NAACCR XML Data Exchange Standard

Implementation Guide

Based on Specifications v1.4 of the NAACCR XML Standard

Last updated on
April 2019

Sponsoring Organizations
Canadian Partnership Against Cancer
Centers for Disease Control and Prevention
College of American Pathologists
National Cancer Institute
National Cancer Registrars Association
Public Health Agency of Canada

Sponsors with Distinction
American Cancer Society
American College of Surgeons
American Joint Committee on Cancer

Table of Contents

1 Introduction

The NAACCR "fixed-width" format (sometimes called the NAACCR flat file format) has been around for many years and has been widely accepted by the NAACCR community. The simplicity of the format makes it easy to implement in different types of software, but it also means it has some limitations. For example, it lacks the ability to handle large amounts of text. It is difficult for organizations to define their own data items and ensure that they are used in a consistent way. And finally, it is difficult to add new data items and retire existing ones when their position and length in the file affects the position of every other data item.

In 2014, a NAACCR XML Task Force was created to define and demonstrate a custom, NAACCR-defined XML data exchange standard for the NAACCR community. The NAACCR Board approved the initial version (Specifications v1.0) of the standard in 2015.

One of the primary design goals of that initial version was to stay fully compatible with the corresponding fixed-width format, providing an easy way to convert from XML to fixed-width and back again. As the standard evolves, more features will be added to take advantage of the XML structure, eventually breaking compatibility with the fixed-width standard. However, in the short-term, the NAACCR community and its many software tools and applications must be given time to adapt to the new XML standard during a transitional period where both XML and the fixed-width standard can co-exist.

The NAACCR Board approved an update to the standard (Specifications v1.1) in 2016 and another one (Specifications v1.2) in 2017 which introduced minor changes to the specifications.

For more information about the NAACCR XML standard or to download related documentation and supplementary files, visit:
http://www.naaccr.org/xml-data-exchange-standard/.

2 Intended Audience

The intended audience for this document is anyone wanting to have a better understanding of the NAACCR XML data exchange standard, anyone using a provided NAACCR XML library in their own software, or anyone implementing their own software libraries to read and write NAACCR XML.

3 Changes to the Specifications

3.1 Changes introduced in version 1.1

- A new "allowUnlimitedText" attribute was introduced in the dictionary to allow some text items to not be bound by their length in the XML data files.
- Items defined in user-defined dictionaries do not have to provide a start column anymore.

- User-defined dictionaries do not have to provide a version anymore; if missing, the version will be assumed to be the same as the provided base dictionary.
- A new "specificationVersion" attribute was added to the dictionaries and data files, it defaults to 1.0 if not provided.

3.2 Changes introduced in version 1.2

- The "userDictionaryUri" appearing in the data file can now provide more than one dictionary URI by separating them with a space.
- The data type of several data items was relaxed.
- The "regexValidation" attribute was deprecated and removed from the dictionaries.
- A new "GroupedItemDefs" section was added to the base dictionaries, it defines the "GroupedItemDef" for every grouped item that NAACCR supports. Note that grouped items cannot appear in data files.

3.3 Changes introduced in version 1.3

- The range restriction (9500-99999) imposed on non-standard items defined in user dictionaries has been removed. Non-standard items can use any number as long as
    - The number is not already defined in the corresponding base dictionary.
    - The number is not already defined in another user dictionary when several dictionaries are provided for a given data file.

3.4 Changes introduced in version 1.4

- The maximum length for NAACCR IDs was changed from 50 to 32 characters. As a result, several NAACCR 18 IDs had to be renamed to follow this new requirement (the full list is provided in Appendix A). For software that create XML data files, it is recommended to start writing the new IDs as soon as possible. For software that consume XML data files, one of the following approaches can be taken:
    - Switch the software to only accept the new IDs and reject any data files that still use the old IDs (and if possible, contact the organization who created the file and request them to provide it again with the new IDs).
    - Switch the software to accept both IDs for a certain period of time (until it is clear that no organization will create data files with the old IDs).

4 Standard Overview

The NAACCR XML standard is based on XML 1.0 (https://www.w3.org/TR/REC-xml/) and defines two sets of specifications:

1. **Dictionary specifications**: defines NAACCR Volume II data items, their mapping between XML and the fixed-width format, and their location in the NAACCR XML hierarchy.

2. **Data Exchange specifications**: defines the overall structure of a NAACCR XML data exchange document and some basic syntax conventions.

Line terminations are optional in a NAACCR XML data exchange file, but since the data files and dictionaries might be opened in text editors, it is recommended to use them along with proper indentation. New lines can be indicated with any combination of carriage-return and line-feed characters.

The standard does not enforce a specific character encoding, but recommends using the default for XML, UTF-8, which can be optionally specified in the XML header:

```
<?xml version="1.0" encoding="UTF-8"?>
```

or, alternately:

```
<?xml version="1.0"?>
```

Since version 1.0 is also the current default for XML, a file not defining a header is also valid and should be assumed to be XML version 1.0 with the UTF-8 character encoding.
Some cancer registry software systems and tools have not been written to properly handle characters outside the US-ASCII printable character range, so it is safest for text nodes in a NAACCR XML data exchange document to only contain ASCII characters in the decimal range 23 through 126.

All XML tags and attributes defined by this standard exist under the following namespace:

```
http://naaccr.org/naaccrxml
```

Dictionaries and data files should define that default namespace as a root attribute. More information about namespaces is provided in the sections related to root attributes.

4.1 Dictionary Specifications

NAACCR XML Dictionaries are XML files that define key metadata about NAACCR data items as they relate to a NAACCR XML data exchange file, their valid XML parent elements, and processing rules for dealing with text nodes containing coded values.

There are two types of dictionaries:

1. **Base dictionaries** define standard NAACCR items that are maintained by the NAACCR organization, defined in Volume II of the official NAACCR data standards.

2. **User dictionaries** define non-standard (state or organization-specific) items, they are maintained by the organizations defining them.

There is a “base” dictionary for each NAACCR version, maintained by the NAACCR organization. The XML standard also defines a default “user” dictionary for each version for reading and writing NAACCR XML data files when no other user dictionary is provided.

4.1.1 XML Elements

The XML structure for a given dictionary is defined by the following elements:

```
<NaaccrDictionary>
    <ItemDefs>
        <ItemDef/>
    </ItemDefs>
    <GroupedItemDefs>
        <GroupedItemDef/>
    </GroupedItemDefs>
</NaaccrDictionary>
```

The `<NaaccrDictionary>`, `<ItemDefs>` and `<GroupedItemDefs>` elements occur once per document, the `<ItemDef>` and `<GroupedItemDef>` elements are repeated for each data item definition.

4.1.1.1 NaaccrDictionary

The `<NaaccrDictionary>` root element allows the following attributes:

- **dictionaryUri (required)**: a unique string that defines the dictionary. It acts as an identifier and is referenced by the NAACCR XML data files. While typical values look like a website URL, they often do not represent actual websites that can be visited using a web browser. The base and default user-defined dictionaries maintained by the NAACCR organization use a strong naming convention on their ID:
    - Base dictionaries use the format `http://naaccr.org/naaccrxml/naaccr-dictionary-xxx.xml` where xxx is the corresponding NAACCR version.
    - Default user dictionaries use the format `http://naaccr.org/naaccrxml/user-defined-naaccr-dictionary-xxx.xml` where xxx is the corresponding NAACCR version.

  Using those conventions, a simple regular expression can be used to determine if a given dictionary is a base one or a user-defined one.

- **naaccrVersion (required for base dictionaries, optional for user dictionaries)**: NAACCR version that this dictionary defines. This field corresponds to the NAACCR Version data item 50 and allows the same values. Since version 1.1 of the NAACCR XML specification, this attribute is optional in user dictionaries; if not provided, it is assumed to be the same as the corresponding base dictionary.

- **specificationVersion (optional)**: the version of the specifications that the dictionary uses. If not provided, "1.0" is assumed.

- **description (optional)**: a short description of the dictionary.

In addition to those attributes, a dictionary should also define the default namespace: "http://naaccr.org/naaccrxml". Note that this value may not be a valid Internet address, it is simply a unique name for a specific XML namespace that defines all of the allowed elements and attributes. In summary, a dictionary must define the following attribute on the root XML element:

```
xmlns="http://naaccr.org/naaccrxml"
```

4.1.1.2 ItemDefs

The `<ItemDefs>` element acts as a container for repeated `<ItemDef>` elements and does not allow any attributes.

4.1.1.3 ItemDef

The base dictionary for a particular NAACCR version contains an `<ItemDef>` element for every NAACCR Volume II data item defined in that version, except for the following items:

- State/Requestor Items
- NPCR Specific Field
- Reserved items

Those are defined in the default user dictionary.

The `<ItemDef>` element allows the following attributes:

- **naaccrId (required)**: a unique value that identifies a data item. This value needs to be retained from one NAACCR version to the next, or the data item will not be recognized as being the same in both versions. The value should only contain letters and digits; it should start with a lowercase letter and use an uppercase letter to separate the words. The value must be 32 characters or less. Here are a few examples of valid values:
  - recordType
  - myItem
  - myOtherVariable2
  - old1970Variable

  The values for the base and default user dictionaries were derived from the NAACCR Name in Volume II with the following rules applied:
  1. Spaces, dashes, slashes, ampersands, periods and underscores are considered as word separators and replaced by a single space.
  2. Anything in parenthesis is removed (along with the parenthesis).
  3. Any non-digit and non-letter character is removed.
  4. The result is split by spaces (called words in the rest of this logic).
  5. Roman numeral words are converted to the corresponding numbers (so I becomes 1, II becomes 2, etc...); this applies only to full words, and only for numbers up to IX (9).
  6. If the two last words were converted roman numerals, a "to" word is inserted between them.

7. The first word is uncapitalized, the other words are capitalized. All abbreviations are considered words (so EOD becomes Eod).
8. All the words are concatenated back together.
9. The resulting ID is manually reviewed to ensure it is no more than 32 characters.

The NAACCR organization will maintain the NAACCR ID of all the standard items it defines. Organizations defining their own non-standard items are responsible for providing the NAACCR ID of those items in their user-defined dictionaries.

- **naaccrNum (required):** the NAACCR Number from the Volume II standard. User dictionaries can use numbers falling in any range as long as they follow these rules:
  - The number is not already defined in the corresponding base dictionary.
  - The number is not defined in another user dictionary when several dictionaries are provided for a given data file.

- **naaccrName (optional):** the NAACCR Name from the Volume II standard. Unlike IDs, Names can contain spaces, parentheses and other special characters, as long as they are escaped properly as an XML attribute value. Names must be 50 characters or less. The following table shows the characters that must be escaped as an XML attribute value:

Original Character	Escaped Value
<	<
>	>
"	"
&	&
'	'

- **startColumn (required for base dictionaries, optional for user dictionaries)**: the starting column where this data item will be read or written from a fixed-width file. If not provided, the data item is not expected to appear in a fixed-width data file.

- **length (required):** the maximum length of the value in a NAACCR XML data file. It is also used as the exact length in a fixed-width data file.

- **allowUnlimitedText (optional):** boolean indicating that the value of the data may be longer than the specified length attribute, breaking compatibility with the fixed-width standard. This attribute may be needed for data items that contain unstructured text that may not fit within the limits of a fixed-width record. The XML standard does not put a restriction on the actual length of text values in data files, therefore, applications may create data items that contain data outside the bounds of the length attribute with the understanding that some data consumers that are

expecting a fixed-width compatible XML file may truncate the data.

- **recordTypes (optional):** a comma-separated list of the record types from Volume II where this data item can appear (the value for an item defined in a NAACCR Incidence record would be "A,M,C,I" while an item defined only in a NAACCR Abstract/Modified record would be "A,M"). Defaults to "A,M,C,I".

- **sourceOfStandard (optional):** the source of standard for the item. This attribute is currently not used.

- **parentXmlElement (required):** expected parent tag in the XML data files; either "NaaccrData", "Patient", or "Tumor". This parent element is what defines the nested structure of a NAACCR XML data exchange document. The dictionary is the only place that defines this relationship and therefore the library needs to enforce the nesting rules when validating a NAACCR XML file.

- **datatype (optional)**: a name for the type of data contained in a data item, maps directly to a regular expression that can be used to validate the value. If not provided, "text" is assumed.

Data Type	Usage	Regular Expression
digits	Codes composed of digits only, which always occupy the full width of the field (n in the regular expression).	^\d{n}$
alpha	Codes composed of uppercase characters only, which always occupy the full width of the field (n in the regular expression).	^[A-Z]{n}$
mixed	Codes composed of digits and/or uppercase characters, which always occupy the full width of the field (n in the regular expression).	^[A-Z\d]{n}$
numeric	Variable length digits with an optional period.	^\d+(\.d+)?$
date	A NAACCR-style full or partial date (yyyy, yyyymm or yyyymmdd).	^(18|19|20)\d\d((0[1-9]|1[012])(0[1-9]|[12]\d|3[01])?)?$
text	Variable length text that can contain any characters, up to the specified with of the field (n in the regular expression).	^.{1,n}$

- **padding (optional):** a name for the text padding rules when writing a fixed-width file; can be set to “rightBlank”, “leftBlank”, “rightZero”, or “leftZero”; defaults to “rightBlank”.

- **trim (optinal)**: a name for the text trimming rules when converting a fixed-width file to XML; can be set to “none” or “all”, defaults to “all”.

4.1.1.4 GroupedItemDefs

The `<GroupedItemDefs>` element acts as a container for the repeated grouped item definitions. It does not allow any attribute. It can only appear in a base dictionary, which means organizations cannot define their own grouped items.

4.1.1.5 GroupedItemDef

Grouped item definitions follow the same rules as the regular `<ItemDef>` definitions, allowing the same attributes and having the same requirements. In addition, every grouped item definition must supply the following attribute:

- **contains (required)**: a comma-separated list of naaccrIds referencing the items making up the grouped item, in the order they are specified. The list should not contain any spaces (so “1,2,3” and not “1, 2, 3”) and must reference naaccrIds that exist in the current base dictionary.

4.1.2 XML Samples

Sample base dictionary:

```
<NaaccrDictionary
  dictionaryUri="http://naaccr.org/naaccrxml/naaccr-dictionary-160.xml"
  naaccrVersion="160"
  specificationVersion="1.2"
  description="NAACCR 16 base dictionary"
  xmlns="http://naaccr.org/naaccrxml">
    <ItemDefs>
        <ItemDef naaccrId="recordType"
            naaccrNum="10"
            naaccrName="Record Type"
            startColumn="1"
            length="1"
            recordTypes="A,M,C,I"
            parentXmlElement="NaaccrData"
        <ItemDef naaccrId="registryType"
            naaccrNum="30"
            naaccrName="Registry Type"
            startColumn="2"
            length="1"
            recordTypes="A,M,C,I"
            parentXmlElement="NaaccrData"
            dataType="digits"/>
        …
    </ItemDefs>
</NaaccrDictionary>
```

Sample user dictionary:

```
<NaaccrDictionary
  dictionaryUri="http://mycompany.org/mycompany-naaccr-dictionary.xml"
  specificationVersion="1.2"
  xmlns="http://naaccr.org/naaccrxml">
    <ItemDefs>
        <ItemDef naaccrId="myCompanyVariable1"
            naaccrNum="10001"
            length="1"
            parentXmlElement="NaaccrData"/>
        <ItemDef naaccrId="myCompanyVariable2"
            naaccrNum="10002"
            length="1"
            parentXmlElement="Patient"/>
        <ItemDef naaccrId="myCompanyVariable3"
            naaccrNum="10003"
            length="1"
            parentXmlElement="Tumor"/>
    </ItemDefs>
</NaaccrDictionary>
```

4. 2 Data Exchange Specifications

The NAACCR XML data files must define a base dictionary and may define one or several user dictionaries. They can only contain NAACCR data items that are defined in one of those dictionaries.

4.2.1 XML Elements

The structure of the data files is defined by the following elements:

```
<NaaccrData>
    <Item/></Item>
    <Patient>
        <Item></Item>
        <Tumor>
            <Item></Item>
        </Tumor>
    </Patient>
</NaaccrData>
```

The `<NaaccrData>` element occurs once per document, the `<Patient>` elements occur once per patient, and the `<Tumor>` elements occur once per tumor within a given patient. Each of those levels can contain any number of `<Item>` elements, defined by the dictionaries specified in the file.

Note that grouped data items from the current Volume II Appendix E are not supported as `<Item>` elements in data files, even though their `<GroupedItemDef>` elements are included in the base dictionary for the convenience of parsing logic. Since they overlap with

other Volume II data items, their data will be retained in a conversion from XML to a fixed-width file and back.

4.2.1.1 NaaccrData

The `<NaaccrData>` element is the root of the data files. It can have one or more `<Item>` children as long as they are defined in the base or user dictionary with a parentXmlElement attribute set to "NaaccrData". The `<NaaccrData>` element can also contain any number of `<Patient>` elements.

It allows the following attributes**:**

- **baseDictionaryUri (required)**: the base dictionary URI that defines the data items used in the data file.

- **userDictionaryUri (optional)**: the user dictionary URI that defines the data items used in the data file. If not provided, a default user dictionary will be used. Several URIs can be provided in this attribute, they must then be separated by a space.

- **recordType (required)**: the record types contained in the data file (A, M, C or I). Dictionaries will typically contain the definitions of all data items for any record type, but this attribute restricts which items can actually appear in the data exchange file.

- **specificationVersion (optional)**: the version of the specifications that the data file uses. If not provided, "1.0" is assumed.

- **timeGenerated (optional)**: the date the file was created (in ISO 8601 format).

In addition to those attributes, a data file should also define a default namespace of: "http://naaccr.org/naaccrxml". Note that this value may not be a valid Internet address, it is simply a unique name for a specific XML namespace that defines all of the allowed elements and attributes. In summary, a data file must define the following attribute on the root XML element:

```
xmlns="http://naaccr.org/naaccrxml"
```

As part of defining "extensions" (see corresponding section), other namespaces can be defined. For example, the following attributes can be added to define an "ext" tag prefix for a custom namespace, "http://my.company.org/naaccrxml":

```
xmlns:ext="http://my.company.org/naaccrxml"
```

This is an advanced feature and in most situations, only the default namespace needs to be specified.

Finally, the NAACCR XML standard allows for user-defined root attributes in their own custom namespaces. The standard makes no assumptions on what those are and what they mean. A library implementing the standard needs to allow all those user-defined attributes to be retrieved when a data file is parsed. As an example, the following attribute can appear on the root:

```
ext:internalFileId="XYZ-1"
```

4.2.1.2 Patient

A <NaaccrData> element can contain any number of <Patient> elements as children. Each <Patient> element can have one or more <Item> children as long as they are defined in the base or user dictionary with a parentXmlElement attribute set to "Patient". <Patient> elements can also contain any number of <Tumor> elements.

<Item> elements that are direct children of a <Patient> element apply to all <Tumor> elements underneath that <Patient>. For example, the social security number of a patient defined as <Item naaccrId="socialSecurityNumber"> will apply to every <Tumor> underneath that <Patient>. For a complete list of <Item> elements that are valid children of <Patient>, see the base dictionary description and file.

This element has no attributes.

4.2.1.3 Tumor

A <Patient> element can have any number of <Tumor> elements as children. Each <Tumor> element can have one or more <Item> children as long as they are defined in the base or user dictionary with a parentXmlElement attribute set to "Tumor". Each <Tumor> element corresponds to a separate tumor record for its parent <Patient> element. For a complete list of all <Item> elements that are valid children of <Tumor>, see the base dictionary description and file.

This element has no attributes.

4.2.1.4 Item

Items define the value for specific data items. The <Item> element specifies the following attributes:

- **naaccrId (required)**: the unique identifier of the data item.

- **naaccrNum (optional)**: the NAACCR Item Number of the data item. While the NAACCR community would typically use the NAACCR Item Number to uniquely identify items, those are not a good fit for identifying them in XML data files. The main reason is that different organizations tend to re-use the same numbers for their own custom items; which can cause confusion when multiple organizations need to be referenced in one data file. A secondary reason to require naaccrIds

instead of item numbers is to make the XML data files more readable when opened in a text editor.

The order of <Item> elements under their parent does not matter, and their item value can contain newline characters without disrupting the syntax of the XML document. If an <Item> element does not have a value, it should be omitted entirely from the data file (but it is technically not an error for the item to still appear in the file). On the other hand, it is recommended to include an item if its value only consists of whitespaces (usually spaces and/or tabs) as such a value could have a meaning for specific items.

4.2.2 XML Samples

Sample XML data file with two patient entities, second one has no tumor:

```
<NaaccrData
  baseDictionaryUri="http://naaccr.org/naaccrxml/naaccr-dictionary-160.xml"
  recordType="I"
  timeGenerated="2016-08-16T08:09:19-04:00"
  specificationVersion="1.2"
  xmlns="http://naaccr.org/naaccrxml">
    <Item naaccrId="registryId">0000000001</Item>
    <Patient>
        <Item naaccrId="patientIdNumber">00000001</Item>
        <Tumor>
            <Item naaccrId="primarySite">C123</Item>
        </Tumor>
    </Patient>
    <Patient>
        <Item naaccrId="patientIdNumber">00000002</Item>
    </Patient>
</NaaccrData>
```

Simple XML data file using a user-dictionary and specifying the NAACCR Item Numbers:

```
<NaaccrData
  baseDictionaryUri="http://naaccr.org/naaccrxml/naaccr-dictionary-160.xml"
  userDictionaryUri="http://mycompany.org/mycompany-naaccr-dictionary.xml"
  recordType="I"
  timeGenerated="2016-08-16T08:09:19-04:00"
  specificationVersion="1.2"
  xmlns="http://naaccr.org/naaccrxml">
    <Item naaccrId="registryId" naaccrNum="40">0000000001</Item>
    <Item naaccrId="myCompanyVariable1" naaccrNum="10001">X</Item>
    <Patient>
        <Item naaccrId="patientIdNumber" naaccrNum="20">00000001</Item>
        <Item naaccrId="myCompanyVariable2" naaccrNum="10002">Y</Item>
        <Tumor>
            <Item naaccrId="primarySite" naaccrNum="400">C123</Item>
            <Item naaccrId="myCompanyVariable3" naaccrNum="10003">Z</Item>
        </Tumor>
    </Patient>
</NaaccrData>
```

4.3 Other Considerations

4.3.1 Consolidated Data and Nested Elements

Since the standard defines a nested structure for patients and tumors, registries with consolidated patient information can send a single set of patient data in a <Patient> element when multiple tumors are included in <Tumor> elements. If a registry does not have consolidated patient information, then a separate <Patient> and <Tumor> element will be sent for every tumor record. The standard does not enforce <Patient> or <Tumor> element uniqueness, for example, multiple <Patient> elements can be sent with the same Patient ID and multiple <Tumor> elements can be sent for the same tumor record to represent a patient visit history. The standard has been designed to be flexible enough to accommodate many data collection scenarios, while also supporting consolidated, heavily curated cancer abstracts. In any scenario, <Item> elements must be nested under the appropriate parent, <Patient> or <Tumor>, according to the Base Dictionary.

4.3.2 Placement of Items at the Patient or Tumor Level

One of the most commonly discussed aspects of the standard during its development was how to place the current Vol. II data items at the correct level in the Patient/Tumor hierarchy. The current Vol. II data exchange standard alludes to some patient and tumor delineations, but can be ambiguous and confusing about whether some data items should appear once per patient or once per tumor record. As a result, the current Patient/Tumor designations are derived from a document produced by the NAACCR Consolidated Items workgroup for an initial mapping of patients and tumor items, while realizing that future versions of this standard may need to change those designations. The NAACCR XML standard is designed in such a way that changing the parent element of a data item can be done quickly and with few side effects.

4.3.3 XSD files

The standard includes a W3C-compliant XML Schema Definition (XSD) file called "naaccr_data_VERSION.xsd" (where VERSION is the specifications version) that defines the valid elements and attributes in a NAACCR XML data exchange file. Some XML parsers can use the XSD file to validate the basic syntax of a NAACCR XML data exchange file.

The XSD was specifically designed to avoid both data type validation and <Item> parent-child validation because of two limitations inherent in most XSD validation software. First, many XSD validators read an entire XML file before reporting its validity, making them inappropriate for large registry data exchange files. Second, most XSD validators will reject an entire XML file as soon as they encounter any portion that is non-compliant with the XSD. This behavior is problematic for most NAACCR data exchange use cases that need an overall picture of the portions of a file that are invalid rather than a Boolean valid-invalid result, as well as the ability to ignore certain validation errors based on special circumstances. For these reasons, sophisticated validation of NAACCR XML data exchange files is left to a custom software tool instead of the XSD.

The standard also includes an XSD file for the dictionary called "naaccr_dictionary_VERSION.xsd" (where VERSION is the specifications version).

4.3.4 Validation

The NAACCR XML standard defines a stepwise approach to XML data validation where each step provides an increasing level of validation complexity:

Step 1. Element and Attribute Validation

XSDs provide basic validation of the correct element and attribute naming conventions in a NAACCR XML data exchange file. This step relies on W3C standards-compliant XML parsing software.

Step 2. Data Type and Nesting Structure

The Base Dictionary and User Dictionary files contain all of the information necessary to validate the data types of data items and their nested structure within <Patient> and <Tumor> elements. This step can be accomplished with an XML software tool provided by NAACCR, or custom NAACCR XML processing software.

Step 3. Coding and Context

After conversion of a NAACCR XML file into a the fixed-width data exchange format, standard Edits metafiles can be used the same way they are currently used to validate a NAACCR data exchange file.

4.3.5 Extending the NAACCR XML Standard

The NAACCR XML standard allows custom, user-defined data and metadata at multiple insertion points in a data exchange document. This built-in extensibility is a forward-looking feature of the standard that encourages experimentation with new data types and structures while supporting the development of registry software where the details of an individual registry does not need to be known to the entire NAACCR community. All of these extensibility features can be used without any change to the XSDs.

One means of extending a NAACCR XML data exchange document is to add custom attributes to the root elements in both data files and dictionaries.

Another extensibility feature allows the inclusion of arbitrary XML data in a NAACCR XML data exchange document at the <NaaccrData>, <Patient>, or <Tumor> levels. This extension method allows sophisticated data exchange scenarios where biomarker data, synoptic pathology reports, or anything else that can be encoded as valid XML can be included in a NAACCR XML file. Including data in this manner is different from defining a new <Item> in a custom User Dictionary because it is not bound by the same space limitations. These new data inclusions can be any size and do not have to be defined as <Item> elements with a NAACCR Item Number, but they do have some important limitations. First, the extra data cannot be read or saved into a fixed-width format file. And second, these data extensions require some kind of external data dictionary outside of the NAACCR XML specifications to define their semantics and syntax for communication to other registries.

While these extensions are powerful, custom data that can be contained in an <Item> element and defined in a custom User Dictionary is preferable because it will be preserved

in a conversion to and from the fixed-width format. However, when the custom data is too large to fit within an <Item> element, or it has a sophisticated structure that needs to be retained, the extensibility features of the standard permit advanced users to satisfy those needs.

4.3.6 Guidelines for Creating a CSV File from a NAACCR XML File

As the NAACCR data exchange model transitions from a fixed-width, flat buffer format, into a hierarchical XML data model, there are use cases where a standardized method of creating a flattened version of a limited set of NAACCR XML data items would be helpful. For example, some statistical software applications cannot input or output XML without difficulty, moreover, loading XML into a relational database often requires an intermediate flattening of the data to fit into tables. In these scenarios, when a software application needs to consume a flattened version of data items from a NAACCR XML document, delimited files are the best intermediate data format.

This recommendation should not be confused with the previously supported method of converting between NAACCR XML and the older fixed-width NAACCR format, since a fixed-width file and a delimited file have different characteristics, namely, a delimited file does not require start positions or a specific order of data items. Since XML does not prescribe an order of elements and the NAACCR XML standard is designed to promote deep, hierarchical models of data where the length and presence of large sections of the XML cannot be predicted, a delimited file is better suited for a flattened representation of NAACCR XML data. The added flexibility of the XML data model fits more closely with the flexible nature of delimited files where the presence and order of each item can be changed easily. The complexity of an XML data model will never translate efficiently and completely into a flattened format, but in scenarios where a limited number of data items or a limited level of the data hierarchy needs to be flattened, delimited files are the best choice.
In order to prevent the reintroduction of limitations and problems associated with the previous fixed-width data exchange format, while still addressing the need to flatten data in certain scenarios, the following guidelines are suggested:

1. The delimited file should use a pipe character (|, ASCII 124) as the delimiter
2. Delimiter characters that are not meant to be used as delimiters must be escaped with a backslash(\, ASCII 92)
3. Backslash characters that are not used for escaping a delimiter should be escaped into a double backslash (\\, ASCII 92 + ASCII 92)
4. Line endings that denote the end of a record must contain a CR(ASCII 13) followed by a LF (ASCII 10)
5. All CR and LF characters that are not part of a line ending must be removed
6. All text must be UTF-8 encoded
7. The delimited file must have a header line as the first line in the file
8. The header line must use naaccrIds as header names
9. The naaccrId header names can be in any order
10. There is no minimum or maximum number of naaccrIds that can be included in a delimited file

The NAACCR XML Data Exchange standard was initially created with two levels of hierarchy in the data model, a Patient and a Tumor, but with the intention and extensibility to support additional levels in the future through the NAACCR Volume II change management process and any custom user-defined expansions. Creating a delimited file from an XML file can neither represent the complexity of a hierarchical data model nor retain the integrity of text values that contain newline characters, but with those limitations in mind, a delimited version of a NAACCR XML file can be helpful where a small number of data items needs to be analyzed by software that does not natively support XML or when text values are not needed.

5 Library Requirements

A custom software library for reading and writing NAACCR XML should address the following requirements:

- Provide entities representing the data model of the standard.
- Provide functionality to manipulate those entities.

5.1 Implementing the Data Model

Typically the entities representing an XML data model correspond directly to the XML syntax. The following entities make up the data model of the NAACCR XML standard:

- NaaccrData
- Patient
- Tumor
- Item

The library must also provide the entities for the dictionaries:

- NaaccrDictionary
- NaaccrDictionaryItem

In addition to the attributes and/or text content defined by the standard, the entities can define some convenience methods. Here are a few examples:

- NaaccrData, Patient and Tumor should allow retrieving the value of a given item by its ID.
- NaaccrData, Patient and Tumor should allow retrieving any validation error that happened at their level (like an unknown item ID for example).
- When reading an XML file, NaaccData, Patient, Tumor and Item should all provide the line number in which they start in the data file; this is especially important for this standard which tends to create large XML files with many small lines.

5.2 Implementing the Functionality

5.2.1 Data Functionality

The main purpose of a NAACCR XML library is reading and writing NAACCR XML data files. Because those files can be very large, it is not acceptable for the library to load all the data

into memory. For that reason, the most convenient way to provide the reading and writing functionality is through the concept of streams.

A reading stream needs to support the following functionalities:
- Being opened and initialized with a data file (or some other low-level stream).
- Reading the NaaccrData attributes.
- Reading the items from the NaaccrData level.
- Reading the next Patient.
- Indicating there is no more Patient to read.
- Being closed.

A writing stream needs to support the following functionalities:
- Being opened and initialized with a data file (or some other low-level stream).
- Writing the NaaccrData attributes.
- Writing the items from the NaaccrData level.
- Writing the next Patient.
- Being closed.

More complex operations (like reading/writing an entire file) can easily be implemented in terms of the functionality of the streams. In theory those operations do not need to be provided with the library, but in practice it is convenient to include some of them. The following are some example of such functionalities:
- Reading an entire XML data file, returning a NaaccrData entity.
- Writing an entire XML data file given a NaaccrData entity.
- Getting all the NaaccrData attributes of a given file.

5.2.2 Dictionary Functionality

The dictionaries can be assumed to have a reasonable size and therefore they do not need to be handled with streams. The library needs to provide the following functionality:
- Reading a dictionary from XML.
- Writing a dictionary to XML.
- Validating a base dictionary.
- Validating a user dictionary.
- Getting a base dictionary by URI or by version.
- Getting a default user dictionary by URI or by version.

5.3 Other Features and Considerations

5.3.1 Reporting Errors

There are two types of errors that the library needs to deal with: syntax errors and data validation errors.

Syntax errors are critical and should prevent any further operations on the file. The following are examples of syntax errors:

- Unable to read/write from/to the data file (file not found, operation not allowed, etc.)
- Unknown or misplaced XML tag.
- Missing required attribute.
- Invalid attribute value.
- Multiple items with same NAACCR ID in same entity (Patient or Tumor for example).
- Item under the wrong entity according to its definition in the dictionary.

Data validation errors are not critical (most applications using the library should be able to gracefully handle them). They should be reported to the caller of the library without interrupting the flow of operations. The following are examples of data validation errors:

- Unknown NAACCR ID.
- Wrong NAACCR Number.
- Invalid value according to the item's length.
- Invalid value according to item's data type.

The library can report those validation error as part of different entities (this is convenient for streams which can only return "the next Patient entity") or provide a different mechanism to retrieve the errors for the current entity.

5.3.2 Reading and Writing Options

The library should offer some options to allow an application to customize the behavior of the reading and writing operations. Different libraries might support different options. The following are examples of options a library can support:

- When writing, include the NAACCR Item Numbers.
- When writing, report an error for values that are too long.
- When writing, apply the padding rules.
- When reading, ignore the items with an unknown NAACCR ID.
- When reading, filter the items using an inclusion/exclusion list of NAACCR IDs.
- When reading, turn off the data types validation.

Making those types of options available will allow a library to be much more flexible for the different environments it needs to work in.

5.3.3 Observing the Reading and Writing Operations

A library might allow the processing of large file to be "observed" and a feedback to be returned to the "observer".

In its simplest form, an observer needs to provide feedback on the following basic operations:

- A Patient entity was read.
- A Patient entity was written.

Other events can be captured if needed:

- A NaaccrData entity was read.
- A NaaccrData entity was written.

- A Tumor entity was read.
- A Tumor entity was written.
- An error was reported on a given entity.

A library can be fully functional without allowing an observer, but in practice, applications dealing with large data files always need to have a feedback mechanism.

5.3.4 Compression

Because the NAACCR XML format tends to generate large data files, compression is an import part of the standard.

While compression can be achieve outside the library, it is convenient to abstract it within the library.

The following compression method should be considered:

- GZip (https://en.wikipedia.org/wiki/Gzip)

That compression standard is well supported in a wide variety of languages and in many standalone software (like the free 7-Zip tool).

6 Appendixes

Appendix A – NAACCR IDs renamed because of new length requirement

- dateRegionalLymphNodeDissection: dateRegionalLNDissection
- dateRegionalLymphNodeDissectionFlag: dateRegionalLNDissectionFlag
- phase1RadiationExternalBeamPlanningTech: phase1RadiationExternalBeamTech
- phase1RadiationPrimaryTreatmentVolume: phase1RadiationPrimaryTxVolume
- phase1RadiationToDrainingLymphNodes: phase1RadiationToDrainingLN
- phase2RadiationExternalBeamPlanningTech: phase2RadiationExternalBeamTech
- phase2RadiationPrimaryTreatmentVolume: phase2RadiationPrimaryTxVolume
- phase2RadiationToDrainingLymphNodes: phase2RadiationToDrainingLN
- phase3RadiationExternalBeamPlanningTech: phase3RadiationExternalBeamTech
- phase3RadiationPrimaryTreatmentVolume: phase3RadiationPrimaryTxVolume
- phase3RadiationToDrainingLymphNodes: phase3RadiationToDrainingLN
- radiationTreatmentDiscontinuedEarly: radiationTxDiscontinuedEarly
- numberOfPhasesOfRadTreatmentToThisVolume: numberPhasesOfRadTxToVolume
- npcrDerivedAjcc8TnmPostTherapyStgGrp: npcrDerivedAjcc8TnmPostStgGrp
- chromosome1pLossOfHeterozygosity: chromosome1pLossHeterozygosity
- chromosome19qLossOfHeterozygosity: chromosome19qLossHeterozygosity
- bilirubinPretreatmentTotalLabValue: bilirubinPretxTotalLabValue
- bilirubinPretreatmentUnitOfMeasure: bilirubinPretxUnitOfMeasure
- creatininePretreatmentUnitOfMeasure: creatininePretxUnitOfMeasure
- estrogenReceptorPercentPositiveOrRange: estrogenReceptorPercntPosOrRange
- extranodalExtensionHeadAndNeckClinical: extranodalExtensionHeadNeckClin
- extranodalExtensionHeadAndNeckPathological: extranodalExtensionHeadNeckPath
- gestationalTrophoblasticPrognosticScoringIndex: gestationalTrophoblasticPxIndex
- internationalNormalizedRatioForProthrombinTime: iNRProthrombinTime
- ipsilateralAdrenalGlandInvolvement: ipsilateralAdrenalGlandInvolve
- lnAssessmentMethodFemoralInguinal: lnAssessMethodFemoralInguinal
- lnAssessmentMethodParaAortic: lnAssessMethodParaaortic
- lnAssessmentMethodPelvic: lnAssessMethodPelvic
- lnDistantAssessmentMethod: lnDistantAssessMethod
- lnStatusFemoralInguinalParaAorticPelvic: lnStatusFemorInguinParaaortPelv
- methylationOfO6MethylguanineMethyltransferase: methylationOfO6MGMT
- oncotypeDxRecurrenceScoreInvasive: oncotypeDxRecurrenceScoreInvasiv
- progesteroneReceptorPercentPositiveOrRange: progesteroneRecepPrcntPosOrRange
- progesteroneReceptorSummary: progesteroneRecepSummary
- progesteroneReceptorTotalAllredScore: progesteroneRecepTotalAllredScor
- residualTumorVolumePostCytoreduction: residualTumVolPostCytoreduction
- serumBeta2MicroglobulinPretreatmentLevel: serumBeta2MicroglobulinPretxLvl
- visceralAndParietalPleuralInvasion: visceralParietalPleuralInvasion
- dateOfSentinelLymphNodeBiopsy: dateSentinelLymphNodeBiopsy
- dateOfSentinelLymphNodeBiopsyFlag: dateSentinelLymphNodeBiopsyFlag

6 Document History

Revision Date	Contributing Authors/Description
August 2015	Fabian Depry, Isaac Hands • Initial Version.
January 2017	Fabian Depry • Fixed a few minor descriptions for the tags and attributes.
March 2017	Fabian Depry, Lori Havener, Isaac Hands, Kathleen Beaumont • Added new section to describe changes between versions. • Improved overview section talking about XML version, encoding and default namespace. • Updated dictionary and data specifications sections according to changes made in version 1.2 of the standard. • Reviewed and updated entire document.
August 2017	Fabian Depry • Updated dictionary specifications sections according to change made in version 1.3 of the standard.
April 2019	Fabian Depry, Isaac Hands • Added 32-character restriction to the NAACCR ID as per v1.4 specifications. • Added Appendix A with a listing of the NAACCR 18 IDs that were renamed because of the new length restriction. • Updated NAACCR ID section describing the default algorithm used to create IDs from Names. • Added 50-character restriction to the NAACCR Name as per v1.4 specifications. • Added a section with guidelines on creating a pipe-delimited CSV data file from a NAACCR XML file. • Clarified the how blank values should be handled in NAACCR XML files. • Remove reference to XZ compression in Library Requirements guidelines.