FILE NAME: BDDL\_developers\_hints\_and\_tips.txt

AUTHOR: BDDL Developers

COPYRIGHT (c) 2013 GLASSWALL SOLUTIONS LIMITED

------------------------------------------------------------------------

1. Introduction

This document is for use by the BDDL developers. If there are any

tips and hints you want to pass on to your colleagues then enter

them in here.

I suggest you use something like this format:

2. Example

Here's a suggestion for the format:

AUTHOR: Julie Noted

DATE: 12 July 2013

TIP: When writing a restriction and you want to test if a bit

is set, DO NOT compare with 1, e.g.

... optional restrictions { fABCD & 0x7 == 1 } <-- WRONG

use this:

... optional restrictions { fABCD & 0x7 }

3. Developer Hints and Tips

Tip 0001 - Table Binaries and "type [\*] item" construct.

--------------------------------------------------------

[Author: Steven Warner]

No Table section is self contained.

I should have made this clearer from the outset. The binary files of

the form 'fibRgFcLcb<nnnn>.<AAAA>.bin' (e.g. 'fibRgFcLcb97.Clx.bin')

are extracted from the 1Table.bin file, which is a stream extracted

from the CFB file that the whole Office file in stored in.

These are extracted for Development Purposes Only !

So the actual storage of the separate 'fibRgFcLcb<nnnn>.<AAAA>.bin'

files is more like this

1Table.bin contains:

fibRgFcLcb97.Clx.bin

fibRgFcLcb97.Cmds.bin

fibRgFcLcb97.DggInfo.bin

fibRgFcLcb97.Dop.bin

fibRgFcLcb97.PlcfBteChpx.bin

fibRgFcLcb97.PlcfBteLvc.bin

fibRgFcLcb97.PlcfBtePapx.bin

fibRgFcLcb97.PlcfFldMom.bin

fibRgFcLcb97.PlcfFldTxbx.bin

fibRgFcLcb97.PlcfSed.bin

fibRgFcLcb97.PlcfSpl.bin

fibRgFcLcb97.PlcfTxbxBkd.bin

fibRgFcLcb97.PlcftxbxTxt.bin

fibRgFcLcb97.PlcSpaMom.bin

fibRgFcLcb97.PlfLfo.bin

fibRgFcLcb97.PlfLst.bin

fibRgFcLcb97.Stshf.bin

fibRgFcLcb97.StshfOrig.bin

fibRgFcLcb97.SttbfAssoc.bin

fibRgFcLcb97.SttbfFfn.bin

fibRgFcLcb97.SttbListNames.bin

fibRgFcLcb97.SttbSavedBy.bin

fibRgFcLcb97.Wss.bin

Which means you cannot make any assumptions about being able to use

the [\*] array range in these files, becuase they actually live inside

a larger structure in the real file and the '[\*]' mechanism will keep

going, into following structures incorrectly.

Tip 0002 - About the usage of 'restrictions' and 'AcceptRule' on arrays

-----------------------------------------------------------------------

[Author: Elvin Barreto-White]

It is best to use 'AcceptRule' in an array of a structure type as

opposed to 'restrictions' on the array.

Example

Avoid doing this:

struct Example {

uint8 Size;

StructType [Size] data optional restrictions {parent.Size == 0x50};

uint64 [15] MoreStuff;

}

struct StructType {

uint8 A;

uint8 B;

<cut>

uint8 z;

}

Do this instead:

struct Example {

uint8 Size;

StructType [Size] data optional;

uint64 [15] MoreStuff;

}

struct StructType {

AcceptRule parent.Size == 0x50;

uint8 A;

uint8 B;

<cut>

uint8 z;

}

In an array data item, the restrictions clause will be applied to

each element, so if there’s a sentinel value in the array (for

example), then the restrictions clause can be used on an array.

Consider this:

Example

struct WhiteSpace {

char[\*] str optional restrictions {iswhitespace()};

}

Here, str, will be the number of characters up to, but not including,

any character that is not whitespace.

or

struct equalsZero {

uint8[15] padding optional restrictions {current == 0x00};

}

Here the 'padding' field will stop processing if it encounter a value

that is not 0.

Tip 0003 - Using 'allowempty'

----------------------------

Use 'allowempty' on Arrays that may or can be empty. It may be best

to apply it to all arrays, unless the spec says otherwise.

Tip 0004 - Bit-masking

----------------------

[Author: Jose Alonso]

For a detailed introduction to bit masking, go to the BDDL

Development page in the Wiki ---> Presentation Notes ---> Bits and

Pieces, written by Martin O'Brien.

Let's suppose we have a one-byte variable, called 'var\_name', equal

to ABCDEFGH (binary). If we want to know the value of the second bit,

G (LSB being H), a way of doing it is as follows:

Example

We reset the other bits (we put them to zero). The binary number used

in this operation is called a mask. In this example, the mask would

be 00000010 (binary), which is 0x02 (hexadecimal).

This is implemented as follows:

(var\_name & 0x02)

The result of this operation is either zero or two, depending on the

value of the G flag.

The bit-masking expression MUST be enclosed into parenthesis.

AcceptRule parent.fEFGHIJKL & 0x80 == 0x80; This is not probably what you want to

AcceptRule (parent.fEFGHIJKL & 0x80) == 0x80; Do this INSTEAD

Tip 0005 - Where you have a string defined in the bdd file

---------------------------------------------------------

You should use one of the standard ones in common/plex.bdd

Tip 0006 - About the usage of 'optional'

----------------------------------------

[Author: Jose Alonso]

If the 'optional' keyword is used, a condition should be always used.

Otherwise, 'optional' is useless.

Example

Avoid doing this:

struct myStruct {

STD std;

uint8 padding optional;

}

Do this instead:

struct Padding {

AcceptRule (sizeof(std) % 2) != 0;

}

struct myStruct {

STD std;

Padding padding optional;

}

Tip 0007 - Accessing specific element of an array

-------------------------------------------------

[Author: Roman Danilov]

It is possible to access specific element of an array with the square

bracket notation as you would in many programming languages:

[comment="Fetches the first element of the aCP array"]

aCP[0];

It is also possible to access a specific substructure from a specific

element of the array. For example:

struct CharCount{

uint8 count;

}

document ParagraphDocument{

CharCount[5] charCount;

[comment="Accesses the first element of the charCount array and uses its count substructure to specify the size of the array"]

char[charCount[0].count] word1;

char[charCount[1].count] word2;

...

}

Tip 0008 - Accessing MS-ODRAW structures

----------------------------------------

[Author: Roman Danilov]

All the properties are stored in a structure called OfficeArtRGFOPTE

alongside their corresponding complex structure. OfficeArtRGFOPTE is

made up of two arrays. The first array stores the properties, made up

of an OfficeArtFOPTEOPID followed by a 4 byte data field. The second

array stores the corresponding complex structures when the fComplex

bit is set.

In order to access the property to check whether the the fComplex, or

any other bit, is set you could do something similar to this:

parent.rgfopte[type() == "ThreeD\_StyleBooleanProperties"].F == 0x0 ? ... : ...;

In this example we are checking whether the F (fc3DConstrainRotation)

bit from the "3D-Style Boolean Properties" property is set ("3D-Style

Boolean Properties" is renamed to "ThreeD\_StyleBooleanProperties"

to keep the DVL checker and miraplacid happy). The square bracket

notation is used to find the ThreeD\_StyleBooleanProperties structure

and then access the F (fc3DConstrainRotation) field. This should

mostly work as long as there exists only one property of a specific

type in the array (from the data we have seen, this seems to be the

case). Replace the ThreeD\_StyleBooleanProperties with your own

structure that you need to access and .F with the field that you

need to access.

Further explanation

----------------------

[Author: Jose Alonso]

“rgfopte” is an array of OfficeArtFOPTE records, which specifies

property table entries (page 97 of [MS-ODRAW]).

If we have

rgfopte[0] = “c3DFillX”;

rgfopte[1] = “c3DFillY”;

rgfopte[2] = “c3DFillZ”;

rgfopte[3] = “ThreeD\_StyleBooleanProperties”;

...

and we want to access the element containing the

“ThreeD\_StyleBooleanProperties” structure, we can do the following

rgfopte[3]

or

rgfopte[type() == “ThreeD\_StyleBooleanProperties”]

Therefore, in order to access any other structure or property set

inside a .dvl or a .bdd file, this prefix has to be used

parent.rgfopte[type() == "pib"]

“pib” being the name of the structure we want to access to.

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Developers: place your tips and hints here. \*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Leave this marker at the end of the file \*\*\*\*\*\*\*\*\*\*\*\*\*