LOTAR

LONG TERM ARCHIVING AND RETRIEVAL

# TECHNICAL SPECIFICATION

## "PRODUCT STRUCTURE VALIDATION"

***Release 1.2***

**2013-10-28**

Status: Released


| **LOTAR** | |
|---|---|
| Jochen Boy<br>PROSTEP AG<br>jochen.boy@prostep.com | Jean-Yves Delaunay<br>Airbus<br>jean-yves.delaunay@airbus.com<br><br>Rick Zuray<br>The Boeing Company<br>richard.s.zuray@boeing.com |
| **Technical** | |
| Dan Ganser<br>Gulfstream Aerospace<br>dan.ganser@gulfstream.com | Heinrich Byzio<br>Cassidian<br>heinrich.byzio@cassidian.com<br><br>Cecil New<br>GE<br>cecil.new@ge.com |

# Contents

## Table of Contents

1 Introduction ..... 1
1.1 LOTAR ..... 1
1.2 Overview ..... 1
1.3 Scope of this document ..... 1
1.4 Out Scope of this document ..... 1
1.5 Definition of terms ..... 2
1.5.1 Detail Node ..... 2
1.5.2 Assembly Node ..... 2
1.5.3 Hash Value ..... 2
1.5.4 AHash Attributes ..... 2
1.5.5 CPAH ..... 2
1.5.6 AHash Value ..... 2
2 References ..... 2
3 Node Unique Definition ..... 2
3.1 Detail Node ..... 3
3.2 Assembly Node ..... 3
4 Attribute Validation Properties ..... 4
4.1 AHashAttributes ..... 4
4.2 AHash ..... 4
5 Data Definition ..... 5
5.1 Text ..... 5
5.1.1 End-Of-Line (CR-LF, LF-CR, LF) ..... 5
5.2 Integer ..... 6
5.3 Float ..... 6
5.4 Date ..... 6
5.5 Time (UTC) ..... 6
5.6 Date Time (UTC) ..... 6
6 Hash Algorithm ..... 7
6.1 SHA-1 ..... 7
7 Example Product Structure with XML ..... 7
7.1 Detail Item Examples ..... 8
7.1.1 Company Detail ..... 9
7.1.2 Industry Standard Detail ..... 10
7.2 Assembly Examples ..... 10

7.2.1 Assembly Examples with a Single Child ..... 11
7.2.2 Assembly Example with Multiple Children ..... 12

## List of Figures

Figure 1 - SAMPLE PRODUCT STRUCTURE ..... 3
Figure 2 - EXAMPLE PART VALIDATION XML FORMAT ..... 4
Figure 3 - ASSEMBLY HASH FORMULA ..... 5
Figure 4 - XML EXAMPLE PRODUCT STRUCTURE ..... 7

## List of Tables

Table 1 - END-OF-LINE CHARACTERS ..... 5
Table 2 - XML TRANSLATION FOR SPECIAL CHARACTERS ..... 8

## 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 |

# 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.

## 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.

### 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 AHash values 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

# 3 Node Unique Definition

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

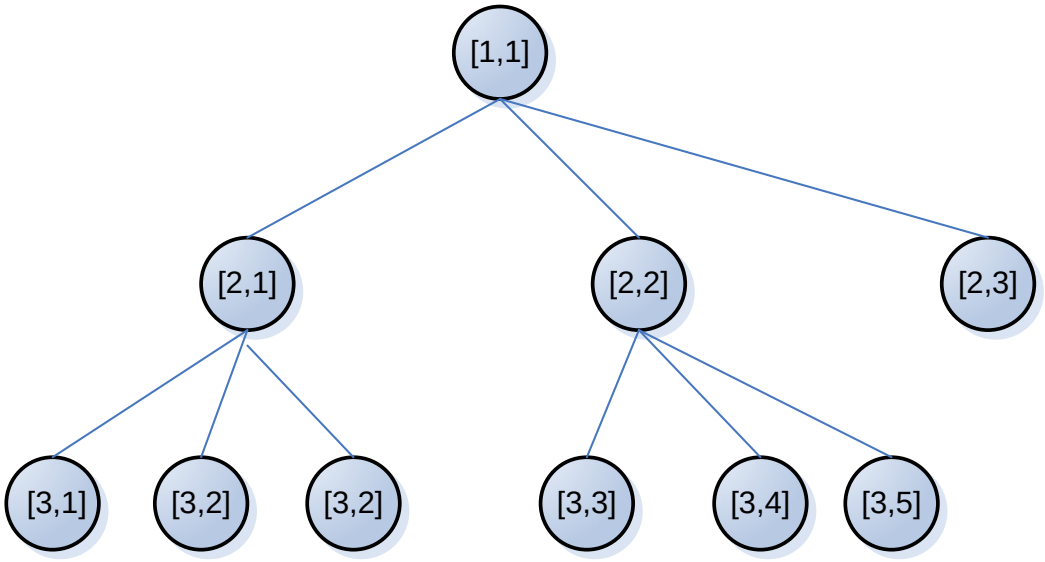


**Figure 1 - SAMPLE PRODUCT STRUCTURE**

## 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.

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 or node identifier must be defined. Typically this includes all critical attributes for the function that the data is being archived.

The included attributes are concatenated in alphabetical order and the hash value calculated from that string. The order of the AHash attributes may be defined in the XML however, it is recommended that they be in alphabetic order. This way, if the data is loaded into another PDM system in the future, it can be re-extracted and validated without that PDM system having to maintain the original order of the AHash attributes.

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>
</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 may be defined within the validation section as shown in Figure 2 or identified with the definition of each individual attribute as shown in the examples in section 7.
If the AHashAttributes are defined in the validation section as shown in Figure 2, each attribute is assumed to be a string (Text) by default, but may have a format included after a “double colon” in the string. The defined formats are listed in the “Data Definition” section of this document.
If the AHash attributes are defined as part of the other attributes then the formats may be specified within the attribute definition as shown in section 7.

## 4.2 AHash

AHash is the hash value of the node based on the attributes defined by the AHashAttributes value and its direct children and their hash values.

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 ordered in alphanumeric order by attribute name.
The AHash of a detail node is the CPAH of the object.

**Step 2:** Assembly AHash

The AHash of an assembly is the CPAH of the assembly concatenated with the quantity and AHash of each of its distinct and direct children. Each value is separated by a colon.

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

```
CPAH:qty1:AHASH1:qty2:AHASH2

qty# is a number.
```

**Figure 3 - ASSEMBLY HASH FORMULA**

# 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)

ISO specifies in ASCII that you can use either CR-LF or just LF. The most commonly used End Of Line is CR-LF, so it is recommended here to standardize on CR-LF.

There is a wide variety of opinions on the appropriate methodology for defining the end of line and therefore many different implementations. In order to generate, regenerate, and validate the data consistently a consistent format should be used to represent the end of line character(s).

Any use of the following encodings in data should be converted to CR-LF before calculating the validation property.

| Type | Description | ASCII HEX | 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**

CR or LF found independently should be replaced by CR-LF in the text.

These other values 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 Float

Format per IEEE-754-1985 using 64 Bit formatting.

+/- x.xxxxxxxxxxxxxxx e +/- xxx

2.2250738585072014e-308 (Smallest Positive Number)
1.7976931348623157e308 (Largest Positive Number)

All trailing zeros should be included in the calculation.

Format Name: Float

## 5.4 Date

Dates should be converted to ISO 8601 format for calculations: YYYY-MM-DD
Example: 2013-02-05 corresponds to February 5th, 2013

## 5.5 Time (UTC)

Times should be converted to ISO 8601 format for calculations: hh:mm:ssZ

Seconds or minutes may be dropped if not required in definition: hh:mm or hh
Example: 13:15:30Z corresponds to 1:15:30 pm UTC (Zulu).

Format Name: UTCTime

## 5.6 Date Time (UTC)

Dates with times should be converted to ISO-8601 format for calculations: YYYY-MM-DDThh:mm:ssTZD

Seconds or minutes may be dropped if not required in definition: hh:mm or hh.
Z is the Time Zone Designation for UTC (Zulu).
2013-02-05T13:15:30Z corresponds to Feburary 5th, 2013 at 1:15:30pm UTC (Zulu)

Format Name: UTCDateTime

# 6 Hash Algorithm

A hash value must be calculated using a standard cryptographic algorithm to generate a unique signature (or finger print) 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 SHA-1

SHA-1 is used in calculations of this validation property.

All characters in the calculated hash value must be uppercase.

# 7 Example Product Structure with XML

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

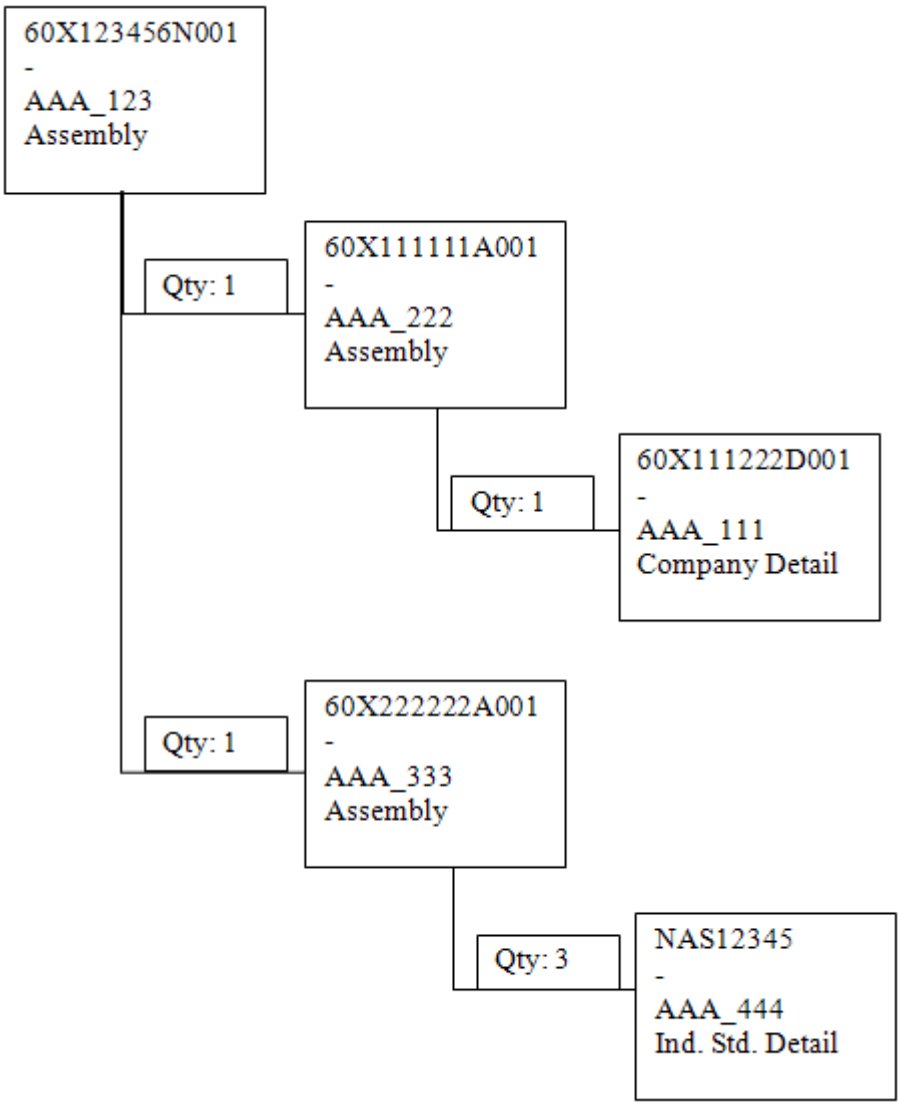


**Figure 4 - XML EXAMPLE PRODUCT STRUCTURE**

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 Tha | &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" format="Text">![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 ahash_rank="10">AAA_111</PartID>
```

Other attributes are optional and defined such as:

```
<Property name="ProcessCodes" ahash_rank="12" 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 Item 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 ahash_rank="1">AAA_111.CATPart</CADFileName>
      <CADFileType ahash_rank="2">CATPart</CADFileType>
      <Nomenclature ahash_rank="8">COMPANY DETAIL PART 1</Nomenclature>
      <PartID ahash_rank="10">AAA_111</PartID>
      <PartNumber ahash_rank="11">60X111222D01</PartNumber>
      <Revision ahash_rank="14">-</Revision>
      <Property name="CageCode" ahash_rank="3" format="Text">12345</Property>
      <Property name="FastenerQty" ahash_rank="4" format="Text">0</Property>
      <Property name="FinishCodes" ahash_rank="5" format="Text">144, 213</Property>
      <Property name="MasterOfOpposite" ahash_rank="6" format="Text"></Property>
      <Property name="Material" ahash_rank="7" format="Text">AL ALLOY</Property>
      <Property  name="PartDisposition"  ahash_rank="9"  format="Text">60X111111D01,---,
REWORK</Property>
      <Property name="ProcessCodes" ahash_rank="12" format="Text">AV, GM</Property>
      <Property name="ReleaseDate" ahash_rank="13" format="Date">2008-11-14</Property>
      <Property name="Status" ahash_rank="15" format="Text">Released</Property>
    </Properties>
    <Validation>
      <AHash>6D5DB54436A3F72CE2D3D9D4A6992FE6FC83E1EF</AHash>
    </Validation>
  </CompanyDetail>
</Arch_Part>
```

The attributes used for the calculation of the AHash have an ahash_rank specified in their definition which indicates the order in which those attributes should be concatenated. In this example, the ahash_rank orders the attributes alphabetically by attribute name as suggested in section in section 4.1. 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 ahash_rank in the definition of the attribute implies 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 ahash_rank="1">AAA_444.CATPart</CADFileName>
      <CADFileType ahash_rank="2">CATPart</CADFileType>
      <Nomenclature ahash_rank="4">THREADED SCREW</Nomenclature>
      <PartID ahash_rank="6">AAA_444</PartID>
      <PartNumber ahash_rank="7">NAS12345</PartNumber>
      <Revision ahash_rank="9">-</Revision>
      <Property name="CageCode" ahash_rank="3" format="Text">54321</Property>
      <Property name="NonBOM" ahash_rank="5" format="Text">0</Property>
      <Property name="ReleaseDate" ahash_rank="8" format="Date">2008-01-22</Property>
      <Property name="Status" ahash_rank="10" format="Text">Released</Property>
    </Properties>
    <Validation>
      <AHash>77313EACB1C61E926B5376404DE77D78EA3820A3</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_444NAS123452008-1-22-Released”

Creating a SHA1 hash of this string yields:

77313EACB1C61E926B5376404DE77D78EA3820A3

**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 for an assembly is comprised of the hash value of its relevant attributes combined with the quantities and hash values of its children using a single colon as a field separator. This is done using the following formula:

Assembly AHash = CPAH:Child1 Qty:Child1 AHash:Child2 Qty:Child2 AHash:Childn Qty:Childn AHash

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 AHash 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.

### 7.2.1 Assembly Examples with a Single Child

The method used to calculate the AHash information is not different when the assembly has only 1 child, but it is broken out that way here to conform to the product structure example in Figure 2.

The XML for 60X111111A001 is as follows:

```
<Arch_Part>
  <Assembly>
    <Properties>
      <CADFileName ahash_rank="1">AAA_222.CATProduct</CADFileName>
      <CADFileType ahash_rank="2">CATProduct</CADFileType>
      <Nomenclature ahash_rank="5">SUB ASSEMBLY_1</Nomenclature>
      <PartID ahash_rank="7">AAA_222</PartID>
      <PartNumber ahash_rank="8">60X111111A001</PartNumber>
      <Revision ahash_rank="11">-</Revision>
      <Property name="CageCode" ahash_rank="3" format="Text">12345</Property>
      <Property name="FinishCodes" ahash_rank="4" format="Text"></Property>
      <Property name="PartDisposition" ahash_rank="6" format="Text"></Property>
      <Property name="ProcessCodes" ahash_rank="9" format="Text"></Property>
      <Property name="ReleaseDate" ahash_rank="10" format="Date">2008-11-14</Property>
      <Property name="Status" ahash_rank="12" format="Text">Released</Property>
    </Properties>
    <Validation>
       <AHash>69078E85494C5CBC5B92A089254934CFC30407F7</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 quantity and AHash of its child must be combined with its CPAH, which produces the following string:

“E8535916412FCE0931F632D10E33E038F04578EE:1:6D5DB54436A3F72CE2D3D9D4A6992FE6FC83E1EF”

The SHA1 value of the string above is:  69078E85494C5CBC5B92A089254934CFC30407F7

Which is what is stored as the AHash value of this assembly.

The XML for 60X222222A001 is as follows:

```
<Arch_Part>
  <Assembly>
    <Properties>
      <CADFileName ahash_rank="1">AAA_333.CATProduct</CADFileName>
      <CADFileType ahash_rank="2">CATProduct</CADFileType>
      <Nomenclature ahash_rank="5">SUB ASSEMBLY_2</Nomenclature>
      <PartID ahash_rank="7">AAA_333</PartID>
      <PartNumber ahash_rank="8">60X222222A001</PartNumber>
      <Revision ahash_rank="11">-</Revision>
      <Property name="CageCode" ahash_rank="3" format="Text">12345</Property>
      <Property name="FinishCodes" ahash_rank="4" format="Text"></Property>
      <Property name="PartDisposition" ahash_rank="6" format="Text"></Property>
      <Property name="ProcessCodes" ahash_rank="9" format="Text"></Property>
      <Property name="ReleaseDate" ahash_rank="10" format="Date">2008-11-14</Property>
      <Property name="Status" ahash_rank="12" format="Text">Released</Property>
    </Properties>
    <Validation>
       <AHash>0222F6AF30666A9BD87002E713046E2903565433</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: 81E13DD2C774A6BD60C9F031AF3D61B9A28E0396

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

“81E13DD2C774A6BD60C9F031AF3D61B9A28E0396:3:77313EACB1C61E926B5376404DE77D78EA3820A3”

The SHA1 value of the string above is: 0222F6AF30666A9BD87002E713046E2903565433

Which is what is stored as the 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 AHash of the child parts.

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

```
<Arch_Part>
  <Assembly>
    <Properties>
      <CADFileName ahash_rank="1">AAA_123.CATProduct</CADFileName>
      <CADFileType ahash_rank="2">CATProduct</CADFileType>
      <Nomenclature ahash_rank="5">TOP ASSEMBLY</Nomenclature>
      <PartID ahash_rank="7">AAA_123</PartID>
      <PartNumber ahash_rank="8">60X123456N001</PartNumber>
      <Revision ahash_rank="11">-</Revision>
      <Property name="CageCode" ahash_rank="3" format="Text">12345</Property>
      <Property name="FinishCodes" ahash_rank="4" format="Text"></Property>
      <Property name="PartDisposition" ahash_rank="6" format="Text"></Property>
      <Property name="ProcessCodes" ahash_rank="9" format="Text"></Property>
      <Property name="ReleaseDate" ahash_rank="10" format="Date">2008-11-14</Property>
      <Property name="Status" ahash_rank="12" format="Text">Released</Property>
    </Properties>
    <Validation>
       <AHash>CC24DFE9A8789952405736508FCB2AE5415509A5</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 quantities and AHash values of its children must be combined with its CPAH which produces the following string:

“2BFF3643CF930C0CCBB5F0CB17749FA93DDED79D:1:81E13DD2C774A6BD60C9F031AF3D61B9A28E0396:1:E8535916412FCE0931F632D10E33E038F04578EE”

The SHA1 value of the string above is:
CC24DFE9A8789952405736508FCB2AE5415509A5

Which is what is stored as the AHash value of this assembly.

Note that the child information in the string used to calculate the AHash are in alphanumeric order. The value “81” comes before “E8” in the alphanumeric sort order. If you look at the AHash values for the 2 children in section 7.2.1 above you will see that the order of the hashes in this string is inverse to the order that parts are listed in the CAD_Children section of the XML.