



Optimizing EDSs for EtherNet/IP™ I/O Connection Target Devices

Viktor Schiffer
Ingenieurbüro Schiffer

October 14, 2015



Agenda

- Introduction
- Methodology
- EDS Details under Investigation
 - Basic Functionality EDS Group
 - General Extensions of Basic Functionality EDS Group
 - I/O Assemblies EDS Group
 - Direct Parameters EDS Group
 - Configuration Assemblies EDS Group
 - Miscellaneous Functionality EDS Group
- Conclusion



Introduction

- EDSs have been around from the very beginning of CIP
- They have been in very common use for DeviceNet
- When ControlNet was introduced a few years later, EDSs were still the preferred configuration method, but the technology leader later decided to go for proprietary configuration methods
- The same happened for EtherNet/IP
- It is good to see the situation is changing by now, yet the overall situation is less than perfect:
 - EDS support varies from company to company and from tool to tool
 - EDSs of existing product often are very basic
 - EDSs of existing products sometimes have to be adapted on a tool by tool basis to give reasonable results
- This presentation give a summary of the current situation and shows a path forward how the situation can be improved



Motivation

- When EDSs for CIP were invented, they were far superior to device descriptions of competing networks
- Configuration tools for DeviceNet made good use of the EDS functionality
- Unfortunately, the use of EDSs on EtherNet/IP is less common with (typically) less functionality than with the device descriptions of competing Ethernet-based networks
- Device developers have no guidance how to create “good” EDSs, i.e. EDSs that work well with many (ideally all) configuration tools
- Chapter 7 of Volume 1 of the CIP Specifications defines all EDS constructs, yet offers little or no guidance on their use



Methodology

- A number of EDS files were created and installed in various EDS-based scanner configuration tools for EtherNet/IP
- These EDS files started as very simple ones with increasing functionality added to test various aspects of the EDS system
- In a few cases certain EDS details would lead to install issues and even software crashes
- Therefore, instead of including a lot of functionality in one single EDS, a kind of “single functionality” approach was taken to make sure the observed effects could be isolated and separated
- Thus, various groups of EDSs were created and tested.



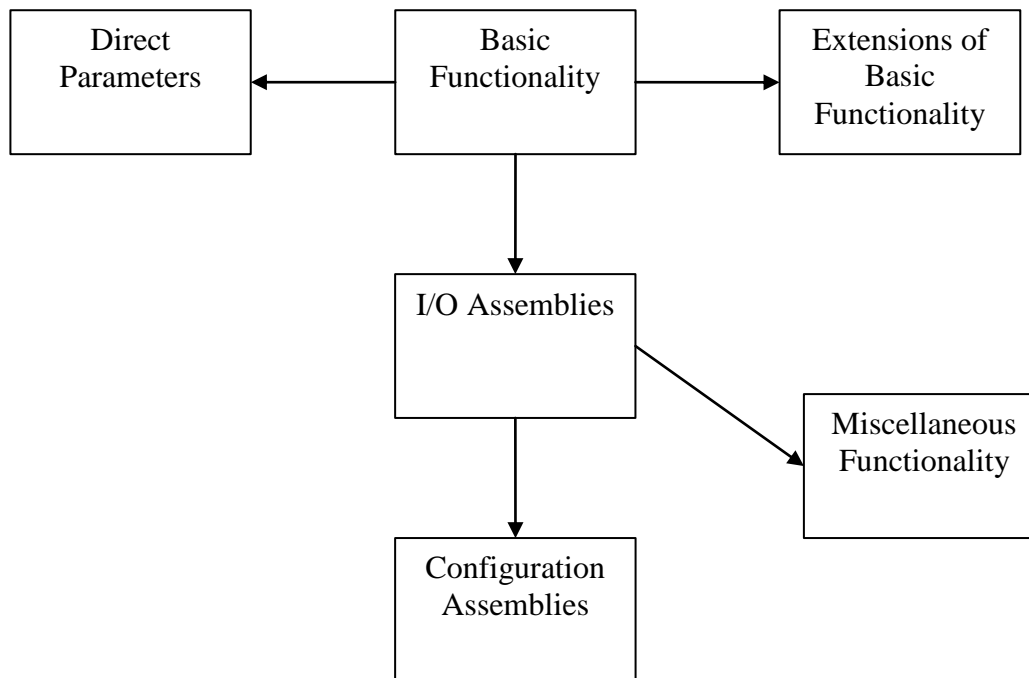
Tools in the Investigation

The following vendors and their tools have been included in the investigation:

- Rockwell Automation (RSNetWorx for EtherNet/IP, Logix Designer)
- Omron (Network Configurator, part of the CX-One package)
- 3S-Smart Software Solutions (Codesys); this technology is also used in Schneider Electric and Bosch Rexroth PLCs
- Moxex (EIP-CT); this technology is also used in Schneider Electric PLCs)
- Hilscher (SYCON.net)
- Mitsubishi Electric (EIP4CCPU)

All publicly shown results are “neutralized”, i.e. no “finger-pointing” takes place
Additional tools may be included in the future if there is sufficient support from the tool vendors

Family of EDS Files





Details shown in this Presentation

- More than 100 EDS files were created in the course of this investigation, covering some 200 to 300 individual aspects (estimated)
- Showing all of them and explaining all details would go far beyond the scope of this presentation
- Therefore, the approach taken in this presentation is to show the Basic Functionality group of EDS files in reasonable detail and then show the Configuration Assemblies group in more detail as an example of what this investigation has revealed



Most Basic EDS File - 1

```
[Connection Manager]
```

```
Connection1 =
```

```
0x04030002,
```

```
0x66640405,
```

```
, 8,,
```

```
, 16,,,,,
```

```
"Exclusive Owner Connection",
```

```
"" ,
```

Only I/O sizes,
no formats

Decoded details
shown in the next two
slides using EZ-EDS

Connection Manager Section, other required EDS details omitted

Most Basic EDS File - 2

[Connection Manager] Entry - Connection1 Comment

Normal Connection ▼

Connection Name: Exclusive Owner Connection Comment Help String: Comment

Trigger and Transport Comment

Transport Class*

<input type="checkbox"/> Class 0: null	<input type="checkbox"/> Class 8: reserved
<input checked="" type="checkbox"/> Class 1: duplicate detect	<input type="checkbox"/> Class 9: reserved
<input type="checkbox"/> Class 2: acknowledged	<input type="checkbox"/> Class 10: reserved
<input type="checkbox"/> Class 3: verified	<input type="checkbox"/> Class 11: reserved
<input type="checkbox"/> Class 4: non-blocking	<input type="checkbox"/> Class 12: reserved
<input type="checkbox"/> Class 5: non-blocking, fragmenting	<input type="checkbox"/> Class 13: reserved
<input type="checkbox"/> Class 6: multicast, fragmenting	<input type="checkbox"/> Class 14: reserved
<input type="checkbox"/> Class 7: reserved	<input type="checkbox"/> Class 15: reserved

Trigger Mode*

- cyclic
- change of state
- application
- reserved (Bit 19)
- reserved (Bit 20)
- reserved (Bit 21)
- reserved (Bit 22)
- reserved (Bit 23)

*If more than one Class or Trigger Mode is selected, the user may be asked which one to use by a tool

Application Type: Exclusive-Owner ▼ Reserved Bits: Bit 28 Bit 29 Bit 30 Direction: Client ▼

Typical settings
for a Class 1
connection

Most Basic EDS File - 3

Connection Parameters Comment

O->T

fixed size supported**
 variable size supported**

Number of bytes per slot in the O->T real time data packet for adapter rack connections (obsolete -> AssemN) 1 byte

Real time transfer format
32-bit run/idle header

reserved (Bit 11)

Connection Type:**
 NULL POINT2POINT
 MULTICAST reserved (Bit 19)

Priority:**
 LOW SCHEDULED
 HIGH URGENT

T->O

fixed size supported**
 variable size supported**

Number of bytes per slot in the T->O real time data packet for adapter rack connections (obsolete -> AssemN) 1 byte

Real time transfer format
connection is pure data and is modeless

reserved (Bit 15)

Connection Type:**
 NULL POINT2POINT
 MULTICAST reserved (Bit 23)

Priority:**
 LOW SCHEDULED
 HIGH URGENT

**In the case of several checkmarks, the user may be asked which one to use by a tool

Typical settings
for a Class 1
connection



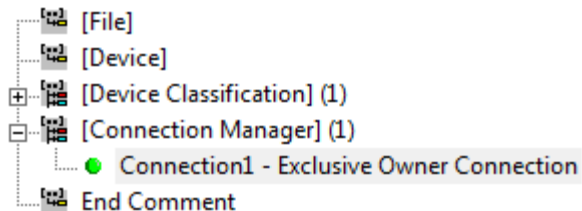
Basic EDS File - Extensions

The following extensions were used to add more content to the most basic EDS:

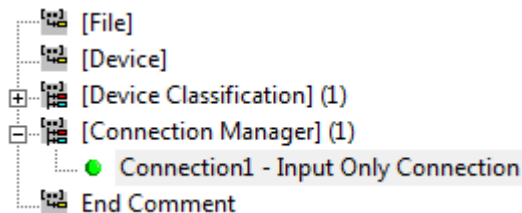
- Input Only and Listen Only connections in addition to the Exclusive Owner connection
- O→T zero length idle indication instead of 32-bit real-time header
- 32-bit header in both directions
- Embedded icon to check its automatic extraction.



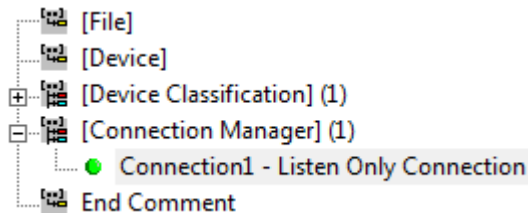
Exclusive Owner, Input Only and Listen Only connections



```
[Connection Manager]
  Connection1 =
    0x04030002,
    0x66640405,
    ,8,,
    ,16,,,,,
    "Exclusive Owner Connection",
    "",
    "20 04 24 01 2C 02 2C 03";
```



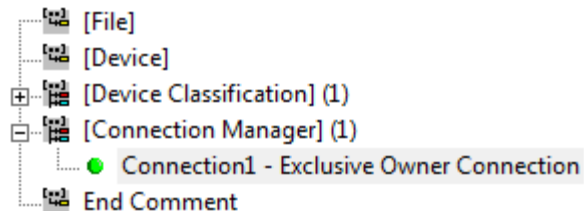
```
[Connection Manager]
  Connection1 =
    0x02030002,
    0x66640305,
    ,0,,
    ,16,,,,,
    "Input Only Connection",
    "",
    "20 04 24 01 2C 64 2C 03";
```



```
[Connection Manager]
  Connection1 =
    0x01030002,
    0x66640305,
    ,0,,
    ,16,,,,,
    "Listen Only Connection",
    "",
    "20 04 24 01 2C 65 2C 03";
```

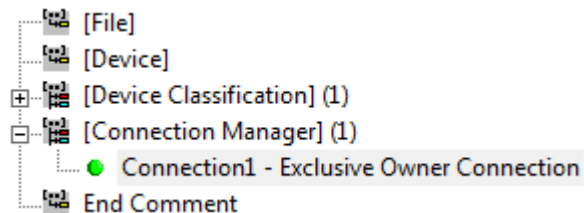
Different paths,
O→T size = 0, other
settings similar

O→T 32-bit real-time header vs. zero length idle indication



Real time transfer format
32-bit run/idle header

O→T
 fixed size supported**
 variable size supported**



Real time transfer format
use 0 data length packet to indicate idle

O→T
 fixed size supported**
 variable size supported**

Not ok

O→T
 fixed size supported**
 variable size supported**

Ok



Results of the Basic EDS File Group

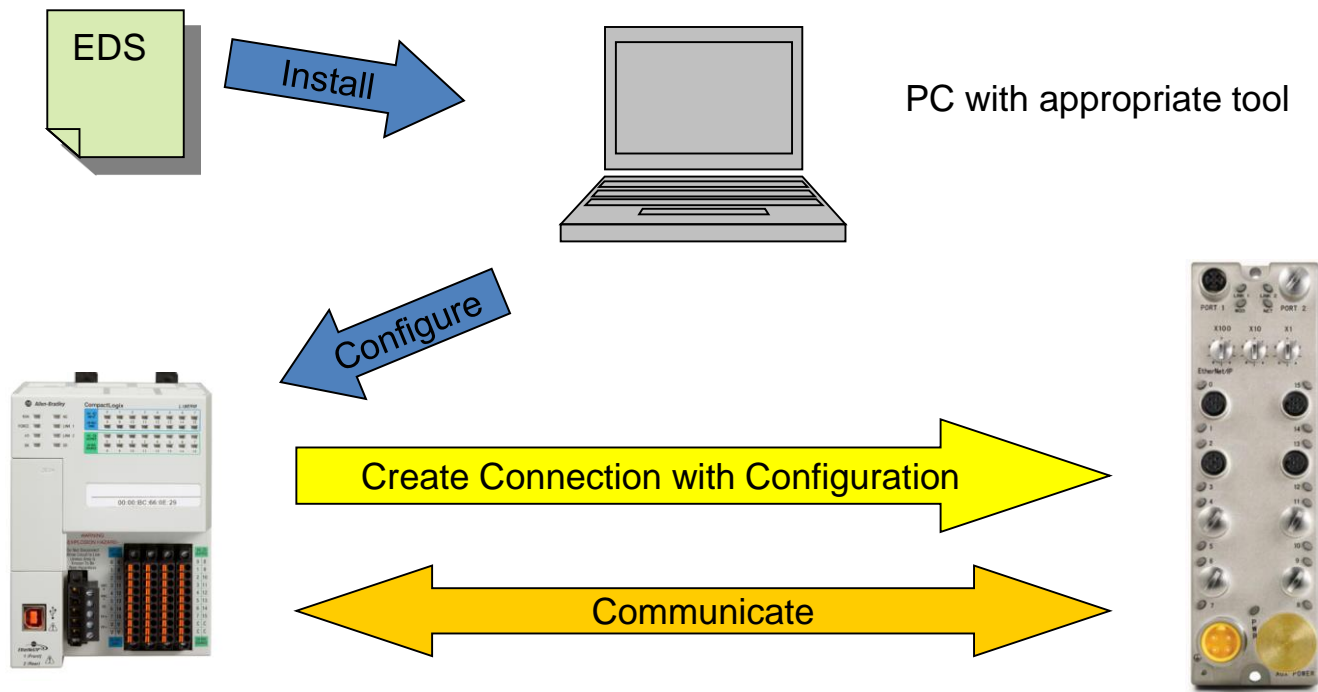
- All common application types (Exclusive Owner, Input Only, Listen Only) were generally understood
- One tool requires more than the bare minimum for the connection to become usable, all others had no problems
- The 32-bit real-time header in O→T direction is supported in all cases, some tools do not seem to support it in the T→O direction
- Zero length data format to indicate idle is only understood and supported by the scanners in some tools, but they support such connections with “fixed size” settings even though “variable size” is required
- Some tools are not “immune” against incorrectly formatted Input Only or Listen Only Connections, i.e. they accept these connections with O→T size > 0
- The majority of tools use the “Product Name” field to represent a product in the resources list, only one of them gives preference to the “Catalog Number” field, if available
- Icons are used in some of the tools; most of those tools support automatic extraction of the icon from the EDS
- The fixed I/O size values in this EDS type are not always interpreted as fixed; some tools require a ParamN entry limited to just one value instead of the I/O size value to make sure the I/O size cannot be modified



Conclusion for Developers, Basic EDS File Group

- Make sure your Input Only and Listen Only connections are correctly formatted for O→T size; EZ-EDS does not catch this error
- Make sure both the “Product Name” and “Catalog Number” fields are filled with meaningful strings
- Do include an icon in the EDS, but be also prepared to supply a separate icon; make sure the icon meets the recommendations spelled out in Section 7-3.6.3 of Volume 1
- Be careful with the 32-bit real-time header in the T→O direction, it may not be understood by some of the tools; if you do not need it, remove it
- If zero length data format to indicate idle is required, then this should be defined in ConnectionN entries in addition to the 32-bit real-time header connections
- If you want the EDS to be understood by all tools in the market, make sure I/O assembly details are included; see also the appropriate section of the paper
- If you want to make sure your I/O sizes cannot be modified, use a ParamN entry limited to the desired number of bytes instead of the I/O size value
- All common application types (Exclusive Owner, Input Only, Listen Only) were generally understood

Configuration Assembly, Principle





The Configuration Assembly

- The Configuration Assembly discussed in these slides is a chunk of data that is sent to the connection target as a simple data segment sent with the Forward_Open request
- This simple data segment contains a collection of parameters in packed form, i.e. only the parameter values
- The individual parameters are described in full in the Parameter Section of the EDS
- Due to the nature of the simple data segment (up to 255 words of data) and other factors that might limit the length of the Forward_Open request, the amount of data is somewhat limited, but data may be packed
- Larger amounts of configuration data can be transmitted by a series of set requests before the connection is established



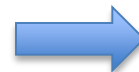
General Structure of a (Configuration) Assembly

Parameter 1
Parameter 2
Parameter 3
Parameter 4
...
Parameter N



```
[Assembly]
```

```
Assem1 =  
" ",,, 0x0000,,,  
32,Param1,  
16,Param2,  
16,Param3,  
16,Param4,  
...  
16,ParamN;
```



```
[Params]
```

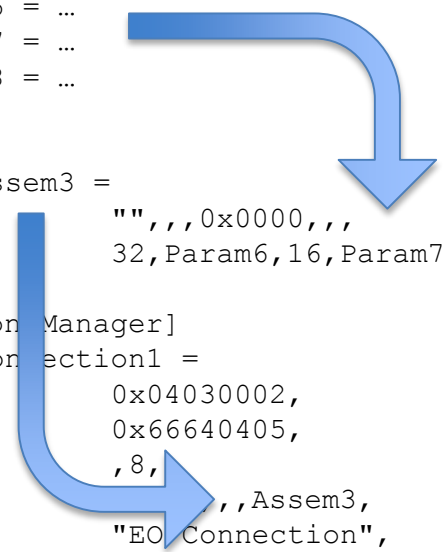
```
Param1 = ...  
Param2 = ...  
Param3 = ...  
Param4 = ...  
...  
ParamN = ...
```

Configuration Assemblies EDS File Group, EDS Detail

```
[Params]
  Param6 = ...
  Param7 = ...
  Param8 = ...

[Assembly]
  Assem3 =
    "",,,,0x0000,,,
    32,Param6,16,Param7,16,Param8;

[Connection Manager]
  Connection1 =
    0x04030002,
    0x66640405,
    ,8,
    ,,Assem3,
    "EO Connection",
    "",
    "20 04 24 01 2C 02 2C 03";
```



Typical example of a configuration assembly defined through parameters

Assem3 is made up of the three parameters defined by Param5, Param7 and Param8, in each case with a bit size that matches the natural size of the parameters



Things to Consider when Creating the Configuration Assembly

- Every element (member) of the assembly needs to be described in the EDS
- Typically, an assembly member is a parameter (described in the [Params] Section of the EDS), a constant value or padding (bits to fill space)
- Assembly members can have the natural bit size of what they consist of or an explicitly stated size (in bits) can be defined; size conflicts are resolved per spec, i.e. size definition overrides natural size
- The ParamN entries of the EDS allow a detailed description of each parameter:
 - Name, unit, help string, may also be given as international strings
 - Data type, data size (in bytes)
 - Min/max/default values
 - Scaling information, if allowed by data type
 - Enumeration, if allowed by data type
 - Parameter properties, e.g. “read only”
- All of the above should be handled correctly by the configuration tool as long as legal constructs per spec are used



User Expectation

- Padding and constants do not need to be visible in the tool, but every parameter should be well represented:
 - By its name; unit and help string will help understanding the meaning without the need to access the product manual
 - By data type; the tool should automatically handle the parameter according to the possibilities and limitations of the data type; incorrectly formatted data is to be rejected
 - Min/max/default values should be visible and data entry outside the allowed range should be prohibited
 - Scaling should be applied correctly and the displayed value should indicate whether it is raw or scaled
 - Enumeration should be available whenever it is defined
 - Parameter properties should be adhered to, e.g. a “read only” parameter should not be modifiable

But what does the user get in reality?



Name, Units, Help String

Value	Encountered Support/Deficiencies	Workarounds
Name	Displayed in all tools, may get truncated	None required, at least 40 characters available
Units	Only displayed in some tools	None
Help String, may only be displayed on demand	Only displayed in some tools	None



Data Types and their Limitations

Data Type	Encountered Support/Deficiencies	Workarounds
Integer	Missing support for 64-bit data types, incorrect/incomplete handling of negative values, no support for hex notation/entry	None, try to avoid LINT and ULINT
Bit string	Missing support for LWORD, not support for hex/binary notation/entry	None, try to avoid LWORD and hex/binary
Real	Missing support for LREAL	None, try to avoid LREAL
String	Only some string data types supported; there is no string data type that is supported by all tools	None



Min/Max/Default Values

Value	Encountered Support/Deficiencies	Workarounds
Min/Max	Missing values may be interpreted as “0”; values may be completely ignored, incorrect interpretation of min/max values for bit string data types	Always use min/max values, no workaround for ignoring min/max values
Default	Supported by all tools, but may be interpreted incorrectly, e.g. no hex notation supported, incorrect/incomplete support for Real data types	Use only decimal notation; not always practical
Value Range	Incorrect/incomplete support of value range, e.g. no negative values allowed	None



Scaling

Property	Encountered Support/Deficiencies	Workarounds
Scaling as such	Only supported by some tools	None
Correctness of scaling	Displayed values may be incorrect, e.g. for large scaled values	None
Scaling indication	A scaled value may look no different form an unscaled one	None



Enumeration

Property	Encountered Support/Deficiencies	Workarounds
Enumeration as such	Only supported by some tools	None
Enumeration for bit string data types	Enumeration may be handled incorrectly, e.g. like for an integer data type	None



Parameter Properties (Descriptor), Incomplete List

Property	Encountered Support/Deficiencies	Workarounds
Read Only (Bit 4)	Only supported by some tools	None
Scaling (Bit 2, 3)	Scaling may not be supported at all	None
Enumeration and Min/Max Range Values (Bit 8)	Only the enumerated values may be used, data entry / enumerated value selection may be screwed up	Try to avoid combination of enumerated and min/max range values



Conclusion for Developers, Configuration Assemblies EDS File Group

- Always fill in min and max values of the ParamN entry
- For bit string min and max values, only set one bit in each of the values, i.e. set the lowest supported bit in the min value and the highest supported bit in the max value
- Try to avoid 64-bit data types
- For string data types, check with the configuration tool for the planned application which string data types are supported; unfortunately, there is no string data type supported by all tools, some support SHORT_STRING but not STRING and vice versa
- Try to avoid large scaling and/or check with the configuration tool for the planned application to what extent scaling is supported
- Try to avoid enumerations outside the min/max value range
- Do not rely on correct interpretation of bit 8 of the parameter descriptor (enumerations & values)
- Try to avoid hex and binary notation for min/max/default values (even where allowed); for bit string data types this is not really practical
- Try to keep parameter names as short as possible (no longer than 40 characters), but still readable
- Try to limit the positive value range of unsigned integer parameters to the value range of the equivalent signed data type



Recommendations for Tool Vendors, Configuration Assemblies EDS File Group

- Some of the bugs encountered in this group are severe; they should be fixed by the tool vendors ASAP:
 - Make sure all empty non-mandatory fields in parameters are interpreted correctly
 - Make sure min/max values are interpreted correctly, some tools simply ignore them
 - Make sure enumeration is supported, some tools simply ignore it
 - Make sure enumeration is supported correctly for bit string data types
 - Make sure hex or binary values in the EDS (where allowed) are understood
 - ...
- Consider adding support for more EDS features to improve user experience



General Conclusions

- The functional capabilities for the configuration of EtherNet/IP devices and their connections provided by the CIP Specifications are excellent and well adapted to the requirements in the market
- However, their “translation” into scanner configuration tools is sometimes faulty and/or poor
- EDS authors can circumnavigate some of these deficiencies, but not all of them
- As a way forward, the following general recommendations are compiled for EDS authors, tool vendors and ODVA



Conclusions for EDS Authors

- Try to keep your EDSs simple and fill in all optional fields whenever this is possible with respect to functionality of the EDS
- Unfortunately, EDSs have to be tested in all tools under which they are to be used;
 - In some cases, tool-specific EDSs may have to be published
 - While this is a perversion of the intent of the CIP Specification, there is no short term fix
 - This should be used as a hammer by the device maker to pressure the tool provider to cleanup his tool



Conclusions for Tool Vendors

- **Fix your bugs and deficiencies ASAP!**
- For the time being: Make your tool available to all device suppliers on a no charge basis on request so they can understand your deficiencies and have app notes, work-arounds and/or alternate EDSs available
- For verification, test your tools with test EDS files that check most or (better) all of the aspects outlined above



Conclusions for ODVA

- While EDSs offer excellent functionality in principle, the lack of support by tool vendors results in bad user experience
 - This has led to competitive disadvantages
 - Massive variation in user guidance and experience between configuration tools
- To improve the situation in the future, try to establish guidelines and tests for scanner configuration tools; this could include multiple compliance levels
 - The test EDS files created for this investigation could serve as a starting point



THANK YOU