



## **A Modern Approach to CIP Device Descriptions**

**Rick Blair**  
**Schneider Electric**

**October 14, 2015**



# Agenda

- Introduction to XML
- Sample Conversion Concepts
- Validation Using Schema
- Extracting Data Examples
- Migration Strategy
- Comparing EDS and XML
- Recommendation

# Introduction to XML

- Designed to describe data
- Self descriptive
- Tree structure
- Tags
  - Case sensitive
  - No spaces

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a comment -->
<collection xmlns:xsi="http://www.w3.org/
  <album type="Vinyl">
    <title>Dark Side of the Moon</title>
    <artist>Pink Floyd</artist>
    <year>1971</year>
  </album>
  <album type="CD">
    <title>Splinter</title>
    <artist>The Offspring</artist>
    <year>2003</year>
  </album>
  <album type="MP3">
    <title>Turn Blue</title>
    <artist>The Black Keys</artist>
    <year>2014</year>
  </album>
</collection>
```

# Introduction to XML

- Designed to describe data
- Self descriptive
- Tree structure
- Tags
  - Case sensitive
- Elements
  - Opening and closing tags

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a comment -->
<collection xmlns:xsi="http://www.w3.org/
  <album type="Vinyl">
    <title>Dark Side of the Moon</title>
    <artist>Pink Floyd</artist>
    <year>1971</year>
  </album>
  <album type="CD">
    <title>Splinter</title>
    <artist>The Offspring</artist>
    <year>2003</year>
  </album>
  <album type="MP3">
    <title>Turn Blue</title>
    <artist>The Black Keys</artist>
    <year>2014</year>
  </album>
</collection>
```

# Introduction to XML

- Designed to describe data
- Self descriptive
- Tree structure
- Tags
  - Case sensitive
- Elements
  - Opening and closing tags
  - Contain other elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a comment -->
<collection xmlns:xsi="http://www.w3.org/
  <album type="Vinyl">
    <title>Dark Side of the Moon</title>
    <artist>Pink Floyd</artist>
    <year>1971</year>
  </album>
  <album type="CD">
    <title>Splinter</title>
    <artist>The Offspring</artist>
    <year>2003</year>
  </album>
  <album type="MP3">
    <title>Turn Blue</title>
    <artist>The Black Keys</artist>
    <year>2014</year>
  </album>
</collection>
```

# Introduction to XML

- Designed to describe data
- Self descriptive
- Tree structure
- Tags
  - Case sensitive
- Elements
  - Opening and closing tags
  - Contain other elements
  - Contain attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a comment -->
<collection xmlns:xsi="http://www.w3.org/
  <album type="Vinyl">
    <title>Dark Side of the Moon</title>
    <artist>Pink Floyd</artist>
    <year>1971</year>
  </album>
  <album type="CD">
    <title>Splinter</title>
    <artist>The Offspring</artist>
    <year>2003</year>
  </album>
  <album type="MP3">
    <title>Turn Blue</title>
    <artist>The Black Keys</artist>
    <year>2014</year>
  </album>
</collection>
```

# Introduction to XML

- Designed to describe data
- Self descriptive
- Tree structure
- Tags
  - Case sensitive
- Elements
  - Opening and closing tags
  - Contain other elements
  - Contain attributes
  - Contain text

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a comment -->
<collection xmlns:xsi="http://www.w3.org/
  <album type="Vinyl">
    <title>Dark Side of the Moon</title>
    <artist>Pink Floyd</artist>
    <year>1971</year>
  </album>
  <album type="CD">
    <title>Splinter</title>
    <artist>The Offspring</artist>
    <year>2003</year>
  </album>
  <album type="MP3">
    <title>Turn Blue</title>
    <artist>The Black Keys</artist>
    <year>2014</year>
  </album>
</collection>
```

# Introduction to XML Schema

- Written in XML
- Validates an XML document against specified criteria
- Validation criteria examples
  - Sequence of elements

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="collection">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="album" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="album">
    <xs:complexType mixed="true">
      <xs:sequence minOccurs="0">
        <xs:element ref="album" minOccurs="0"/>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="artist" type="xs:string"/>
        <xs:element name="year">
          <xs:simpleType>
            <xs:restriction base="xs:integer">
              <xs:minInclusive value="1948"/>
              <xs:maxInclusive value="2015"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Vinyl"/>
            <xs:enumeration value="CD"/>
            <xs:enumeration value="MP3"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>

```





# Introduction to XML Schema

- Written in XML
- Validates an XML document against specified criteria
- Validation criteria examples
  - Sequence of elements
  - Data types

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="collection">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="album" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="album">
    <xs:complexType mixed="true">
      <xs:sequence minOccurs="0">
        <xs:element ref="album" minOccurs="0"/>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="artist" type="xs:string"/>
        <xs:element name="year">
          <xs:simpleType>
            <xs:restriction base="xs:integer">
              <xs:minInclusive value="1940"/>
              <xs:maxInclusive value="2015"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Vinyl"/>
            <xs:enumeration value="CD"/>
            <xs:enumeration value="MP3"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

# Introduction to XML Schema

- Written in XML
- Validates an XML document against specified criteria
- Validation criteria examples
  - Sequence of elements
  - Data types
  - Data ranges

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="collection">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="album" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="album">
    <xs:complexType mixed="true">
      <xs:sequence minOccurs="0">
        <xs:element ref="album" minOccurs="0"/>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="artist" type="xs:string"/>
        <xs:element name="year">
          <xs:simpleType>
            <xs:restriction base="xs:integer">
              <xs:minInclusive value="1948"/>
              <xs:maxInclusive value="2015"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Vinyl"/>
            <xs:enumeration value="CD"/>
            <xs:enumeration value="MP3"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

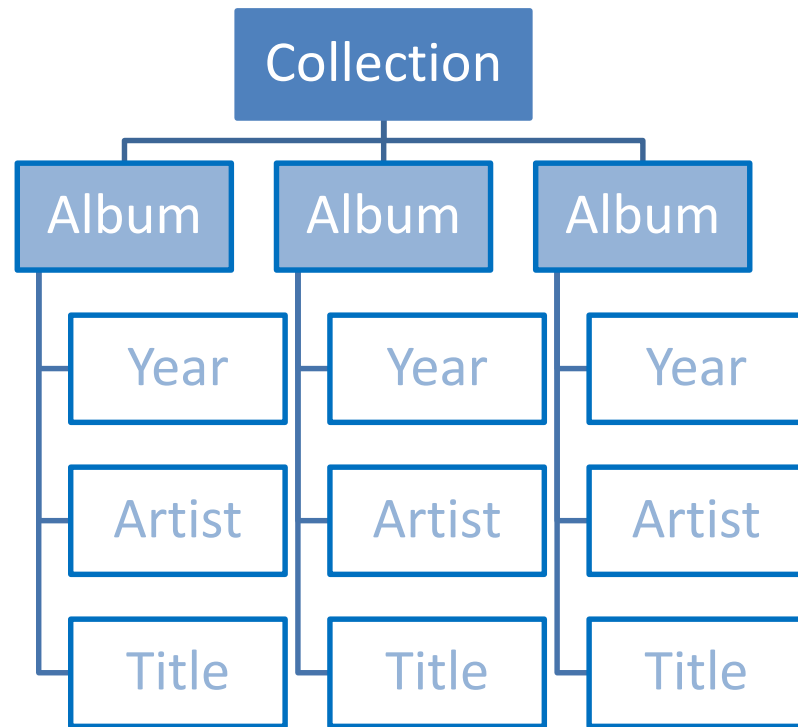
# Introduction to XML Schema

- Written in XML
- Validates an XML document against specified criteria
- Validation criteria examples
  - Sequence of elements
  - Data types
  - Data ranges
  - Data enumeration

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="collection">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="album" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="album">
    <xs:complexType mixed="true">
      <xs:sequence minOccurs="0">
        <xs:element ref="album" minOccurs="0"/>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="artist" type="xs:string"/>
        <xs:element name="year">
          <xs:simpleType>
            <xs:restriction base="xs:integer">
              <xs:minInclusive value="1948"/>
              <xs:maxInclusive value="2015"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Vinyl"/>
            <xs:enumeration value="CD"/>
            <xs:enumeration value="MP3"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

## Introduction to XML Document Object Model (DOM)

- Standard for accessing and manipulating XML
- Views XML documents as tree structure called node tree
- Each element, attribute and text field is a node
- Access nodes by
  - Node name
  - Traversing the tree



## EDS vs. XML File Formats

```

BasicFormat.eds - Notepad
File Edit Format View Help
[section name]$ Comment - extends to end of line
Entry1=value,value,value;          $ Entire entry on one line
Entry2=                             $ Multiple line entry
value,                             $ Field1
value,                             $ Field2
value;                             $ Field3
Entry3=                             $ Combination
value, value,                      $ Fields 1 and 2 on one line
value,                             $ Field3
value;                             $ Field4
Entry5 = 1,                         $ Field 1 specifies the value 1
{1,2,3};                          $ Field 2 specifies an array or
                                $ structure with three values
Entry6 = { 44, {22,33,11} };        $ Entry 6 specifies a single
                                $ The field contains two sets
                                $ The first set is a single
                                $ The second set contains th

```

```

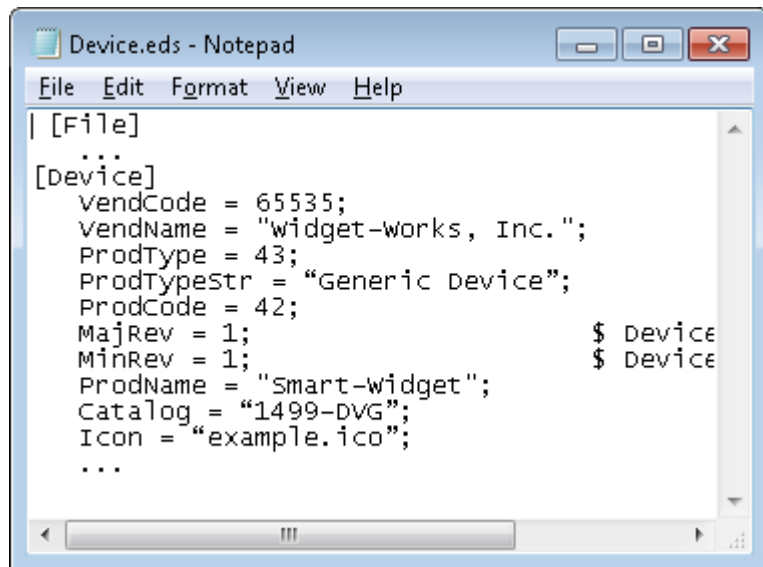
<?xml version="1.0" encoding="UTF-8"?>
<sectionName>
<!-- Entry all on one line -->
  <entry1><field1>value</field1><field2>value</field2><field3>value</field3></entry1>

<!-- All elements (fields) on separate lines -->
<entry2>
  <field1>value</field1>
  <field2>value</field2>
  <field3>value</field3>
</entry2>

<!-- Mixed format -->
<entry3>
  <field1>value</field1> <field2>value</field2>
  <field3>value</field3>
  <field4>value</field4>
</entry3>
</sectionName>

```

## Device Section



```
Device.eds - Notepad
File Edit Format View Help
[File]
...
[Device]
vendCode = 65535;
vendName = "widget-works, Inc.";
prodType = 43;
prodTypeStr = "Generic Device";
prodCode = 42;
MajRev = 1;           $ Device
MinRev = 1;           $ Device
prodName = "Smart-widget";
Catalog = "1499-DVG";
Icon = "example.ico";
...
```

```
<?xml version="1.0" encoding="UTF-8"?>

<Device>
  <VendorCode>65535</VendorCode>
  <VendorName>Widget Works, Inc.</VendorName>
  <ProductType>1.</ProductType>
  <ProductTypeString>Generic Device</ProductTypeString>
  <ProductCode>42</ProductCode>
  <MajorRevision>1</MajorRevision>
  <MinorRevision>1</MinorRevision>
  <ProductName>Smart Widget</ProductName>
  <Catalog>1492-RAB</Catalog>
  <Icon>Example.ICO</Icon>
</Device>
```

# Parameter Section

```
Param.eds - Notepad
File Edit Format View Help
[File]
...
[Params]
Param1 = 0,
6,"20 04 24 68 30 03", $ Link Path Size, Lir
0x0002, $ Descriptor
0xD1, 1, $ BYTE Data Type, Dat
"Idle Action", $ Name
" ", $ Units
" ", $ Help string
,,0, $ Min/Max/default
,,, $ Scaling
Enum1 =
0, "Fault",
1, "Stop",
2, "Zero Data",
3, "Hold Last",
4, "Send Flt Cfg";
Fixed1 =
1, 0, $ The idle action is never stop
4, 1; $ The fault configuration is a
Param2 = $ not addressable from link...
0, , 0x0082, 0xC6, 1, "speed control",
ConstructedParam1 =
"20 4e 24 01 30 09", $ Link Path
0x0000, $ Descriptor
1,10, $ SIGNED_ODOM
"Total Energy", $ Name
" ", $ Units
{"Twh", "Gwh", "Mwh", "kwh", "wh"}, $ Member unit:
"Total energy in kwh", $ Help string
{"This is Terawatt-Hours",
"This is Gigawatt-Hours",
"This is Megawatt-Hours",
"This is Kilowatt-Hours",
"This is Watt-Hours"},
{-999,-999,-999,-999,-999}, $ Member Help
{999,999,999,999,999}, $ Min
{0,0,0,0,0}, $ Max
1, $ Default
-3; $ 1 Type Spec
$ decimal point
```

```
<?xml version="1.0" encoding="UTF-8"?>
<product>
  <parameters>
    <parameter id="1">
      <path>20 04 24 68 30 03</path>
      <descriptor>
        <getEnumStrSupported>1</getEnumStrSupported>
      </descriptor>
      <name>IO Action</name>
      <data size="1" type="BYTE">
        <default>0</default>
        <enum bit="0">Fault</enum>
        <enum bit="1" fixed="0">Stop</enum>
        <enum bit="2">Zero Value</enum>
        <enum bit="3">Hold Last Value</enum>
        <enum bit="4" fixed="1">Send Flt Cfg</enum>
      </data>
    </parameter>
    <parameter id="2">
      <descriptor>
        <getEnumStrSupported>1</getEnumStrSupported>
        <disjointEnumsSupported>1</disjointEnumsSupported>
      </descriptor>
      <name>Speed Control</name>
      <data size="1" type="USINT">
        <default min="3" max="12">3</default>
        <enum value="3">Slow</enum>
        <enum value="8">Medium</enum>
        <enum value="12">Fast</enum>
      </data>
    </parameter>
    <parameter id="3" constructed="TRUE" members="5" size="10" units="kWh">
      <path>20 4E 24 01 30 09</path>
      <name>Total Energy</name>
      <help>Total energy in kWh</help>
      <data id="1" size="2" type="INT" units="TWh">
        <default min="-999" max="999">0</default>
        <help>This is Terawatt-Hours</help>
      </data>
      <data id="2" size="2" type="INT" units="GWh">
        <default min="-999" max="999">0</default>
        <help>This is Gigawatt-Hours</help>
      </data>
      <data id="3" size="2" type="INT" units="MWh">
        <default min="-999" max="999">0</default>
        <help>This is Megawatt-Hours</help>
      </data>
      <data id="4" size="2" type="INT" units="kWh">
        <default min="-999" max="999">0</default>
        <help>This is kiloWatt-Hours</help>
      </data>
      <data id="5" size="2" type="INT" units="Wh">
        <default min="-999" max="999">0</default>
        <help>This is Watt-Hours</help>
      </data>
      <!-- Decimal point location between Wh and kWh -->
      <typeSpecific id="1">-3</typeSpecific>
    </parameter>
  </parameters>
</product>
```





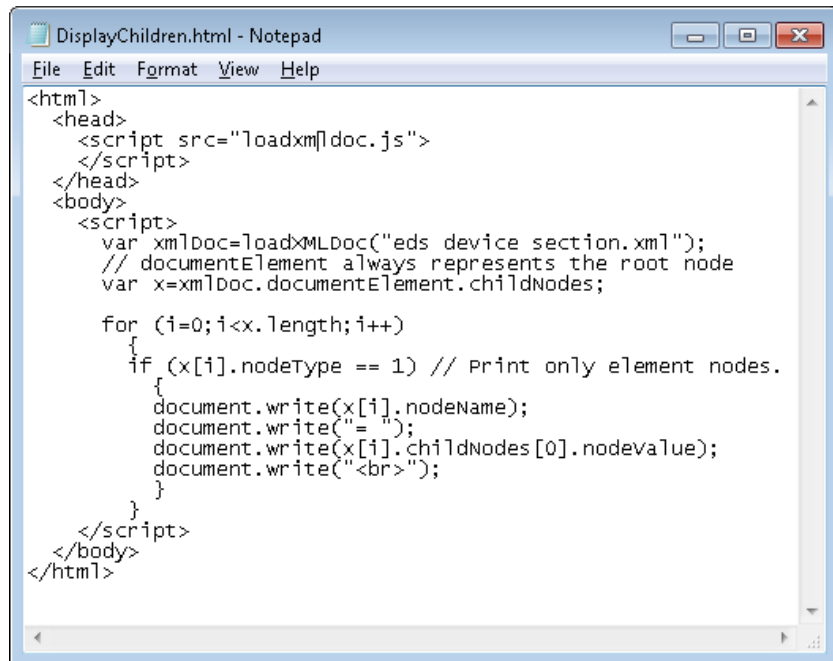
# Validation Example Using Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="CIP_DD_V03_18.xsd">
  <parameters>
    <parameter id="1">
      <name>RPILimits</name>
      <helpstring>This parameter defines the minimum, maximum and default RPI values for this product.</helpstring>
      <data>
        <datasize>4</datasize>
        <datatype>UDINT</datatype>
        <dataunits>microseconds</dataunits>
        <defaultvalue>10000</defaultvalue>
        <minvalue>2000</minvalue>
        <maxvalue>50000</maxvalue>
      </data>
      <descriptor>1</descriptor>
    </parameter>
    <parameter id="2">
      <name>InputSize</name>
      <helpstring>This parameter defines the input data size</helpstring>
      <path>20 04 24 65 30 04</path>
      <data>
        <datasize>2</datasize>
        <datatype>UINT</datatype>
        <dataunits>bytes</dataunits>
        <defaultvalue>0</defaultvalue>
        <minvalue>0</minvalue>
        <maxvalue>496</maxvalue>
      </data>
      <descriptor/>
    </parameter>
  </parameters>
</product>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="product">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="parameters"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="parameters">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="parameter" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="parameter">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="helpstring" type="xs:string"/>
        <xs:element name="path" minOccurs="0"/>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="([0-9a-fA-F]{2}){2}([0-9a-fA-F]{2}){2}([0-9a-fA-F]{2}){2}"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:element ref="data"/>
        <xs:element name="descriptor" type="xs:unsignedShort" default="0" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:positiveInteger" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="datasize" type="xs:positiveInteger"/>
        <xs:element name="datatype">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="INT"/>
              <xs:enumeration value="DINT"/>
              <xs:enumeration value="UINT"/>
              <xs:enumeration value="UDINT"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="dataunits" type="xs:string"/>
        <xs:element name="defaultvalue" type="xs:string"/>
        <xs:element name="minvalue" type="xs:string"/>
        <xs:element name="maxvalue" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

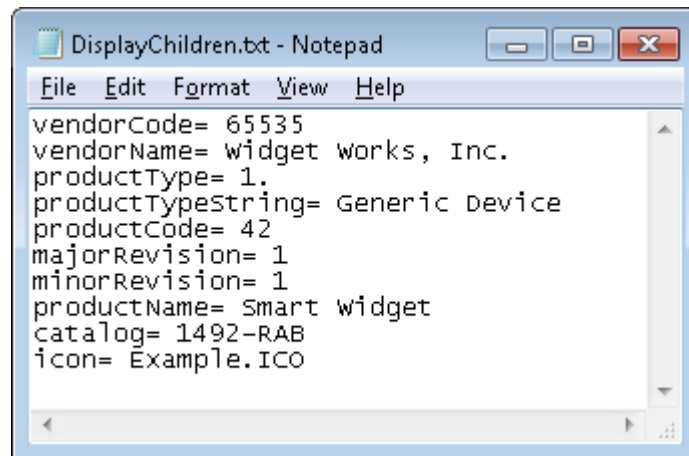


## Example Extracting Data Using Relationships



```
File Edit Format View Help
<html>
  <head>
    <script src="loadxml.js">
    </script>
  </head>
  <body>
    <script>
      var xmlDoc=loadXMLDoc("eds device section.xml");
      // documentElement always represents the root node
      var x=xmlDoc.documentElement.childNodes;

      for (i=0;i<x.length;i++)
      {
        if (x[i].nodeType == 1) // Print only element nodes.
        {
          document.write(x[i].nodeName);
          document.write(" = ");
          document.write(x[i].childNodes[0].nodeValue);
          document.write("<br>");
        }
      }
    </script>
  </body>
</html>
```



```
File Edit Format View Help
vendorCode= 65535
vendorName= widget works, Inc.
productType= 1.
productTypeString= Generic Device
productCode= 42
majorRevision= 1
minorRevision= 1
productName= Smart widget
catalog= 1492-RAB
icon= Example.ICO
```

## Example Extracting Data Using Node Name

```

DisplayByTag.html - Notepad
File Edit Format View Help
<html>
<head>
<script src="loadxml.js">
</script>
</head>
<body>
<script>
var xmlDoc=loadXMLDoc("NIP_Parameters.xml");
// Place all parameter elements in param
var param=xmlDoc.getElementsByTagName("parameter");
for (i=0;i<param.length;i++)
{
document.write("Element Name = ");
document.write(param[i].
getElementsByTagName("name")[0].childNodes[0].nodeValue);
document.write("<br>");
document.write("Element Help string = ");
document.write(param[i].
getElementsByTagName("helpstring")[0].childNodes[0].
nodeValue);
}
</script>
</body>
</html>

```

```

DisplayByTag.txt - Notepad
File Edit Format View Help
Element Name = RPILimits
Element Help String = This parameter defines the minimum, maximum and d
Element Name = InputSize
Element Help String = This parameter defines the input data size
Element Name = OutputSize
Element Help String = This parameter defines the output data size
Element Name = DiagnosticSize
Element Help String = This parameter defines the Diagnostic data size
Element Name = DiagnosticRPILimits
Element Help String = This parameter defines the minimum, maximum and d

```



## Migration

- Create new XML based DD Specification
  - Opportunity to start with a relatively clean slate
  - Remove limitations based upon current implementation
  - Remove constructs that cannot be validated using schema
- Create translation tool
  - Use EZ EDS as base
- New tools use XML
- Legacy tools support both XML and EDS (for a period of time)



## Conclusion

- XML parsing is built into many modern compilers and web browsers
- XML is understood by most recent Computer Science graduates
- Schema can and should be updated at the same time as XML enhancements
- XML files can be dynamically verified and validated by tools prior to data extraction
- XML supports more than just ASCII
- XML is extensible
- XML can be made human readable without user comments

**It's time once more to investigate XML as the future for describing devices**



THANK YOU