GovXML .Net Standards and Notes

# XSD:Group are definitions that are simply reused

From <http://msdn.microsoft.com/en-us/library/d79t4w33(v=vs.80).aspx>

The <group> element allows the schema designer to globally define a group of elements and then reuse that group in any number of complex types, via references.

The .NET Framework does not have an idiom for expressing element groups in code. Instead, when generating source code from an XML Schema document, Xsd.exe expands each <group> reference via the ref attribute directly into the class corresponding to the <complexType> definition that contains the reference. For each element, a public field is produced.

Before and after every group reference place a comment like

// Begin group ActorSelect

member fields from group go here

// End group ActorSelect

# xsd:choice forces only one member to have a value

Create a function that can be used in the set part of the member that eliminates clears the contents of other items in the choice group For instance msgGetCAPs has multiple items in the xsd choice. I use the function ChoiceClearAllBut() to clear all of the other values

Before and after every choice reference place a comment like

// Begin choice

member fields from choice go here

// End choice

# minOccurs="0" or Optional means field isn't always represented in XML

XMLSerializer doesn't include string values that are null. However, enumerations, date/times, boolean, integer, double... all require a way to suppress the value from being serialized. There is a special naming convention that appends "Specified" after the public member name immediately after the member that works. The following code snippet demonstrates how to do this.

private datasetChangeEnum \_ContextType = datasetChangeEnum.ForceExclusion;

public datasetChangeEnum contextType

{

get { return \_ContextType; }

set { \_ContextType = value; }

}

public bool contextTypeSpecified

{

get

{

if (\_ContextType == datasetChangeEnum.ForceExclusion)

{

return false;

}

else

{

return true;

}

}

}

# xsd:simpleType (enumeration)

All enumerations are in the file "Enumerations.cs" with the sole exception of enumerations used in the class to clear choice blocks

# xsd Namespaces

It would be nice if they worked. Unfortunately the only way, I've found that messages are accepted is to strip namespaces. Maybe they will work in a future version at which point they will have to be added in the appropriate places.

# Naming conventions (files)

All files have one only one class definition as its contents. Files are named the same as the class it holds. There are 4 special files that don't have a Hungarian start:

* Constants - contains constants used (date/time formats...) and defaults for your site in case they are not defined in the .config file
* Enumerations - contains all enumerations used as the member values
* GovXML.cs - is used as the start point for sending/receiving messages from Accela
* Transport.cs - handles serializing/deserializing and sending/receiving the messages

The rest of the files use the following conventions for naming both the file and class. As a hint you can right click on a variable type and select "Go To Definition" to get there quickly. To see what other classes may use the one you are in, you can right click on the definition and select "Find All References" to see where it is used.

* cls (class) - classes defined as part of XML with the following exceptions
* ae (array element) - elements in a sequence that is unbounded but are not in their own complex type
* clsIFC (class ifc) - classes that are defined in ifc2x\_final\_stage2\_03.xsd (because they have a different namespace)
* lct (local complex type) - inline definition in the xsd file, but needs to be indented so requires a new class. I've been naming them as lct[class it was used in][name of complex type]
* msg (message) - these are the request/response pairs sent to/from Accela
* und (undefined) - definitions that match sections of XML found in the response message from Accela. When they are defined in future updates of the xsd definitions we'll rename them to cls

xsd:simpleType (enumeration)

All enumerations are in the file "Enumerations.cs" with the sole exception of enumerations used in the class to clear choice blocks

# Programming standards

* Use the same name if possible as the element in the xsd file.
* Date/Time data types append String for the variables being serialized and use [XmlIgnore] to bypass the actual date/time field
* For private variables that are the backend of the member prefix with an underscore
* Order of items in class
  + Members
  + Constructors
  + Methods

# Modifications

Please document changes when you make them at the top of the class. If you don't want to make a change please forward the request/response xml file so it can be fixed.

# Suggestions

If there is a better way of doing something please suggest it.

If a new constructor would be helpful speak up.