LOTAR

LONG TERM ARCHIVING AND RETRIEVAL

TECHNICAL SPECIFICATION

"PRODUCT STRUCTURE VALIDATION"

Release 2.2

2019-09-11

Status: Released

LOTAR

Jochen Boy
PROSTEP AG
jochen.boy@prostep.com

Jean-Yves Delaunay
Airbus
jean-yves.delaunay@airbus.com

Rick Zuray
The Boeing Company
richard.s.zuray@boeing.com

Technical

Cullen Simpson
Gulfstream Aerospace
cullen.simpson@gulfstream.com

Jeff Klein
The Boeing Company
jeff.r.klein@boeing.com

Contents

Table of Contents

List of Figures

List of Tables

Document History

Revision	Date	Change
1.0	2013-09-23	Initial creation
1.1	2013-10-23	Incorporation of team review feedback
1.2	2013-10-28	Final editorial changes and release for publication
2.0	2019-03-10	Updated based on feedback from Jotne implementation attempt and Gulf-stream usage. Modified AHASH calculation for assemblies. Added allowance for additional hash algorithms. Added examples for assemblies with positional information of children. Added examples for multi valuated attributes. Added examples for CAD and other files along with associated hash methodology.
2.1	2019-08-15	Eliminated ahash_rank references Corrected some minor wording in section 7.2.1
2.2	2019-09-11	Added comment in section 1.2 about XML being only an example and that other formats are acceptable. Eliminated ahash_rank reference in section 1.5.6 Added reference to Extensible Markup Language (XML) 1.0 (Fifth Edition) Corrected a comment about the ahash calculation in 4.2 step 2. Rewrote section 5.1.1 Changed section 5.3 from Float to Double and provided better references. Combined Date, Time and Date Time into a single major subsection 5.4 rather than multiple subsections and added definitions for clarity. Changed Section 6 from Hash Algorithm to Hash Identification. Created subsections for Algorithm and Specification and updated examples in 7

1 Introduction

This document is intended to clearly describe the recommendations of the LOTAR PDM Team as it pertains to product structure validation. The document includes the outcomes of the research contracted by the LOTAR PDM team. This document, in the future, may be absorbed by, or referenced by the recommended practices that will be developed by the PDM Implementor Forum (PDM-IF).

1.1 LOTAR

The LOTAR team is an international working group jointly hosted by ASD-Stan and the prostep ivip association in Europe, and PDES Inc, and AIA in the US. Its aim is to develop a standard designed to provide the capability to store digital product information in a standard neutral form that can be read and reused throughout its lifecycle, independent of changes in the IT application environment originally used to create it. The multi-part standard EN/NAS 9300 covers both the information content and the processes required to ingest, store, administer, manage and access the information.
The scope of this technical report refers to LOTAR Part 210 Edition 1.

1.2 Overview

It is recognized in LOTAR PDM that Product Structures are the key to successful archive and retrieval of PDM data. These product structures must be maintained precisely and a proof that this has been accomplished is required. Product structures can only be reused if the exact product structure content is known to represent the required structure.

The attribute “validation property” is used to make sure that the part attributes loaded into the second PDM (or archive) are the same as the ones extracted from the original PDM (or archive). This validation property is also used to validate that the node represents a re-usable structure. Often nodes will have similar names in the archive (Part Number) but are archived for different purposes (ex: Reference Structure, Manufacturing, Conformity, Design and Construction, Usage, etc.). Typically, a structure stored for the function of Reference Data is very different from the one stored for Design and Construction. Each node may also contain different attributes as well depending on function.

This document defines a method to validate a product structure using hashes of attributes and hierarchical relationships. The examples in this document are all XML. The algorithm described in this document will work in any format, but the attribute representations will need to be adjusted accordingly.

1.3 Scope of this document

This document defines a Recommended Practice for Product Structure validation. The objective is to validate the product structure of data ingested, extracted or re-used by the archive.

This document defines a method to uniquely identify each node in the product structure and to uniquely define the structure of each assembly node.

1.4 Out Scope of this document

This document will not provide validation properties for documents; CAD or other.

1.5 Definition of terms

Some terms used in this document have different meanings in different contexts. Therefore, a definition of how these terms are used in this document is given.

1.5.1 Detail Node

Any part defined in the product structure that has no children specified.

1.5.2 Assembly Node

Any part defined in the product structure that has children specified.

1.5.3 Hash Value

A hexadecimal string created by a standard cryptographic algorithm. The SHA-1 algorithm is what is referenced in this document. Other Industry Standard hash mechanisms may be used in place of SHA-1. This hash provides a unique signature that is altered if any of the referenced data differs. Uniqueness is the goal of this hash signature, not encryption.

1.5.4 AHash Attributes

The list of attributes whose values are concatenated and passed into the hash algorithm to generate a hash value.

1.5.5 CPAH

The string which results from running the hash algorithm against the concatenated values of the AHash attributes.

1.5.6 AHash Value

The hash value that will be stored in the XML for each part to validate the package on extract. The AHash value of a Detail part is equivalent to the CPAH. The AHash value of an Assembly is calculated from the CPAH of the Assembly and the unique identifiers and quantities of its children as explained in section 4.2.

2 References

1. Patent Number 4309569 – Method of Providing Digital Signatures – Merkle 01/05/1982
2. Unicode Technical Report #13 – UNICODE NEWLINE GUIDELINES - 11/23/1999
3. The Unicode Standard, Version 4.0
4. FIPS 180-4: Secure Hash Standard (SHS) – NIST – March 2012
5. Extensible Markup Language (XML) 1.0 (Fifth Edition): https://www.w3.org/TR/xml/

3 Node Unique Definition

Structures are made up of two types of Nodes: Detail and Assembly.

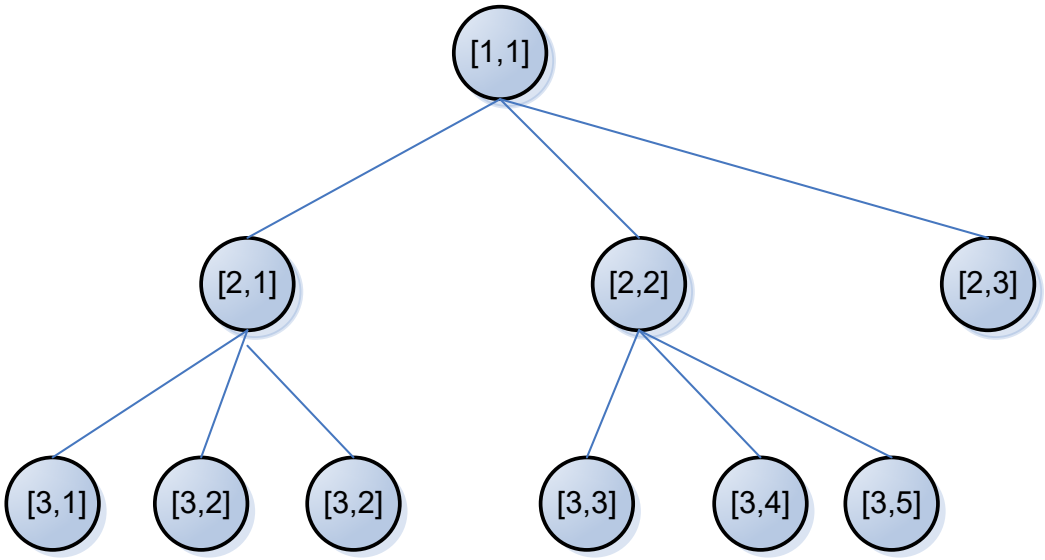

Figure 1 - SAMPLE PRODUCT STRUCTURE

3.1 Detail Node

A Detail Node is defined as any node in the structure that does not have any children in the structure. Detail Nodes may contain a different set of attributes depending on the type or classification of Detail part. For instance, details whose type design data is owned by the Company will likely contain many more significant attributes than Details that represent Industry Standard parts.

Inseparable assemblies and purchased assemblies with no defined product structure would also be considered detail nodes since the PDM system would not break them down any further.

Detail Nodes can reference files, CAD or otherwise. This reference is not included in this validation property.

Examples in Figure 1: [2,3] and all [3,x]

3.2 Assembly Node

An Assembly Node is defined as a node that is dependent on other nodes (children).
An Assembly node may have attributes of its own.

Assembly Nodes can reference files, CAD or otherwise. This reference is not included in this validation property. The referenced files have an independent hash as defined in section 4.3.

Examples in Figure 1: [1,1], [2,1], [2,2]

4 Attribute Validation Properties

A list of attributes used to generate the unique attribute hash must be defined. Typically, this includes all critical attributes for the function that the data is being archived.

The included attributes are concatenated in a specific order as defined by the AHashAttributes element and then converted to a hash signature using an industry standard algorithm. The algorithm used should be specified.

It is recommended that the validation property be maintained outside the data package for use when determining re-use of data in the archive or PDM.

```
<Validation>
      <AHashAttributes>PartNumber,Revision,…</AHashAttributes>
      <AHash>field value</AHash>
      <AHash_Algorithm>SHA1</AHash_Algorithm>
      <BHash>field value</BHash>
      <BFileName>field value</BFileName>
      <BHash_Algorithm>SHA512</BHash_Algorithm>
      <CHash>field value</CHash>
      <CFileName>field value</CFileName>
      <CHash_Algorithm>SHA512</BHash_Algorithm>
</Validation>
```

Figure 2 - EXAMPLE PART VALIDATION XML FORMAT

4.1 AHashAttributes

AHashAttributes is a list of the attributes that have their values included in the AHash calculation. This is a subset of the attributes in the part XML files.
The AHash attribute list is defined within the validation section as shown in Figure 2. This element defines the list of items to be concatenated and the order of the concatenation. It is not the formula, meaning that the field separator is not included in the actual concatenation and neither is the format. Each attribute is assumed to be a string (Text) by default. Deviation from that default can be defined by leveraging the “format” specification with each attribute as shown in section 6.2.

4.2 AHash

AHash is the hash value of the node based on the attributes defined by the AHashAttributes value and the unique identifier for and quantity of its direct children.

The AHash value is calculated using a two-step process.

Step 1: Current Part Attribute Hash (CPAH)

The value is calculated by concatenating all the attribute values listed in the AHashAttributes values. The values must be order of concatenation is specified in the AHashAttributes element.

The AHash of a detail node is equal to the CPAH of the object.

Step 2: Assembly AHash

The AHash of an assembly is the CPAH of the assembly concatenated with a unique key for each distinct direct child part and its quantity. Each value is separated by a colon. The unique key for the part can be any combination of recognizable attributes which provide enough information to uniquely identify that part, such as Part Number and Part Revision or Part ID and Part Revision. The child information in the concatenated string should be in alphanumeric order based on the unique key of each child.

The child AHash values are concatenated in alphanumeric order by AHash value.

```
CPAH:Child1_ID:Child1_Rev:Child1_Qty:Child2_ID:Child2_Rev:C
hild2_Qty

qty# is a number.
```

Figure 3 - ASSEMBLY HASH FORMULA

4.3 BHash and CHash

The validation section also contains hash signatures of any files that need to be intrinsically tied to the object. In the example in Figure 2, BHash is used to provide a SHA512 signature of a PDF file of the BOM associated to the part and CHash is used to provide a SHA512 signature of the CAD file associated to the part. The files to validate will be site specific. Their names may be referenced in the attribute section in the XML or called out explicitly in the validation section.

The SHA512 algorithm is used for files rather than SHA1 due to its greater length but any industry standard algorithm that ensures file uniqueness may be used and should be specified in the appropriate section of the validation properties as shown in Figure 2.

5 Data Definition

The following data types are used to extend the AHashAttributes beyond the text format.

Data that has a semantic meaning other than simple text can be stored in multiple formats for differing reasons. To clarify the calculation all semantic data formats will converted to simple text formats based on the following rules. The data does not need to be stored in this format; this is only required for calculation of the validation property.

All data should be encoded as UTF-8.
Time and DateTime values should be represented in UTC for purposes of elements that are included in the AHash attributes. Local time may be used in the XML for elements that are not included in the AHash calculation. Using local time in the AHash calculation would be very difficult to support validation across multiple sites.

5.1 Text

Encode as stored. This would include all string and integer elements.

5.1.1 End-Of-Line (CR-LF, LF-CR, LF)

There is a wide variety of methodologies in applications for defining the end of line. A consistent format should be used to represent the end of line character(s) for purposes of calculating the hash used for validation.

ISO specifies in ASCII that you can use either CR-LF or just LF. The W3C organization specifies in section 2.11 End-Of-Line Handling of Extensible Markup Language (XML) 1.0 (Fifth Edition):

To simplify the tasks of applications, the XML processor MUST behave as if it normalized all line breaks in external parsed entities (including the document entity) on input, before parsing, by translating both the two-character sequence #xD #xA and any #xD that is not followed by #xA to a single #xA character.

Both #xA and 0x0A are representations of LF. As indicated above any use of the following encodings in data should be converted to LF for purposes of calculating the validation property. Each implementation of this specification will need to adjust the End of Line representation in the XML into the application specific requirements upon loading.

Type	Description	ASCII HEX	ISO 10646-1 [Unicode]
LF	Line Feed	0x0A	U+000A
VT	Vertical Tab	0x0B	U+000B
FF	Form Feed	0x0C	U+000C
CR	Carriage Return	0x0D	U+000D
CR-LF		0x0D then 0x0A	U+000D then U+000A
LF-CR		0x0A then 0x0D	U+000A then U+000D
NEL	New Line		U+0085
LS	Line Separator		U+2028
PS	Paragraph Separator		U+2029

Table 1 - END-OF-LINE CHARACTERS

Values other than LF are converted for calculation of the validation property and should be evaluated for interpretation by the endpoint systems. The validation property is not intended to resolve the meaning of the data, only the consistency.

5.2 Integer

This value is calculated identical to Text, and therefore should not be included in the attribute definition.

5.3 Double

For purposes of the validation property use the following format:

[-]x.xxxxxxe[-]xxx

Zero should simply be represented as 0.
The first non-zero digit should be to the left of the decimal.
Remove trailing zeros.
Remove a trailing decimal.
Remove leading zeros from the exponent.
When the exponent is zero remove the exponent portion altogether.
The negative symbol is only included when required, positive signs are always suppressed.

Validation Representation Examples: `-1e4, 1.43233e12, 1.278e-3, 1.2e1, 0, 3.4e1, 1.75`

Note: When the precision required is beyond the normal default precision of the programming language used to convert double values to strings such as CAD part position matrices, the validation process may need to be defined separately and not included in the AHASH calculation. See ISO 10303 for CAD validation properties.

For representation in the XML follow the Double definition in section 3.2.5 of XML Schema Part 2: Datatypes Second Edition (https://www.w3.org/TR/xmlschema-2/).

XML Representation Examples: `-1e4, -1000, 123.56, -11.2786 1.2, 0, 0.0`

Format Name: Double

5.4 Date and Time Representations

Dates and times should be converted to ISO 8601 format for storage in the XML. Further, the Time and Date Time formats should be converted to UTC prior to calculation of the validation property.

The formats listed below use the following definitions:

YYYY = four-digit year
MM = two-digit month (eg 03=March)
DD = two-digit day of the month (01 through 31)
T = a set character indicating the start of the time element
hh = two digits of an hour (00 through 23, AM/PM not included)
mm = two digits of a minute (00 through 59)
ss = two digits of a second (00 through 59)
mmm = three digits of a millisecond (000 through 999)
TZD = time zone designator (Z or +hh:mm or -hh:mm), the + or - values indicate how far ahead or behind a time zone is from the UTC (Coordinated Universal Time) zone.
Z is the Time Zone Designation for UTC (Zulu).

5.4.1 Date

YYYY-MM-DD
Example: 2013-02-05 corresponds to February 5th, 2013

Format Name: UTCDate

5.4.2 Time (UTC)

hh:mm:ss.[mmm]TZD

Milliseconds may be dropped if not required in definition.
Example: 13:15:30Z corresponds to 1:15:30 pm UTC (Zulu).

Format Name: UTCTime

5.4.3 Date Time (UTC)

YYYY-MM-DDThh:mm:ss[.mmm]TZD

Milliseconds may be dropped if not required in definition.
2013-02-05T13:15:30Z corresponds to Feburary 5th, 2013 at 1:15:30pm UTC (Zulu)

Format Name: UTCDateTime

6 Hash Identification

In order to calculate and validate the hash values consistently certain information must be provided along with the hash signature.

6.1 Hash Algorithm

A hash value must be calculated using a standard cryptographic algorithm to generate a unique signature (or fingerprint) for a node. The hash value is not intended to encrypt the data, so the fact that the value is predictable is not relevant.

6.1.1 SHA-1

SHA-1 is used in calculations of the validation property examples in this document, although any industry standard algorithm that can generate a key to satisfy the requirements for uniqueness may be used. The standard must be clearly identified in the validation section such that the consumers of the data can regenerate the hash after loading to validate that the data is consistent with the intent of the archive (e.g `<AHash_Algorithm>SHA1</AHash_Algorithm>`).

All characters in the calculated hash value must be uppercase.

6.2 Hash Specification

The specification used to dictate the calculation of the hash value must also be included. This may be a reference to a particular revision of this document, or it may be a company specific document which provides the recipe for calculation. The company specification may simply reference this LOTAR specification with any required local modifications. Examples of this reference:

```
<AHash_Specification>LOTAR TS-9300-200-1_R2.2</AHash_Specification>
```

```
<AHash specification='LOTAR TS-9300-200-1_R2.2'>6D5DB54436A3F72CE2D3D9D4A6992FE6FC83E1EF</AHash>
```

7 Example Product Structure with XML

The following examples will use the structure identified in Figure 4.

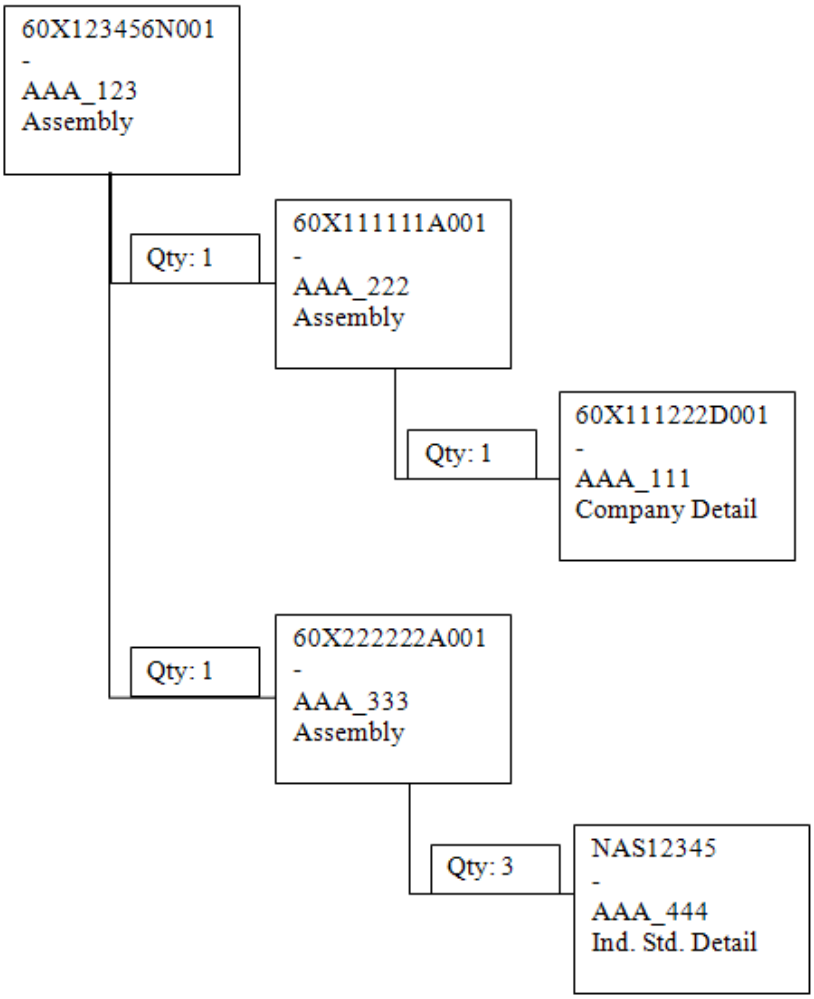

Figure 4 - XML EXAMPLE PRODUCT STRUCTURE

The XML examples shown below are for illustrative purposes only to show a possible method for the AHash definition. They are not intended to indicate the XML standard for LOTAR Part 210. Attributes that have special XML characters such as those shown in Table 2 will either need to be escaped appropriately in the XML or contained within CDATA tags. Regardless of the method, the value used to calculate the AHash should be the raw data and not the translated data.

Character Name	XML Translation	Character Reference	Numeric Reference
Ampersand	&amp;	&	&
Less Than	&lt;	<	<
Greater Than	&gt;	>	&#62;
Double Quote	&quot;	"	&#39;
Apostrophe	&apos;	'	&#34;

Table 2 - XML TRANSLATION FOR SPECIAL CHARACTERS

An example of wrapping an attribute in CDATA tags is as follows:

```
<Property name="Nomenclature">![CDATA[3" Washer]]</Property>
```

In the following examples, some of the attributes defined in the XML are mandatory attributes and are explicitly called out such as:

```
<PartID>AAA_111</PartID>
```

Other attributes are optional and defined such as:

```
<Property name="ProcessCodes" format="Text">AV, GM</Property>
```

This method allows the XSD to enforce the mandatory attributes while allowing greater flexibility with the other attributes. The data format definition for the mandatory attributes should be defined and enforced by the XSD whereas it is explicitly stated on the optional attributes.
The mandatory attributes shown in the examples below are for illustrative purposes only.

7.1 Detail Node Examples

Using the structure above, parts 60X111222D001 and NAS12345 are detail items meaning that they have no children. As described in section 4.2, the AHash values must be calculated in a bottom up fashion so that a parent assembly's AHash includes the hash value of its children. For a detail item, the CPAH (Current Part Attribute Hash) and the AHash are equivalent.

In this example, 60X111222D001 is defined as a "Company Detail" which means that the company owns the type design for this item. Items defined as "Company Detail" in the example have a different set of attributes used to calculate their AHash values than items that are "Industry Standard Detail" or "Assembly".

7.1.1 Company Detail

The XML for 60X111222D001 is as follows:

```
<Arch_Part>
  <CompanyDetail>
    <Properties>
      <CADFileName>AAA_111.CATPart</CADFileName>
      <CADFileType>CATPart</CADFileType>
      <Nomenclature>COMPANY DETAIL PART 1</Nomenclature>
      <PartID>AAA_111</PartID>
      <PartNumber>60X111222D01</PartNumber>
      <Revision>-</Revision>
      <Property name="CageCode" format="Text">12345</Property>
      <Property name="FastenerQty" format="Text">0</Property>
      <Property name="FinishCodes" format="Text">144, 213</Property>
      <Property name="MasterOfOpposite" format="Text"></Property>
      <Property name="Material" format="Text">AL ALLOY</Property>
      <Property name="PartDisposition" format="Text">60X111111D01,---, REWORK</Property>
      <Property name="ProcessCodes" format="Text">AV, GM</Property>
      <Property name="ReleaseDate" format="Date">2008-11-14</Property>
      <Property name="Status" format="Text">Released</Property>
    </Properties>
    <Validation>
      <AHashAttrib-
utes>CADFileName,CADFileType,CageCode,FastenerQty,FinishCodes,MasterOfOpposite,Material,N
omencla-
ture,PartDisposition,PartID,PartNumber,ReleaseDate,Revision,Status</AHashAttributes>
      <AHash_Algorithm>SHA1</AHash_Algorithm>
       <AHash_Specification>LOTAR TS-9300-200-1_R2.2</AHash_Specification>
      <AHash>6D5DB54436A3F72CE2D3D9D4A6992FE6FC83E1EF</AHash>
```

```
    </Validation>
  </CompanyDetail>
</Arch_Part>
```

The attributes used for the calculation of the AHash are concatenated based on the order specified in the AHashAttributes element. In this example, the attributes are defined alphabetically by attribute name but that is not required. Not all attributes defined in the XML need be part of the AHash calculation, there may be additional non-critical attributes used for informational purposes. The omission of the attribute in the AHashAttributes string as defined in section 4.1 indicates that it is not to be used for purposes of the AHash calculation.

Based on the attributes for this part and the AHashAttributes definition, the concatenated string for this part would be:

“AAA_111.CATPartCATPart123450144, 213AL ALLOYCOMPANY DETAIL PART 160X111111D01,---, REWORKAAA_11160X111222D01AV, GM2008-11-14-Released”

All relevant attributes are concatenated together into a single string which includes any embedded newlines. The newline in this example is simply a by-product of having to fit on this page rather than in the actual data. Creating a SHA1 hash of this string yields:

6D5DB54436A3F72CE2D3D9D4A6992FE6FC83E1EF

This has been converted to upper case by convention.

7.1.2 Industry Standard Detail

The XML for NAS12345 is as follows:

```
<Arch_Part>
  <IndustryStandardDetail>
    <Properties>
      <CADFileName>AAA_444.CATPart</CADFileName>
      <CADFileType>CATPart</CADFileType>
      <Nomenclature>THREADED SCREW</Nomenclature>
      <PartID>AAA_444</PartID>
      <PartNumber>NAS12345</PartNumber>
      <Revision>-</Revision>
      <Property name="CageCode" format="Text">54321</Property>
      <Property name="NonBOM" format="Text">0</Property>
      <Property name="ReleaseDate" format="Date">2008-01-22</Property>
      <Property name="Status" format="Text">Released</Property>
    </Properties>
    <Validation>
      <AHashAttrib-
utes>CADFileName,CADFileType,CageCode,Nomenclature,NonBOM,PartID,PartNumber,ReleaseDate,R
evision,Status</AHashAttributes>
      <AHash_Algorithm>SHA1</AHash_Algorithm>
       <AHash_Specification>LOTAR TS-9300-200-1_R2.2</AHash_Specification>
      <AHash>2E648063EDD57A6A3F51EF89EF0D6D4D11B2C3D9</AHash>
    </Validation>
  </IndustryStandardDetail>
</Arch_Part>
```

Based on the attributes for this part and the AHashAttributes definition, the concatenated string for this part would be:

"AAA_444.CATPartCATPart54321THREADED SCREW0AAA_444NAS123452008-01-22-Released"

Creating a SHA1 hash of this string yields:

2E648063EDD57A6A3F51EF89EF0D6D4D11B2C3D9

Note that there are fewer attributes defined in the AHashAttributes for this type of part than there are for the Company Detail Part. The capability to define which attributes should be considered by part type allows for greater flexibility within the system. This can be done by enforcement of mandatory attributes within a part class by the XSD or by modifying the set of optional attributes and the decision to include or exclude them from the AHash calculation.

7.2 Assembly Examples

As discussed in section 4.2, the AHash of an assembly is the CPAH of the assembly concatenated with a unique key for each distinct direct child part and its quantity In this example, Child_ID and Child_Rev combine to uniquely identify a part object. As discussed in Section 4.2, any combination of attributes that uniquely identify the object may be used. The following formula is used in the subsequent examples:

Assembly AHash = CPAH:Child1_ID:Child1_Rev:Child1_Qty:Child2_ID:Child2_Rev:Child2_Qty

The CPAH is the hash value of the attributes of the assembly itself.
The child information in the string should appear in alphanumeric order by child value. This not a requirement of the SHA1 algorithm, but a necessity for validation purposes as there must be a known method to rebuild the string.

To enhance readability for the Assembly AHash a delimiter is included in the string that is used to calculate the final hash value.

7.2.1 Assembly Examples with a Single Child

The XML for 60X111111A001 is as follows:

```
<Arch_Part>
  <Assembly>
    <Properties>
      <CADFileName>AAA_222.CATProduct</CADFileName>
      <CADFileType>CATProduct</CADFileType>
      <Nomenclature>SUB ASSEMBLY_1</Nomenclature>
      <PartID>AAA_222</PartID>
      <PartNumber>60X111111A001</PartNumber>
      <Revision>-</Revision>
      <Property name="CageCode"format="Text">12345</Property>
      <Property name="FinishCodes" format="Text"></Property>
      <Property name="PartDisposition" format="Text"></Property>
      <Property name="ProcessCodes" format="Text"></Property>
      <Property name="ReleaseDate" format="Date">2008-11-14</Property>
      <Property name="Status" format="Text">Released</Property>
    </Properties>
    <Validation>
      <AHashAttrib-
utes>CADFileName,CADFileType,CageCode,FinishCodes,Nomenclature,PartDisposition,PartID,Par
tNumber,ProcessCodes,ReleaseDate,Revision,Status</AHashAttributes>
      <AHash_Algorithm>SHA1</AHash_Algorithm>
       <AHash_Specification>LOTAR TS-9300-200-1_R2.2</AHash_Specification>
```

```
      <AHash> DE8D54C8CFE892ACA486929F20BC7EA7E16144D4</AHash>
    </Validation>
    <CAD_Children>
      <Child>
        <ChildID>AAA_111</ChildID>
        <ChildRevision>-</ChildRevision>
        <ChildQty>1</ChildQty>
      </Child>
    </CAD_Children>
  </Assembly>
</Arch_Part>
```

The concatenated string to generate the CPAH for this assembly based on the values defined in the AHashAttributes is:

“AAA_222.CATProductCATProduct12345SUB ASSEMBLY_1AAA_22260X111111A0012008-11-14-Released”

That string yields a SHA1 value of: E8535916412FCE0931F632D10E33E038F04578EE

To calculate the complete AHash of this assembly, the unique key and quantity of its children must be combined with its CPAH, which produces the following string:

“E8535916412FCE0931F632D10E33E038F04578EE:AAA_111:-:1”

The SHA1 value of the string above is: DE8D54C8CFE892ACA486929F20BC7EA7E16144D4

This becomes the AHash value of this assembly.

The XML for 60X222222A001 is as follows:

```
<Arch_Part>
  <Assembly>
    <Properties>
      <CADFileName>AAA_333.CATProduct</CADFileName>
      <CADFileType>CATProduct</CADFileType>
      <Nomenclature>SUB ASSEMBLY_2</Nomenclature>
      <PartID>AAA_333</PartID>
      <PartNumber>60X222222A001</PartNumber>
      <Revision>-</Revision>
      <Property name="CageCode" format="Text">12345</Property>
      <Property name="FinishCodes" format="Text"></Property>
      <Property name="PartDisposition" format="Text"></Property>
      <Property name="ProcessCodes" format="Text"></Property>
      <Property name="ReleaseDate" format="Date">2008-11-14</Property>
      <Property name="Status" format="Text">Released</Property>
    </Properties>
    <Validation>
      <AHashAttrib-
utes>CADFileName,CADFileType,CageCode,FinishCodes,Nomenclature,PartDisposition,PartID,Par
tNumber,ProcessCodes,ReleaseDate,Revision,Status</AHashAttributes>
      <AHash_Algorithm>SHA1</AHash_Algorithm>
       <AHash_Specification>LOTAR TS-9300-200-1_R2.2</AHash_Specification>
      <AHash>2FE358CA4EE477C53A8E9AE594A7E0B79AC283FF</AHash>
    </Validation>
    <CAD_Children>
      <Child>
        <ChildID>AAA_444</ChildID>
        <ChildRevision>-</ChildRevision>
        <ChildQty>3</ChildQty>
      </Child>
    </CAD_Children>
  </Assembly>
```

```
</Arch_Part>
```

The concatenated string to generate the CPAH for this assembly based on the values defined in the AHashAttributes is:

“AAA_333.CATProductCATProduct12345SUB ASSEMBLY_2AAA_33360X222222A0012008-11-14-Released”

That string yields a SHA1 value of: 8EECDBB17B821225AB7D79A0C61762514B029455

To calculate the complete AHash of this assembly, the unique key and quantity of its children must be combined with its CPAH, which produces the following string:

“8EECDBB17B821225AB7D79A0C61762514B029455:AAA_444:-:3”

The SHA1 value of the string above is: 2FE358CA4EE477C53A8E9AE594A7E0B79AC283FF

That becomes the stored AHash value of this assembly.

7.2.2 Assembly Example with Multiple Children

As mentioned above, the calculation method is not really different for multiple children with the exception that the child information in the string used to generate the AHash must come in alpha-numeric order based on the unique keys and quantities of the child parts.

The XML for the top assembly in our product structure example is as follows:

```
<Arch_Part>
  <Assembly>
    <Properties>
      <CADFileName>AAA_123.CATProduct</CADFileName>
      <CADFileType>CATProduct</CADFileType>
      <Nomenclature>TOP ASSEMBLY</Nomenclature>
      <PartID>AAA_123</PartID>
      <PartNumber>60X123456N001</PartNumber>
      <Revision>-</Revision>
      <Property name="CageCode" format="Text">12345</Property>
      <Property name="FinishCodes" format="Text"></Property>
      <Property name="PartDisposition" format="Text"></Property>
      <Property name="ProcessCodes" format="Text"></Property>
      <Property name="ReleaseDate" format="Date">2008-11-14</Property>
      <Property name="Status" format="Text">Released</Property>
    </Properties>
    <Validation>
      <AHashAttrib-
utes>CADFileName,CADFileType,CageCode,FinishCodes,Nomenclature,PartDisposition,PartID,Par
tNumber,ProcessCodes,ReleaseDate,Revision,Status</AHashAttributes>
      <AHash_Algorithm>SHA1</AHash_Algorithm>
       <AHash_Specification>LOTAR TS-9300-200-1_R2.2</AHash_Specification>
      <AHash>74E795F5F0E71A0CF538370A96C63D24025728C3</AHash>
    </Validation>
    <CAD_Children>
      <Child>
        <ChildID>AAA_222</ChildID>
        <ChildRevision>-</ChildRevision>
        <ChildQty>1</ChildQty>
      </Child>
      <Child>
        <ChildID>AAA_333</ChildID>
        <ChildRevision>-</ChildRevision>
```

```
        <ChildQty>1</ChildQty>
      </Child>
    </CAD_Children>
  </Assembly>
</Arch_Part>
```

The concatenated string to generate the CPAH for this assembly based on the values defined in the AHashAttributes is:

“AAA_123.CATProductCATProduct12345TOP ASSEMBLYAAA_12360X123456N0012008-11-14-Released”

That string yields a SHA1 value of:
2BFF3643CF930C0CCBB5F0CB17749FA93DDED79D

To calculate the complete AHash of this assembly, the unique key and quantity of its children must be combined with its CPAH, which produces the following string:

“2BFF3643CF930C0CCBB5F0CB17749FA93DDED79D:AAA_222:-:1:AAA_333:-:1”

The SHA1 value of the string above is:
74E795F5F0E71A0CF538370A96C63D24025728C3

This becomes the stored AHash value of this assembly.

Note that the child information in the string used to calculate the AHash are in alphanumeric order. The value “AAA_222” comes before “AAA_333” in the alphanumeric sort order.

7.2.3 Assembly Example with Child Positions

The examples up to this point assume the assemblies have physical files that provide the positional information for their children. Positional shifts of the children would be detected in a change to the hash value of the assembly product file. If however, the assemblies in the PDM system do not have a physical file representation then the positional information is required in the XML. In the latter case, the positional information should be calculated in the AHASH by some means. This can be implemented by generating an additional hash of the positional information for each child instance and combining it with the unique key. An alternative approach is to calculate a position hash for each child instance and storing that with each instance. These hash values must be validated on extract to ensure correct positioning.

An example showing the positional information for a child follows:

```
<Arch_Part>
  <Assembly>
    <Properties>
      <CADFileName>AAA_456.CATProduct</CADFileName>
      <CADFileType>CATProduct</CADFileType>
      <Nomenclature>TOP ASSEMBLY</Nomenclature>
      <PartID>AAA_456</PartID>
      <PartNumber>60X123456N003</PartNumber>
      <Revision>-</Revision>
```

```
      <Property name="CageCode" format="Text">12345</Property>
      <Property name="FinishCodes" format="Text"></Property>
      <Property name="PartDisposition"format="Text"></Property>
      <Property name="ProcessCodes" format="Text"></Property>
      <Property name="ReleaseDate" format="Date">2008-11-14</Property>
      <Property name="Status" format="Text">Released</Property>
    </Properties>
    <Validation>
      <AHashAttrib-
utes>CADFileName,CADFileType,CageCode,FinishCodes,Nomenclature,PartDisposition,PartID,Par
tNumber,ProcessCodes,ReleaseDate,Revision,Status</AHashAttributes>
      <AHash_Algorithm>SHA1</AHash_Algorithm>
       <AHash_Specification>LOTAR TS-9300-200-1_R2.2</AHash_Specification>
      <AHash>8DD0F3B74CC7CA9122BBC7B0F9D69B7A2A89BD66</AHash>
    </Validation>
    <CAD_Children>
      <Child>
        <ChildID>AAA_444</ChildID>
        <ChildRevision>-</ChildRevision>
        <ChildQty>2</ChildQty>
        <ChildInstances>
           <ChildInstance>
              <ChildInstanceID>AAA_444.1</ChildInstanceID>
              <RelativePosition>
                 <Rot00>1</Rot00>
                 <Rot01>0</Rot01>
                 <Rot02>0</Rot02>
                 <Rot10>0</Rot10>
                 <Rot11>1</Rot11>
                 <Rot12>0</Rot12>
                 <Rot20>0</Rot20>
                 <Rot21>0</Rot21>
                 <Rot22>1</Rot22>
                 <Tra0>5.00345</Tra0>
                 <Tra1>13</Tra1>
                 <Tra2>6</Tra2>
              </RelativePosition>
           </ChildInstance>
           <ChildInstance>
              <ChildInstanceID>AAA_444.2</ChildInstanceID>
              <RelativePosition>
                 <Rot00>1</Rot00>
                 <Rot01>0</Rot01>
                 <Rot02>0</Rot02>
```

```
                <Rot10>0</Rot10>
                <Rot11>1</Rot11>
                <Rot12>0</Rot12>
                <Rot20>0</Rot20>
                <Rot21>0</Rot21>
                <Rot22>1</Rot22>
                <Tra0>18.04555 </Tra0>
                <Tra1>13</Tra1>
                <Tra2>6</Tra2>
              </RelativePosition>
            </ChildInstance>
          </ChildInstances>
        </Child>
      </CAD_Children>
    </Assembly>
</Arch_Part>
```

In the example above, the CPAH is:

5C43B0C94D03917CD9E2ADFBF818C97A2D41BACF

To factor the child positions into the AHash the CPAH of the assembly is combined with the child instance ids and positional values strung together. Using this string:

5C43B0C94D03917CD9E2ADFBF818C97A2D41BACF:AAA_444.1:1.000:0:0:0:1:0:0:0:1:5.00034
5:13:6:AAA_444.2:1.000:0:0:0:1:0:0:0:1:1804555:13:6

The resultant SHA1 AHash value for the assembly is:
8DD0F3B74CC7CA9122BBC7B0F9D69B7A2A89BD66

7.2.4 MultiValuated Attribute Examples

In the example below, a property has been defined for General Notes which is stored as a single XML item that contains embedded XML. Since there are special XML characters embedded in the property text they must be converted as indicated as described in Table 2 in order to generate compliant XML.

```
<Arch_Part>
  <Assembly>
    <Properties>
      <CADFileName>AAA_555.CATProduct</CADFileName>
      <CADFileType>CATProduct</CADFileType>
      <Nomenclature>SUB ASSEMBLY_2</Nomenclature>
      <PartID>AAA_555</PartID>
      <PartNumber>60X222222A001</PartNumber>
      <Revision>-</Revision>
```

```
      <Property name="CageCode" format="Text">12345</Property>
      <Property name="FinishCodes" format="Text"></Property>
      <Property name="PartDisposition" format="Text"></Property>
      <Property name="ProcessCodes" format="Text"></Property>
      <Property name="ReleaseDate" format="Date">2008-11-14</Property>
      <Property name="Status" format="Text">Released</Property>
      <Property name="GeneralNotes" value="&lt;Notes name=&quot;GENERAL_NOTES&quot;&gt;&lt;Note name=&quot;G1&quot;&gt;FINISH AND PROCESS CODES IN ACCORDANCE WITH COMPANY STANDARDS UNLESS OTHERWISE SPECIFIED&lt;/Note&gt;&lt;Note name=&quot;G2&quot;&gt;DIMENSIONS AND TOLERANCES PER ASME Y14.5M-1994&lt;/Note&gt;&lt;Note name=&quot;G3&quot;&gt;UNLESS OTHERWISE SPECIFIED, TOLERANCES ARE IN INCHES&lt;/Note&gt;&lt;/Notes&gt;"/>
    </Properties>
    <Validation>
       <AHashAttributes>CADFileName,CADFileType,CageCode,FinishCodes,Nomenclature,PartDisposition,PartID,PartNumber,ProcessCodes,ReleaseDate,Revision,Status</AHashAttributes>
       <AHash_Algorithm>SHA1</AHash_Algorithm>
       <AHash_Specification>LOTAR TS-9300-200-1_R2.2</AHash_Specification>
       <AHash>C70DAA823711978AFB9B608F4B11643C62AF1DA4</AHash>
    </Validation>
    <CAD_Children>
      <Child>
        <ChildID>AAA_444</ChildID>
        <ChildRevision>-</ChildRevision>
        <ChildQty>3</ChildQty>
      </Child>
    </CAD_Children>
  </Assembly>
</Arch_Part>
```

The example below shows an alternate method of representing the same object wherein each multivaluated attribute record is represented as an individual property.

```
<Arch_Part>
  <Assembly>
    <Properties>
      <CADFileName>AAA_555.CATProduct</CADFileName>
      <CADFileType>CATProduct</CADFileType>
      <Nomenclature>SUB ASSEMBLY_2</Nomenclature>
      <PartID>AAA_555</PartID>
      <PartNumber>60X222222A001</PartNumber>
      <Revision>-</Revision>
      <Property name="CageCode" format="Text">12345</Property>
      <Property name="FinishCodes" format="Text"></Property>
      <Property name="PartDisposition" format="Text"></Property>
```

```
      <Property name="ProcessCodes" format="Text"></Property>
      <Property name="ReleaseDate" format="Date">2008-11-14</Property>
      <Property name="Status" format="Text">Released</Property>
      <Property name="GeneralNote" note_number="G1" format="Text">FINISH AND PROCESS
CODES IN ACCORDANCE WITH COMPANY STANDARDS UNLESS OTHERWISE SPECIFIED</Property>
      <Property name="GeneralNote" note_number="G2" format="Text">DIMENSIONS AND TOLER-
ANCES PER ASME Y14.5M-1994</Property>
      <Property name="GeneralNote" note_number="G3" format="Text">UNLESS OTHERWISE SPECI-
FIED, TOLERANCES ARE IN INCHES</Property>
    </Properties>
    <Validation>
       <AHashAttrib-
utes>CADFileName,CADFileType,CageCode,FinishCodes,Nomenclature,PartDisposition,PartID,Par
tNumber,ProcessCodes,ReleaseDate,Revision,Status</AHashAttributes>
       <AHash_Algorithm>SHA1</AHash_Algorithm>
       <AHash_Specification>LOTAR TS-9300-200-1_R2.2</AHash_Specification>
       <AHash>C70DAA823711978AFB9B608F4B11643C62AF1DA4</AHash>
    </Validation>
    <CAD_Children>
      <Child>
        <ChildID>AAA_444</ChildID>
        <ChildRevision>-</ChildRevision>
        <ChildQty>3</ChildQty>
      </Child>
    </CAD_Children>
  </Assembly>
</Arch_Part>
```