

VDF Feature Requests for WebApp

Everyone,

Thank you so much for the JSON features added to WebApp Server in 15.0. They will be put to use in the new version of the WebGT system before the end of the year.

Meanwhile, working on this new version I have (once again) come across a few “missing features” of the WebApp Server. The absence of said features makes it a slightly painful experience to develop a series of web services in VDF, when the actual message format is important.

I’ve tried to carefully describe those missing features below, and hope that it will be equally carefully considered to include them in an upcoming release of VDF.

Please note that none of the features requested below apply to the new JSON interface, only to XML.

Jakob Kruse, 2009-08-17.

1. Naming array item wrappers

Given the struct definitions:

```
Struct tChild
    String Name
    Integer Age
End_Struct

Struct tFamily
    ...
    tChild[] Children
    ...
End_Struct
```

The XML produced will be:

```
...
<Children>
  <tChild>
    <Name>...</Name>
    <Age>...</Age>
  </tChild>
</Children>
...
```

Automatically using the name of the struct as the name of the “array items” in the XML is clumsy. In this case I could be lured into renaming the “tChild” struct as simply “Child”, which would probably not be a good idea (it’s against recommendations I believe, and it could clash with a table name or something else).

Consider a slightly simpler case where renaming the struct is not even an option, because there is no struct to rename:

```

Struct tFamily
    ...
    String[] Children
    ...
End_Struct

```

From this struct I would get the XML:

```

...
<Children>
  <string>...</string>
  <string>...</string>
</Children>
...

```

I don't know of any way to influence the element name "string" in this case.

In order to give the developer easier and more flexible control over the XML interface of web services I propose adding an option to manually define what items in the array will be named in XML. This could possibly be done using meta-data tags:

```

Struct tFamily
    ...
    { XMLItemName = "Child" }
    tChild[] Children
    ...
End_Struct

```

This should result in XML such as:

```

...
<Children>
  <Child>
    <Name>...</Name>
    <Age>...</Age>
  </Child>
</Children>
...

```

2. Optional struct members

More than once I've found myself wanting or needing to create web service methods that take partly optional input data. Let me describe what I mean.

Let's say you want to create a method which could search some database table by id or by name. You could set it up like this:

```

Struct tInput
    Integer ID
    String Name
End_Struct

Function Search tInput Criteria Returns tOutput
    ...
End_Function

```

This would require the user to supply both ID and Name to use the method, which is not what was intended. We could fix it by splitting “Search” into “SearchByID” and “SearchByName”. But that’s because this is a trivial example. Assume that we have multiple of those optional or mutually exclusive members in the input. Creating one method for each possible combination quickly gets clumsy. Requiring the user to supply blank/zero fields for unused members might be an option. But if those members are not primitive data types but large structs that is also a bad option.

My usual solution to this problem is to use arrays:

```
Struct tInput
    tSomeStruct[] OptionalOne
    tOtherStruct[] OptionalTwo
End_Struct
```

In this definition my intention was not to allow multiple instances of “tSomeStruct” and “tOtherStruct”, but rather exactly zero or one of each. Even though this is a rather simple concept in XML Schema and thus in web service interfaces, there is currently no way to describe it in VDF. The approach above works, but introduces an unnecessary array item element, and fails to describe the intention accurately to web service consumers.

I propose adding an option to define “zero-or-one item arrays” in structs, for the purpose of web services only. This could possibly be done using meta-data tags:

```
Struct tInput
    { XMLItemOptional = True }
    tSomeStruct[] OptionalOne
    { XMLItemOptional = True }
    tOtherStruct[] OptionalTwo
End_Struct
```

Declaring a struct member as optional like this should have two effects:

1. No array item wrapper element should be created in the XML, just as if the member was not declared as an array.
2. The XML element should be declared as minOccurs=”0” and maxOccurs=”1”.

On the VDF side it would just be a normal array, except that I as a developer would know that when I get instances of such an array from a web service method invocation, it will always contain zero or one elements, never more.

3. Value space restrictions

With XML schemas auto generated from VDF source code it is not surprising that very few XML Schema features are controllable. A few of the simpler features for restricting value space would be nice additions though, and would fit in well with the proposed “meta-data tags for struct members” concept I advocated above.

Example:

```
Struct tFacets
    { XMLMinInclusive = 10 }
```

```
{ XMLMaxInclusive = 20 }
Integer Range // valid values are 10-20
{ XMLEnumeration = "apple, pear, lemon" }
String Enumeration // valid values are "apple", "pear" and "lemon"
{ XMLPattern = "[a-z]*" }
String Pattern // valid values are strings containing lowercase a-z only, or empty
{ XMLMinLength = 4 }
{ XMLMaxLength = 20 }
String Length // valid values are strings with length from 4 to 20 chars
{ XMLFractionDigits = 2 }
{ XMLTotalDigits = 6 }
Number Digits // valid values contain no more than 2 fractional and 6 total digits
End_Struct
```

Note that while this feature is nice for validation, I see the primary value of it in giving web service consumers as accurate a description of the web service format as possible. I can put manual validation code in my application, but I would have to communicate those validation rules to consumers using documentation, and they would have to read it and apply it manually. If it's in the WSDL file, all that can be avoided.

For details on the effects that these value space restrictions should have on the generated schema, see http://www.w3schools.com/schema/schema_facets.asp.

4. Struct member naming using reserved words

This is not really a feature request, more like a small nuisance. Sometimes using VDF reserved words (like "from", "to" or "date") as struct member names might be desirable, to produce a certain web service format. This does actually work, although syntax highlighting in the editor is off. The nuisance is that if you want the member name to be "From" or "date", you'll have to use another editor (or possibly turn something off in the Studio), because automatic case conversion will insist on changing it into "from" and "Date". The same applies to all other reserved words naturally.

Since reserved words in the place of struct member names do work, would it be possible for the editor to stop treating them as reserved words?