# ArcLink Data Transfer User and Developer Document

## Project Definition

The goal of the project is to create an intermediate step between the embedded software being compiled and the weldset data being interpreted. This step will remove any need for the type of processor to be specified prior to the weldset being generated and it will normalize all data going into the processor in regards to the weldsets. Here the data gets structured in such a way that any board can read it no matter the format which is currently being determined from inside ArcLink.

## 

## Memory Outline:

The first part in each new set of data will be one byte and determine:

1. A new set of data is about to be read.
2. The struct in the list of possible structs that is going to be used which is given an index value.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

Section Data

The byte following is determining what kind of data is about to be read. This consist of:

1. 1 bit determining if the data to come is signed or unsigned.
2. 1 bit indicating that the data type is going to be repeated for this section or not. The repeated data type must be consecutive in the list of properties for the struct being read.
3. 1 bit indicating that the data is stored in an array or not. That is, if the property being read is of type array, this bit is assigned 1.
4. 5 bits indicating what type of data is going to be read in. These types are each assigned an index.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Is Rep

Is Array

Data Type

Signed

The next byte is 1 of 3 things.

1. If in the last byte of data specified that the data type was going to be repeated (0100-0000) then this byte will be the amount of times that type is repeated. An example is if a ‘ushort’ is the type of back to back values then the rep byte would show 0000-0001 for 1 repetition.
2. The byte will be the start of the value assigned to the type. If this is the start of the value then the type will determine how many bits get read.
3. If the type byte specified that the type was a c# string / c char or any array then the next byte will be the length of that string/array.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Number of Repetitions / Value / Length

This general format gets interrupted by the following types of data:

1. A struct in a struct
   1. In this case the data loops back to the beginning specifying once again what struct is going to be looked at and what kind of data is in that struct.
2. An array in an array
   1. This sends the array back to the second portion of the data where each inner-array in the outer-array is processed and made into bytes.
3. An array of structs
   1. This is similar to a struct in a struct except it is repeated for each item in the array.

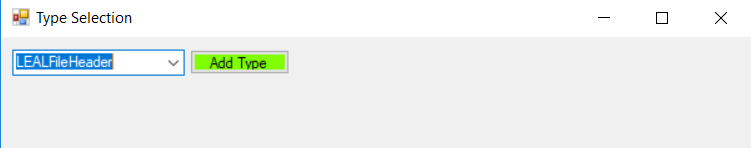
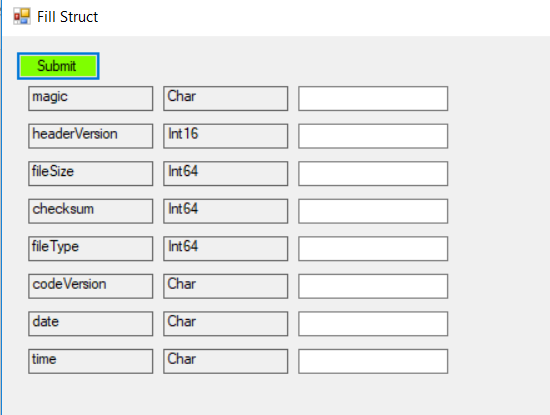
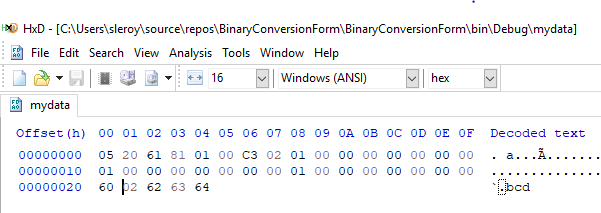
**Section Types Data Types**

|  |  |  |
| --- | --- | --- |
| # | Binary | Struct / Data Type |
| 1 | 0000 | AutoRecord |
| 2 | 0001 | CheckRec |
| 3 | 0010 | DspStateTable |
| 4 | 0011 | FgenDataVerifyStruct |
| 5 | 0100 | ImdPtrU |
| 6 | 0101 | LEALFileHeader |
| 7 | 0110 | LimRec |
| 8 | 0111 | ModeDirectory |
| 9 | 1000 | PwfAmpsEntry |
| 10 | 1001 | PwfAmpsTable |
| 11 | 1010 | PwfCategory |
| 12 | 1011 | PwfCategoryString |
| 13 | 1100 | PwfCOntrol |
| 14 | 1101 | PwfEDSHeader |
| 15 | 1110 | PwfEDSRec |
| 16 | 1111 | PwfExposed |
| 17 | 1-0000 | PwfFat |
| 18 | 1-0001 | PwfFilter |
| 19 | 1-0010 | PwfKey |
| 20 | 1-0011 | PwfProcInfo |
| 21 | 1-0100 | PwfState |
| 22 | 1-0101 | PwfSynergicWFSRec |
| 23 | 1-0110 | PwfSystemType |
| 24 | 1-0111 | PwfWireType |
| 25 | 1-1000 | PWZCompressed |
| 26 | 1-1001 | PWZHeader |
| 27 | 1-1010 | RegRec |
| 28 | 1-1011 | SfRec |
| 29 | 1-1100 | TouchRetract |
| 30 | 1-1101 | WfsControlRec |
| 31 | 1-1110 | none |

|  |  |  |
| --- | --- | --- |
| # | Binary | Type |
| 0 | 0000 | Char |
| 1 | 0001 | Short |
| 2 | 0010 | Long |
| 3 | 0011 | Double |
| 4 | 0100 | String |
| 5 | 0101 | Collection |
| 6 | 0110 | User\_Defined-Type |

Both tables are assigned a binary ID that will tell the system which section or data type it should be expecting to read next. For the section, the properties within it are grabbed and the binary data is loaded into the object accordingly. For the data type, the next X amount of bytes are read and loaded into the property where X is the size of the type.

## Steps to Execute

1. Select the struct type from the drop down list and click “Add Type”.  
   
2. A second screen will prompt the user to fill in the values of each property for the selected struct type and then click submit.  
   
3. Open the file:   
   [Drive the application exists on]\Users\[username]\source\repos\BinaryCOnversionForm\BinaryConversionForm\bin\Debug\mydata.txt  
   in any hex/binary file viewer application to see the results.  
   

## Areas to Address / Notes

1. A long if else statement exists in the writeUnknownType function. Can this be taken down to a few lines where it decides what to convert the object to on the fly and converts it to binary without any metadata included? BinaryFormatter and MemoryStream adds a large amount of metadata to the byte[] making this not plausible. A custom serializer would have to be implemented as far as we know.   
   Notes:
   1. Having a dynamic conversion and using the BitConverter.GetBytes() function does not work because the overloaded method used assumes the value coming in is of type bool but the actual type before running is Object. Our own bitConverter function could be written to take any object but then we run into the same issue of needing to do a custom serializer or the same if/else somewhere else in the project.
2. The other half of this application (the deserializer) is not complete and would be the next big step. A “proof of concept” could be made in c# but preferably this is done in C.
3. There is a commented piece in FillStructForm that should be looked up. It is a failed attempt to make finding an array type dynamic. It is a similar issue to the first item to address ^^^. The long way around this is to explicitly ask “Is this array an integer? Is it a byte? How about a long?
4. The testing UI will not make the inputs for certain classes chosen on the first screen. To test these make an object of that type with values and replace this.currentStruct on the typeSelectionForm file. There is already a test object there as an example.