
- CPSIN XML attributes require the `c:` namespace prefix because they are globally defined.
- `p:` is used as an example of a CPSIN partner namespace prefix.
- `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` is required in instance files to permit the use of the `xml:lang` attribute. However, the value of the related `xsi:schemaLocation` attribute is always the target CPSIN or partner schema.
- `xs:include` is used to refer to other schema files in the `targetNamespace`, i.e. modules of the same schema.
- `xs:import` is used to refer to schema files from another namespace, i.e. a different schema.
- In these examples, various CPSIN schema modules are imported depending on the context. When in doubt, import `case.xsd`. It may imply some unnecessary overhead but the entire Global schema content is accessible from this starting point.
- The partner schema examples follow the CPSIN XML Schema Design Standard presented at the end of this document and use the CPSIN named simple types, referred to as domains in the CPSIN Data Dictionary. This is strongly recommended to maximize data standardization and exchangeability.

# Creating a CPSIN XML Document

CPSIN XML documents are XML instance documents whose root tag is an XML element defined in a CPSIN schema.

## Example 1: A single CPSIN Object

The root element of this XML instance file is `<Person>`, which is a member of the substitution group whose head element is `<Object>`. Any other member could be used, such as `<Document>` or `<Vehicle>`. Only `<Object>` itself cannot be used because it is abstract and therefore cannot be instantiated.

```
<c:Person xmlns="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
  xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0
    ../../schemas/cpsin/gl/2-0-0/sx/case.xsd"
  c:sexId="1" xml:lang="en-CA">
  <c:ObjectFullName c:objectNameCategoryId="4">
    <c:ObjectName c:objectNameTypeId="2">Hwang</c:ObjectName>
    <c:ObjectName c:objectNameTypeId="1">Sun</c:ObjectName>
    <c:ObjectName c:objectNameTypeId="1">Hong</c:ObjectName>
  </c:ObjectFullName>
  <c:BeingBirthYear>1983</c:BeingBirthYear>
  <c:BeingBirthMonth>12</c:BeingBirthMonth>
  <c:BeingBirthDay>07</c:BeingBirthDay>
</c:Person>
```

Similarly, the root element could be a member of the `<CaseComponent>` substitution group, such as `<Case>` or `<Event>`, but not `<CaseComponent>` itself.

## Example 2: A CPSIN Object collection

The root element of this XML instance file is `<Objects>`, which is a container element that allows the file to contain a list of Objects of the same or different kinds. In this case it contains multiple `<Person>` elements.

```
<c:Objects xmlns="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
  xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0
    ../../schemas/cpsin/gl/2-0-0/sx/case.xsd" xml:lang="en-CA">
  <c:Person c:sexId="1">
    <c:ObjectFullName c:objectNameCategoryId="4">
      <c:ObjectName c:objectNameTypeId="2">Hwang</c:ObjectName>
      <c:ObjectName c:objectNameTypeId="1">Sun</c:ObjectName>
      <c:ObjectName c:objectNameTypeId="1">Hong</c:ObjectName>
    </c:ObjectFullName>
    <c:BeingBirthYear>1983</c:BeingBirthYear>
    <c:BeingBirthMonth>12</c:BeingBirthMonth>
    <c:BeingBirthDay>07</c:BeingBirthDay>
  </c:Person>
  <c:Person c:sexId="2" />
  <c:Person c:sexId="3" />
</c:Objects>
```

Similarly, the root element could be `<Cases>`, which is a container for a list of Case Components.

## Extending CPSIN Schemas

Some partners may wish to reuse standard CPSIN data structures, but with minor additions or modifications which are defined in partner namespaces. This is most likely to arise with Object subtypes and there may even be a need to expand the CPSIN Object taxonomy to another level in some area. Customizing an existing Object subtype allows continued use of all its existing features, like Object Roles and Object Affiliations. Such extensions to the standard may be temporary while new content goes through the CPSIN approval process, or remain permanently outside the official CPSIN scope. In any event, the DSS team should be consulted before any form of extension is considered to ensure that existing features are being fully exploited and to make the team aware of additional needs.

## Example 3: Extending a CPSIN Object Subtype

Suppose certain corrections and parole partners wish to add a field called `TravelDuration` to an inmate's home address to record how many hours it takes to travel from the inmate's correctional facility to that address. This information is needed for scheduling absences and passes.

```
<p:CivicAddress
  xmlns:p="http://partner/namespace"
  xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://partner/namespace
  ../../schemas/partner/1-0-0/guide/extension1.xsd"
  xml:lang="en-CA">
  <c:CountryCode>124</c:CountryCode>
  <c:CountrySubdivisionCode>CA-ON</c:CountrySubdivisionCode>
  <c:SiteAddressMunicipalityName>Dire Straits
  </c:SiteAddressMunicipalityName>
  <c:SiteAddressPostalCode>X1Y 2Z3</c:SiteAddressPostalCode>
  <c:CivicAddressStreetNumber>1234</c:CivicAddressStreetNumber>
  <c:CivicAddressStreetName
    c:streetTypeId="115" c:streetDirectionId="8">Main
  </c:CivicAddressStreetName>
  <p:TravelDuration>6</p:TravelDuration>
</p:CivicAddress>
```

The namespace prefix `p:` distinguishes between the original CPSIN `CivicAddress` and the partner's extended version. The additional element, `TravelDuration`, appears after all the original ones.

Although it references a CPSIN namespace, this extension is entirely defined within a partner namespace. The partner schema for this is as follows:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:p="http://partner/namespace"
  xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
  targetNamespace="http://partner/namespace"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
    schemaLocation="../../../cpsin/gl/2-0-0/sx/location.xsd"/>
  <xs:complexType name="CivicAddressType">
    <xs:complexContent>
      <xs:extension base="c:CivicAddressType">
        <xs:sequence>
          <xs:element ref="p:TravelDuration"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="CivicAddress"
    substitutionGroup="c:Object" type="p:CivicAddressType"/>
  <xs:element name="TravelDuration" type="c:DurationType"/>
</xs:schema>
```

The CPSIN complex type called `CivicAddressType` is identified as the base for the partner's `CivicAddressType`, which is then extended by the addition of the `TravelDuration` element. The partner's `CivicAddress` element declares itself to be a member of the CPSIN Object substitution group so that it can be used in any situation where a CPSIN Object subtype would be valid.

The same technique could be applied to the customization of Case Components.

## Example 4: Extending the Object Taxonomy

Suppose certain partners discover an urgent need to capture information on UFO phenomena. They already use Vehicle and perhaps existing subtypes of Vehicle but feel that UFO deserves its own subtype because of its additional, unique data requirements. Other partners remained sceptical but were prepared to approve new values of "Ion Drive" and "Anti-matter" for Vehicle Propulsion Type.

```
<p:UFO
  xmlns:p="http://partner/namespace"
  xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://partner/namespace
  ../../schemas/partner/1-0-0/guide/extension2.xsd"
  c:vehiclePropulsionTypeId="99"
  p:originatingGalaxyQuadrantId="9">
  <c:ObjectDescription xml:lang="en-CA">Centre changes colour with no
organized pattern. Stops on a dime. No snow tires.
</c:ObjectDescription>
  <c:Measurement>
    <c:MeasurementValue
      c:measurementAccuracyId="1"
      c:measurementTypeId="3"
      c:measurementUnitId="1">15</c:MeasurementValue>
    <c:MeasurementTakenDate>2006-06-06</c:MeasurementTakenDate>
  </c:Measurement>
  <p:AlienCrewCount>4</p:AlienCrewCount>
  <p:AbducteeCount>2</p:AbducteeCount>
</p:UFO>
```

Notice the uniquely Canadian mixture of British and American English spelling in the content of ObjectDescription. A UFO can have an ObjectDescription and a Measurement (an estimated width of 15 metres) because it is an extension of a CPSIN Object and therefore everything in ObjectSuperType is available to it.

The UFO vehicle subtype needs two new XML elements (AlienCrewCount, and AbducteeCount ) and one new XML attribute (originatingGalaxyQuadrantId ), as defined in the following partner schema:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:p="http://partner/namespace"
xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
targetNamespace="http://partner/namespace"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
    schemaLocation="../../../../cpsin/gl/2-0-0/sx/property.xsd"/>
  <xs:complexType name="UFOType">
    <xs:complexContent>
      <xs:extension base="c:VehicleType">
        <xs:sequence>
          <xs:element ref="p:AlienCrewCount" />
          <xs:element ref="p:AbducteeCount" />
        </xs:sequence>
        <xs:attribute ref="p:originatingGalaxyQuadrantId" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="UFO"
    substitutionGroup="c:Object" type="p:UFOType" />
  <xs:element name="AlienCrewCount" type="c:CountType" />
  <xs:element name="AbducteeCount" type="c:CountType" />
  <xs:attribute name="originatingGalaxyQuadrantId" type="c:IdType" />
</xs:schema>
```

This is implemented using the same technique as in Example 3, except that the names of the extended complex type and element are now different from their CPSIN equivalents.

# Partner-CPSIN Hybrid Documents

Deviating from pure CPSIN grammar in XML transaction design carries the penalty of reduced interoperability benefits and more custom code. However, partner applications may need to wrap CPSIN content in their own elements or even just incorporate CPSIN content at certain points within documents of their own design.

## Example 5: Wrapping CPSIN Objects

Suppose a police college needs to develop a customized honour roll transaction to distribute a list of annual graduates. But rather than reinvent the wheel, they use person data elements defined in the CPSIN Object subtype `<Person>`.

```
<p:HonourRoll
  xmlns:p="http://partner/namespace"
  xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/g1/2-0-0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://partner/namespace
  ../../schemas/partner/1-0-0/guide/hybrid1.xsd">
  <p:GraduationYear>2004</p:GraduationYear>
  <c:Person xml:lang="en-CA">
    <c:ObjectFullName>
      <c:ObjectName c:objectNameTypeId="1">Mary</c:ObjectName>
      <c:ObjectName c:objectNameTypeId="2">Smith</c:ObjectName>
    </c:ObjectFullName>
  </c:Person>
  <c:Person />
  <c:Person />
  <c:Person />
</p:HonourRoll>
```

It is recommended that partner elements always be identified by a namespace prefix in instance documents for clarity and consistency. `p:` is used here to explicitly distinguish between the partner-defined elements and the CPSIN `Person`.

The partner schema for this document is as follows:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:p="http://partner/namespace"
xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
targetNamespace="http://partner/namespace"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:import namespace="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
schemaLocation="../../../cpsin/gl/2-0-0/sx/being.xsd"/>
  <xs:complexType name="HonourRollType">
    <xs:sequence>
      <xs:element ref="p:GraduationYear" minOccurs="1" />
      <xs:element ref="c:Person" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="HonourRoll" type="p:HonourRollType" />
  <xs:element name="GraduationYear" type="c:YearType" />
</xs:schema>
```

The same technique could be applied to the use of Case Components.

## Example 6: Element level reuse

This is the most granular and least interoperable type of CPSIN usage. However, it can offer useful standardization of specific elements or codes. Suppose some partners want to start using an information sharing authorization scheme that has not yet been formally adopted by the CPSIN community. The data structure implemented by these partners reuses CPSIN data elements by referencing individual XML elements and attributes taken from various parts of the CPSIN XML Global Schema, identified by the `c:` namespace prefix. This is sometimes referred to as cherry-picking.

```
<p:SharingAuthorization
  xmlns:p="http://partner/namespace"
  xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://partner/namespace
  ../../schemas/partner/1-0-0/guide/hybrid2.xsd"
  c:jurisdictionId="1"
  c:securityLevelId="5"
  p:collectionReasonId="1"
  p:dispositionAuthorityId="1">
  <p:SharingAuthorityCode>44</p:SharingAuthorityCode>
  <p:SharingAuthorityCode>23</p:SharingAuthorityCode>
  <c:StatuteAbbreviation xml:lang="en-CA">CDSA
  </c:StatuteAbbreviation>
  <c:CountryCode>124</c:CountryCode>
</p:SharingAuthorization>
```

This instance file validates against the following partner XML schema.

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:p="http://partner/namespace"
xmlns:c="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
targetNamespace="http://partner/namespace"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:import namespace="http://tools.dss-snd.gc.ca/cpsin/ns/gl/2-0-0"
schemaLocation="../../../cpsin/gl/2-0-0/sx/case.xsd"/>
  <xs:complexType name="SharingAuthorizationType">
    <xs:sequence>
      <xs:element ref="p:SharingAuthorityCode"
maxOccurs="unbounded" />
      <xs:element ref="c:StatuteAbbreviation" />
      <xs:element ref="c:CountryCode" minOccurs="1" />
    </xs:sequence>
    <xs:attribute ref="c:jurisdictionId" use="required" />
    <xs:attribute ref="c:securityLevelId" use="required" />
    <xs:attribute ref="p:collectionReasonId" use="required" />
    <xs:attribute ref="p:dispositionAuthorityId" use="required" />
  </xs:complexType>
  <xs:element name="SharingAuthorization"
type="p:SharingAuthorizationType" />
  <xs:element name="SharingAuthorityCode" type="c:CodeType" />
  <xs:attribute name="collectionReasonId" type="c:IdType" />
  <xs:attribute name="dispositionAuthorityId" type="c:IdType" />
</xs:schema>

```

# CPSIN XML Schema Design Standard

CPSIN XML Schema Design Standard incorporates the best practices and conventions of the XML world in general and the field of standard XML vocabularies in particular. They are intended to promote the productivity of experienced XML practitioners tasked with implementing the standard and to encourage the widest possible reuse of common components at all levels of granularity. The most influential sources have been:

- Draft Federal XML Developer's Guide
- [http://xml.gov/documents/in\\_progress/developersguide.pdf](http://xml.gov/documents/in_progress/developersguide.pdf)
- OAGIS 8.0 Design Document and User's Guide for extending OAGIS 8.0
- <http://www.openapplications.org/downloads/memberdocuments.htm>
- Structure and Design Issues for Developing, Implementing, and Maintaining a Justice XML Data Dictionary
- <http://www.it.ojp.gov/initiatives/files/JusticeXMLStructureTaskForceReport.doc>

This section is primarily aimed at DSS staff with XML schema design and maintenance responsibilities. However, it will also help CPSIN partner staff in reading the schemas and may be of value to them in formulating XML schema standards and guidelines for their own environments. Greater consistency across environments means improved productivity and easier transfer of technology among staff in an area where skilled resources are limited.

The goal of this standard is to ensure that the design of CPSIN XML schemas consistently promotes flexible reuse and extension at all levels of granularity. If CPSIN partner organizations also wish to pursue XML standardization through reuse and extension within their own application portfolios, then this schema design style is strongly recommended.

## CPSIN XML Schema Design Rules

These are the detailed rules for the development and enhancement of all CPSIN XML schemas to ensure continuing consistency.

1. Every CPSIN XML element and XML attribute shall be globally defined.
2. In the Global Schema, every CPSIN complex and simple type shall be globally defined. No anonymous types are permitted.
3. In transaction schemas, elements may anonymously restrict the types defined for them in the Global schema.
4. Every type used to represent a CPSIN data element shall be based on the types found in `common.xsd` which provide the XML implementation of the logical domains defined in the CPSIN Data Dictionary. This ensures data type standardization and global maintainability.
5. The language used in all instances of CPSIN XML elements that contain language-specific, translatable text must be identifiable to parsing software. This includes all elements of type Abbreviation, Description, Name, Narrative, Text and Title. Therefore their definition must include `xml:lang` as an optional attribute. In order to permit explicit inheritance of a language value, the elements `Objects`, `Object`, `Cases`,

CaseComponent, and any others with potential for use as document root elements, must also include `xml:lang` as an optional attribute. For data generated by CPSIN partners, `xml:lang` valid values are `en-CA` (Canadian English) and `fr-CA` (Canadian French). In order to support `xml:lang` usage throughout the Global Schema, the `case.xsd` and `common.xsd` modules import the following W3C namespace:

```
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
```

6. As the Global schema is maintained, it must continue to accommodate both the coded and decoded methods of supporting codes.
7. Base simple types, i.e. those ending in `...BaseType`, only serve to specify a maximum string length restriction for the CPSIN Data Dictionary textual domains they represent. These base types are then extended by a similarly named complex types that require the presence of an `xml:lang` attribute. For example, `NarrativeType` extends `NarrativeBaseType`. The two-step technique is required because XML Schema 1.0 does not permit a single type to specify both restrictions and extensions.
8. CPSIN-managed codes shall be expressed as language-independent numeric values represented in XML attributes in the `codes.xsd` schema module. Their language-specific text equivalents will be expressed in XML elements maintained in the `codetext.xsd` schema module. Value enumerations, which could number in the thousands for some codes, will not be maintained in the schema to avoid performance penalties. Lookup tables in both official languages will be maintained and provided for partner download.
9. XML attributes shall only be used for data which is solely intended for programmatic access or manipulation, and not presentation. In CPSIN they have two applications:
  - a. Unique identifiers for data objects, used as cross-references and/or database keys, i.e. "dumb numbers" or "dumb tokens". Compared to sub-elements, the use of attributes for this purpose results in significantly simpler and more robust parsing, validation and object instantiation code.
  - b. CPSIN-managed language-neutral numeric code values, exclusively used for language-specific table lookup. Compared to sub-elements, the use of attributes for this purpose consumes typically half the bandwidth and, more importantly, yields similar memory consumption and parsing time savings.
10. The following rules of thumb must be observed if additional uses of XML attributes are under consideration:
  - a. If a data value may be presented to an end user "as is", then it should be the content of an XML element. This will avoid problems arising from the limitations of presentation technologies.
  - b. XML attributes should not be used for potentially non-atomic data values, since only XML elements will allow extending partners to break down the data value into a sub-element structure. An attribute value would have to be broken down by programmatic parsing. For instance, a Vehicle Identification Number or a NAICS code contain meaningful components.
  - c. XML attributes should not be used where data order is significant. If an XML element has multiple attributes, XML Schema parsers are not required to validate the order in which they appear in instance documents.
11. Attributes should be placed in the element that they actually qualify, not at a higher level of abstraction, in order to support granular reuse cleanly, e.g. `streetDirectionId` and `streetTypeId` qualify `CivicAddressStreetName`, not `CivicAddress`.
12. XML elements shall not be declared nillable.
13. XML elements used to represent CPSIN Data Model data entities must not have mixed content, i.e. a combination of both text content and sub-elements. Mixed content is only of value in representing the internal structure of textual documents.
14. The translation of the CPSIN Data Model into XML schema structures should observe the following principles. Fundamental data entities are represented as an XML element with a complex type to define its content model where the entity's attributes are typically sub-elements. Subordinate entities, dependent, associative, or attributive, will also be

- included in its content model, typically as multiply occurring sub-elements with their own complex types to define their entity attributes as sub-elements.
15. Where there is a need to resolve many-to-many relationships, the use of `key/keyref` is recommended over `ID/IDREF`, since it is much more flexible.
  16. The CPSIN Data Model Object taxonomy shall be implemented in the Global Schema exclusively by the technique of establishing a hierarchy of named complex types derived by extension whose root is `ObjectSuperType`. These complex types shall be referenced by corresponding XML elements named after the Object subtypes which are also members of a substitution group whose head element is `Object`.
  17. The `Object` element shall be abstract. By default, any Object subtype XML element may be instantiated, not just leaf nodes. The decision to make an `Object` subtype element abstract, in order to force the selection of a lower level node, should be on a case-by-case basis, depending on the evolving data exchange requirements of the CPSIN community. Bear in mind that over time the taxonomy will grow and leaves may become branches.
  18. The CPSIN Data Model Case Topic entities shall be implemented in the Global Schema exclusively by the technique of defining them as members of a substitution group whose head element is the abstract `CaseComponent` so that they inherit the properties of `CaseComponentType` that support CPSIN XML grammar.
  19. The CPSIN Global Schema content models should be relaxed and not attempt to specify validation constraints beyond the most fundamentally obvious cardinality restrictions for the data structures. CPSIN and partner XML transactions will require specific business context validation, such as co-occurrence constraints. The facilities of XML Schema may not always be adequate or optimal for these purposes. Consideration should be given to validation options such as Schematron and application program logic.
  20. The CPSIN Global Schema should be maintained in a modular file structure reflecting partitioning by subject matter and function in order to minimize the need to include or import unnecessary content in operational situations and to localize the impact of content updates. The effectiveness of these measures should be assessed periodically. Any modifications should be implemented in a new major version.
  21. To cross-reference CPSIN Data Dictionary data elements to their XML implementation and to permit the comprehensive, multilingual documentation of XML schema content, all CPSIN schema elements shall carry an `id` attribute containing a one letter prefix followed by a 4-digit number from one of 4 separate ranges. This scheme is documented in the CPSIN XML Schema Element Identifier Convention.

## CPSIN XML Schema Naming Conventions

1. Upper camel case (UCC) shall be used to name XML elements, complex and simple types, keys and keyrefs, e.g. `PropertyStatusDescription`
2. Lower camel case (LCC) shall be used to name XML attributes, e.g. `propertyStatusId`
3. XML elements and XML attributes representing CPSIN Data Dictionary data elements shall bear their full CPSIN Data Dictionary names, i.e. the object class term of these ISO 11179-compliant names is not dropped in any context, in order to permit unambiguous granular reuse, e.g. `CivicAddressBuildingName`.
4. XML elements representing CPSIN Logical Data Model data entities shall bear their full CPSIN Data Dictionary names, e.g. `BodyPartDeformity`
5. Complex types shall bear the name of the XML element whose content model they define, with the term “Type” appended, e.g. `CivicAddressType`.
6. The types in `common.xsd` shall bear the name of the CPSIN Data Dictionary domain they represent with the leading lower case “d” removed and the term “Type” appended, e.g. `dNarrative` becomes `NarrativeType`
7. Established acronyms that are widely understood in the CPSIN community are permitted, e.g. UCR, and will be explicitly documented in the Approved Abbreviation List.
8. The use of approved abbreviations is mandatory in all contexts.

## CPSIN XML Schema Approved Abbreviation List

The use of approved abbreviations is mandatory in all contexts.

Term	Approved Abbreviation
Identifier	Id
Uniform Crime Reporting	UCR

## CPSIN XML Schema Element Id Convention

To cross-reference CPSIN Data Dictionary data elements to their XML implementation and to permit the comprehensive, multilingual documentation of XML schema content, all CPSIN schema elements shall carry an `id` attribute containing a one letter prefix followed by a 4-digit number from one of 4 separate ranges.

### CPSIN Data Dictionary Data Elements

XML elements and attributes that implement a CPSIN Data Dictionary data element carry its Element Identifier as published in the French and English dictionaries, plus a prefix of E.

Artifact	Example
Dictionary Data Element	0354
XML Element or Attribute	E0354

### CPSIN Data Model Data Entities

A separate series of numbers is maintained for XML elements implementing data entities from the CPSIN Data Model, published in the dictionary as an Entity-Relationship diagram, plus any supporting data objects for the XML implementation. The elements have an X prefix and their matching types have a Y prefix.

Artifact	Example
XML Element	X0086
XML Type	Y0086

### CPSIN Data Dictionary Domains

The types in `common.xsd` define the standard, default data types used throughout the schema and are the XML implementation of the domains published in the CPSIN Data Dictionary, e.g. `dNarrative` becomes `NarrativeType`. A separate series of numbers is maintained for these types with a D prefix.

Artifact	Example
XML Type	D0017

### Miscellaneous XML Artifacts

A separate series of numbers is maintained for miscellaneous XML elements worthy of documentation, like identity constraints such as `key` or `keyref`, and intermediate types which are the by-products of multi-step restriction/extension derivations. They are given a Z prefix.

Artifact	Example
Miscellaneous XML element	Z0006