# 

# **XML Canonicalization – Design Document**

**SOURCE DATA PROFILING**

# **Source File A Profile**

|  |  |
| --- | --- |
| **File Name** | Consumer\_Complaints\_FileA.xml |
| **Format** | Xml file |
| **MD5 Checksum** | 74cb9b3805d47cb43f34a9018fea1f79  Check sum has been calculated using md5 checksum utility in mac as shown below: |
| **Properties and Contents** | 1. File in question contains 8 consumer complaint records filed with a government agency , each relating to complaints made by customers against financial institutions, like banks and lending companies illustrated in an XML format. 2. From the information provided, it is also evident that this XML represents the data stored in an old complaint software system that this agency maintains. 3. After checking for root element, tag names matching, nesting and XML structure it can be said that the data is represented as a well-formed XML which is machine readable and human readable. 4. The XML cannot be considered a completely valid one as the & value in some elements need to be replaced by &amp; before it can be successfully processed by an XML parser. 5. File A XML does not have the XML version or encoding defined in the beginning of the XML . 6. There is no !DOCTYPE definition in the XML . Hence looking at the XML, the user cannot immediately relate it to any internal or external DTD and might need to come up with one. 7. The data appears to be structured in nature, and is furnished under a root element <consumerComplaints>. 8. Root element of the XML <consumerComplaints> has 8 occurrences of the child element <complaint> under it , indicating that details of 8 individual complaints are represented in the XML. 9. All occurrences of <complaint> has an attribute value ‘id’ which is most likely the unique complaint id for the particular complaint record. It is to be noted that the id value, although always numeric, does not have a pre-defined length and seems to contain 7 , 6 or even 5 digits at times. 10. XML tag names are not consistent throughout the XML. Example <complaint> is a single word tag name , all lower case where as some other tags like <consumerNarrative> have two words and a capitalized second word. 11. Each occurrence of the child element <complaint> has a maximum of 7 sub-elements under it, corresponding to different aspects of the complaint being represented by the XML. List of sub-elements - <event>,<product>, <issue>, <consumerNarrative>, <company>, <submitted>, <response> 12. Some of these sub-elements go deeper by one more level and have another set of sub-elements under them, but not further than that. Ex: <company> has more attributes like <companyName>, <companyZip> and <companyState> under it. Similarly <product> has <productType> and <subproduct> whereas <issue> has <issueType> and <subissue> under it. Similarly <response> has <publicResponse> and <responseType> nested under it. 13. Not all the sub-elements seem to be present on all the 8 complaint records, indicating some of these are not always REQUIRED and could be OPTIONAL. Ex: <consumerNarrative> , <publicResponse> etc. are not present on all complaints. 14. Some of these sub-elements seem to have attributes associated with them , providing additional information about the element, while some do not have associated attributes. Ex: <complaint> has ‘id’ as attribute. 15. Some of these sub-elements are empty in some cases while some others are not. Ex: <submitted> is an empty element with just an attribute ‘via’. 16. Some of these sub-elements seem to occur more than once. Ex: <event> 17. All the sub-elements under <complaint> do not appear in the same order on all instances of <complaint>. Ex: Under complaint id="14038", the last complaint in the XML, 2 <event> elements are present , but both are appearing wide apart unlike the case with other <complaint>s. Unlike all the other 7 complaints in the file, order of some other elements too is different in this particular instance of complaint and this warrants special handling in the DTD element declaration. 18. Upon scrutinizing the XML content, there are data level properties as well, which need be considered while constructing the DTD. Ex: Element <companyName> has the character ‘&’ in at least 2 instances which has to be handled during canonicalization 19. Apart from <complaint>, <response> is the only element that seems to have both attributes and sub-elements associated with it. 20. All the attributes in the XML seem to be REQUIRED attributes. There are no IMPLIED attributes in the XML. |

More details related to elements, entities and attributes gathered based on detailed analysis are given below. This step has also been very useful in the creation of the DTDs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Element/Attribute Name** | **Type** | **Expected number of occurrences** | **Possible Data Type** | **Parent Element** | **Child Elements** | **Attributes** | **Notes** |
| 1 | <consumerComplaints> | ELEMENT | 1 | NA | None | <complaint>  Attributes - none | None | This is the root element for the XML |
| 2 | <complaint> | ELEMENT | 1 or more | NA | <consumerComplaints> | <event>,<product>, <issue>, <consumerNarrative>, <company>, <submitted>, <response> | id | Multiple <complaint> elements under the root element. |
| 3 | id | ATTRIBUTE | 1 | NMTOKEN | <complaint> | NA | NA | Seems to be the unique identifier for each <complaint> |
| 4 | <event> | ELEMENT | 1 or more | NA | <complaint> | None | type , date | Element <event> seems to be an empty element although, there seem to be 2 occurrences on each <complaint>, each with type and date attribute values populated  Each seem to correspond to an event when the complaint was received on a certain date and forwarded on a certain date. |
| 5 | type | ATTRIBUTE | 1 | Enumerated List | <event> | NA | NA | Event ‘type’ attribute seems to have two values – ‘received’ and ‘sentToCompany’ – each present on all the complaints, on the respective <event> element |
| 6 | date | ATTRIBUTE | 1 | NMTOKEN | <event> | NA | NA | Event ‘date’ attribute seems to indicate the date when the <event> associated with the <complaint> happened. |
| 7 | <product> | ELEMENT | 1 | NA | <complaint> | <ProductType>, <subproduct> | None | <productType> seem to be present on all complaints whereas <subproduct> is not available on all individual complaints |
| 8 | <productType> | ELEMENT | 1 | PCDATA | <product> | None | None | Always present. Contains text data, might be a pre-defined set of product types. |
| 9 | <subproduct> | ELEMENT | 0 or 1 | PCDATA | <product> | None | None | Not always present. Contains text data, might be pre-defined set of sub product names. |
| 10 | <issue> | ELEMENT | 1 | NA | <complaint> | <IssueType>, <subissue> | None | <issueType> seem to be present on all complaints whereas <subissue> is not available on all individual complaints |
| 11 | <issueType> | ELEMENT | 1 | PCDATA | <issue> | None | None | Always present. Contains text data. At least one issueType is repeating across complaints. So it might be a pre-defined set of values |
| 12 | <subissue> | ELEMENT | 0 or 1 | PCDATA | <issue> | None | None | Not always present. Contains text data. Couple of values are repeating across complaints. Might be from a pre-defined set of sub issue types. |
| 13 | <consumerNarrative> | ELEMENT | 0 or 1 | PCDATA | <complaint> | None | None | Free form text field . Also, not present on all the complaints |
| 14 | <company> | ELEMENT | 1 | NA | <complaint> | <companyName>, <companyState>, <companyZip> | None | Element <company> and its child elements are present on all the <complaint> records in the XML |
| 15 | <companyName> | ELEMENT | 1 | PCDATA | <company> | None | None | Seems to have & in some cases and those need to be escaped |
| 16 | <companyState> | ELEMENT | 1 | PCDATA | <company> | None | None | Contains 2-character state code |
| 17 | <companyZip> | ELEMENT | 1 | PCDATA | <company> | None | None | 5-digit zip code |
| 18 | <submitted> | ELEMENT | 1 | NA | <complaint> | None | via | EMPTY element, always present on a complaint and with one attribute |
| 19 | via | ATTRIBUTE | 1 | Enumerated List | <submitted> | NA | NA | Could be populated from a pre-defined list of values – Web, Phone, Referral |
| 20 | <response> | ELEMENT | 1 | NA | <complaint> | <responseType>, <publicResponse> | timely, consumerDisputed | Element <response> has 2 attributes and 2 sub elements nested under it. |
| 21 | <responseType> | ELEMENT | 1 | PCDATA | <response> | None | None | Could be a value from a pre-defined list - Closed with explanation, Closed with monetary relief, Untimely response, Closed without relief. But since <responseType> is an element and not an attribute , enumerated list data type cannot be used |
| 22 | <publicResponse> | ELEMENT | 0 or 1 | PCDATA | <response> | None | None | Free form text field. Also, not present on all the <complaint>s |
| 23 | timely | ATTRIBUTE | 1 | Enumerated List | <response> | NA | NA | Could be a Yes or No indicator - Y or N |
| 24 | consumerDisputed | ATTRIBUTE | 1 | Enumerated List | <response> | NA | NA | Could be a Yes or No indicator - Y or N |

# **Source File B Profile**

|  |  |
| --- | --- |
| **File Name** | Consumer\_Complaints\_FileB.xml |
| **Format** | Xml file |
| **MD5 Checksum** | 6dfd2cf9686f6abf471ea56dc6d3c940  Check sum has been calculated using md5 checksum utility in mac as shown below: |
| **Properties and Contents** | 1. File in question contains 8 consumer complaint records filed with a government agency , each relating to complaints made by customers against financial institutions, like banks and lending companies illustrated in an XML format. 2. From the information provided and from the examination of data, it is also evident that this XML represents the data stored in a new complaint software system that this agency maintains and contains the details of the same 8 complaints from File A (old system). 3. After checking for root element, tag names matching, nesting and XML structure it can be said that the data is represented as a well-formed XML which is machine readable and human readable. 4. The XML cannot be considered a completely valid one as the & value in some elements need to be replaced by &amp; before it can be successfully processed by an XML parser. 5. File B XML does not have the XML version or encoding defined in the beginning of the XML . 6. There is no !DOCTYPE definition in the XML . Hence looking at the XML, the user cannot immediately relate it to any internal or external DTD and might need to come up with one. 7. The data appears to be structured in nature, and is furnished under a root element <consumerComplaints>. 8. Root element of the XML <consumerComplaints> has 8 occurrences of the child element <complaint> under it , indicating that details of 8 individual complaints are represented in the XML. 9. All occurrences of <complaint> has an attribute value ‘id’ which is most likely the unique complaint id for the particular complaint record. It is to be noted that the id value, although always numeric, does not have a pre-defined length and seems to contain 7 , 6 or even 5 digits at times. 10. Some occurrences of <complaint> has an additional attribute submissionType, which is not present in File A XML. 11. XML tag names are not consistent throughout the XML. Example <complaint> is a single word tag name , all lower case where as some other tags like <consumerNarrative> have two words and a capitalized second word. 12. Each occurrence of the child element <complaint> has a maximum of 7 sub-elements under it, corresponding to different aspects of the complaint being represented by the XML. List of sub-elements - <event>,<product>, <issue>, <consumerNarrative>, <company>, <submitted>, <response>. Element <submitted> is present only under one complaint record and even there remains EMPTY , unlike File A XML where it is present on all complaints. Corresponding values seem to be populated as an attribute ‘submissionType’ on each <complaint> element within File B XML. In File A XML these values are tied to an attribute ‘via’ although element content is EMPTY. 13. Some of these sub-elements go deeper by one more level and have another set of sub-elements under them, but not further than that. Ex: <company> has more attributes like <companyName>, <companyZip> and <companyState> under it. Similarly <product> has <productType> and <subproduct> whereas <issue> has <issueType> and <subissue> under it. Similarly <response> has <publicResponse> and <responseType> nested under it. 14. Not all the sub-elements seem to be present on all the 8 complaint records, indicating some of these are not always REQUIRED and could be OPTIONAL. Ex: <consumerNarrative> , <publicResponse> not present on all complaints. 15. Some of these sub-elements seem to have attributes associated with them , providing additional information about the element, while some do not have associated attributes. Ex: <complaint> has ‘id’ and ‘submissionType’ as attributes. 16. Some of these sub-elements are empty in some cases while some others are not. Ex: <event> is an empty element, but with 2 attributes – ‘type’ and ‘date’ 17. Order of the attributes ‘type’ and ‘date’ under the element <event> does not seem to be consistent always, unlike the case with File A XML. In File A it was always ‘date’ after ‘type’. 18. Some of the sub-elements seem to occur more than once under <complaint>. Ex: <event> 19. All the sub-elements under <complaint> do not appear in the same order on all instances of <complaint>. Ex: Under complaint id="14038", the last complaint in the XML, 2 <event> elements are present , but both are appearing wide apart unlike the case with other <complaint>s. Unlike all the other 7 complaints in the file, order of some other elements too is different in this particular instance of complaint and this warrant special handling in the DTD element declaration. 20. Upon scrutinizing the XML content, there are data level properties as well, which need be considered while constructing the DTD. Ex: Element <companyName> has the character ‘&’ in at least 2 instances which might need to be handled as XML parser will not be able to process it. 21. Apart from <complaint>, <response> is the only element that seems to have both attributes and sub-elements associated with it. 22. There seems to be a comment in the XML file corresponding to an update done to an <event> element on one of the complaints (id – 837784) - *<!-- Note: Sally modified this event on 2014-05-06 -->* 23. Attributes ‘timely’ and ‘consumerDisputed’ under element <response> seem to hold one character ‘Y‘ or ‘N’ Boolean values in File A XML. But in File B XML, ‘timely’ attribute seems to hold whole word Boolean indicators ‘yes’ or ‘no’ instead of ‘Y‘ or ‘N’ 24. Attributes ‘timely’ and ‘consumerDisputed’ appear in reverse order in File B, when compared to File A 25. Unlike File A, attribute’ timely’ doesn’t always appear on a complaint in File B XML and thus becomes an IMPLIED attribute 26. Content “sentToCompany “ of the <event> element’s attribute ‘type’ has whitespace within the quotes for complaint id 2327502 in File B XML. There is no such whitespace in File A. |

More details related to elements, entities and attributes gathered based on detailed analysis are given below. This step has also been very useful in the creation of the DTDs

| **#** | **Element/Attribute Name** | **Type** | **Expected number of occurrences** | **Possible Data Type** | **Parent Element** | **Child Elements** | **Attributes** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | <consumerComplaints> | ELEMENT | 1 | NA | None | <complaint>  Attributes - none | None | This is the root element for the XML |
| 2 | <complaint> | ELEMENT | 1 or more | NA | <consumerComplaints> | <event>,<product>, <issue>, <consumerNarrative>, <company>, <response> | Id, submissionType | Multiple <complaint> elements under the root element. |
| 3 | id | ATTRIBUTE | 1 | NMTOKEN | <complaint> | NA | NA | Seems to be the unique identifier for each <complaint> |
| 4 | submissionType | ATTRIBUTE | 0 or 1 | Enumerated List | <complaint> | NA | NA | Not always present on a complaint and seems to be from a pre-defined list - Web, Phone, Referral |
| 5 | <event> | ELEMENT | 1 or more | NA | <complaint> | None | type , date | Element <event> seems to be an empty element although, there seem to be 2 occurrences on each <complaint>, each with type and date attribute values populated  Each seem to correspond to an event when the complaint was received on a certain date and forwarded on a certain date. |
| 6 | type | ATTRIBUTE | 1 | Enumerated List | <event> | NA | NA | Event ‘type’ attribute seems to have two values – ‘received’ and ‘sentToCompany’ – each present on all the complaints, on the respective <event> element |
| 7 | date | ATTRIBUTE | 1 | NMTOKEN | <event> | NA | NA | Event ‘date’ attribute seems to indicate the date when the <event> associated with the <complaint> happened. |
| 8 | <product> | ELEMENT | 1 | NA | <complaint> | <ProductType>, <subproduct> | None | <productType> seem to be present on all complaints whereas <subproduct> is not available on all individual complaints |
| 9 | <productType> | ELEMENT | 1 | PCDATA | <product> | None | None | Always present. Contains text data, might be a pre-defined set of product types. |
| 10 | <subproduct> | ELEMENT | 0 or 1 | PCDATA | <product> | None | None | Not always present. Contains text data, might be pre-defined set of sub product names. |
| 11 | <issue> | ELEMENT | 1 | NA | <complaint> | <IssueType>, <subissue> | None | <issueType> seem to be present on all complaints whereas <subissue> is not available on all individual complaints |
| 12 | <issueType> | ELEMENT | 1 | PCDATA | <issue> | None | None | Always present. Contains text data. At least one issueType is repeating across complaints. So it might be a pre-defined set of values |
| 13 | <subissue> | ELEMENT | 0 or 1 | PCDATA | <issue> | None | None | Not always present. Contains text data. Couple of values are repeating across complaints. Might be from a pre-defined set of sub issue types. |
| 14 | <consumerNarrative> | ELEMENT | 0 or 1 | PCDATA | <complaint> | None | None | Free form text field . Also, not present on all the complaints |
| 15 | <company> | ELEMENT | 1 | NA | <complaint> | <companyName>, <companyState>, <companyZip> | None | Element <company> and its child elements are present on all the <complaint> records in the XML |
| 16 | <companyName> | ELEMENT | 1 | PCDATA | <company> | None | None | Seems to have & in some cases which might need to be escaped |
| 17 | <companyState> | ELEMENT | 1 | PCDATA | <company> | None | None | Contains 2-character state code |
| 18 | <companyZip> | ELEMENT | 1 | PCDATA | <company> | None | None | 5-digit zip code |
| 19 | <submitted> | ELEMENT | 0 or 1 | EMPTY | <complaint> | None | None | Present only on one complaint. Not having any content within. |
| 20 | <response> | ELEMENT | 1 | NA | <complaint> | <responseType>, <publicResponse> | timely, consumerDisputed | Element <response> has 2 attributes and 2 sub elements nested under it. |
| 21 | <responseType> | ELEMENT | 1 | PCDATA | <response> | None | None | Could be a value from a pre-defined list - Closed with explanation, Closed with monetary relief, Untimely response, Closed without relief |
| 22 | <publicResponse> | ELEMENT | 0 or 1 | PCDATA | <response> | None | None | Free form text field . Also, not present on all the <complaint>s |
| 23 | timely | ATTRIBUTE | 0 or 1 | Enumerated List | <response> | NA | NA | Could be Boolean indicator values ‘yes’ or ‘no’, not always present. |
| 24 | consumerDisputed | ATTRIBUTE | 1 | Enumerated List | <response> | NA | NA | Could be a Yes or No indicator - Y or N |

# **Canonicalization - STEPS**

Below are the 3 main steps followed in the process of canonicalization of the provided File A and File B XMLs corresponding to old and new systems respectively.

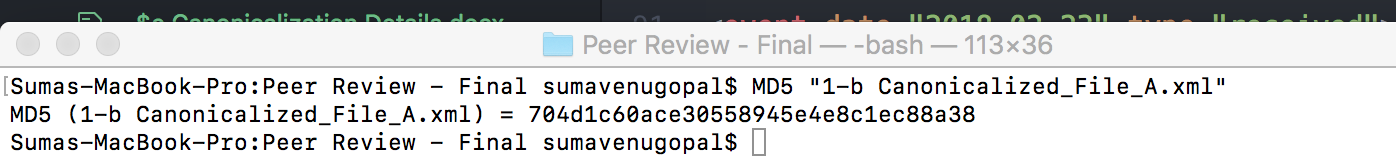
1. Created modified File A XML after applying C14N Canonicalization steps
2. Created modified File B XML after applying C14N Canonicalization steps
3. Applied additional data curation steps to create a final , complete canonical File XML with no data or information loss.

Details of each of these steps are given below:

**Steps to Create Modified File A XML After Applying C14N Canonicalization**

Followed the C14N algorithm-based canonicalization steps from the link - <https://www.w3.org/TR/xml-c14n/>

1. Ensured that the XML document is in [UTF-8](https://www.w3.org/TR/xml-c14n/#UTF-8) format by selecting UTF-8 formatting while creating the file on desktop. Screenshot is provided below. Text editor used is ATOM.
2. Ensured that the Line breaks are normalized. No additional step was needed to achieve this as the XML provided when opened in Mac OS with the text editor ATOM seemed to already have LF line endings indicating line breaks are already normalized.
3. Checked for white spaces outside the XML root element and within start and end tags so that they could be normalized. But this was not applicable for File A XML.
4. Checked to see if the XML declaration part in the XML specifying encoding and version had to be removed. Such details are not present in File A XML and hence this step was not needed.
5. Checked to see if the XML has any DTD or a reference to a DTD so that that it could be removed. Such details are not present in File A XML and hence this step was not needed.
6. Made sure that lexicographic order is imposed on the attribute order for each element. Order of the attributes of <response> and <event> was changed in this step.
7. Also made sure that attribute values are devoid of any unnecessary white spaces.
8. Checked to make sure that attribute values were enclosed within double quotes.
9. Checked to see if there was a need to add any default attributes to elements. This step too was not applicable.
10. Checked if any character or parsed entity references had to be replaced. No entity references seem to be present and this step was not applicable for File A
11. Checked to see if any special characters had to be replaced with character reference. Could not find any.
12. Checked if any CDATA had to be replaced with character content and this too was not applicable as there seem to be no CDATA content.
13. Added ending tags to some of the empty elements which were missing those – Ex: <event>,< submitted>
14. Checked to see if there is any white space within content that had to be retained, but this was not applicable for File A
15. Checked if any superfluous namespace declarations had to be removed , but this step too was not applicable for the XML as there are no namespaces URIs used.
16. Once all the steps above are completed, calculated MD5 Checksum **-** 704d1c60ace30558945e4e8c1ec88a38

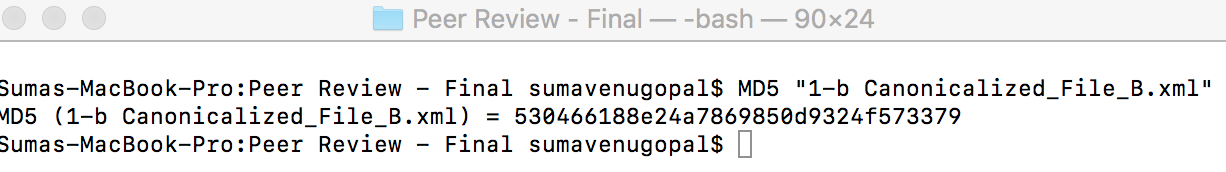
****

**Steps to Create Modified File B XML After Applying C14N Canonicalization**

Followed C14N algorithm-based canonicalization steps from the link - <https://www.w3.org/TR/xml-c14n/>.

C14N can be implemented by retaining comments as well, hence decided to retain the single comment present on File B XML to support provenance and ensure data preservation.

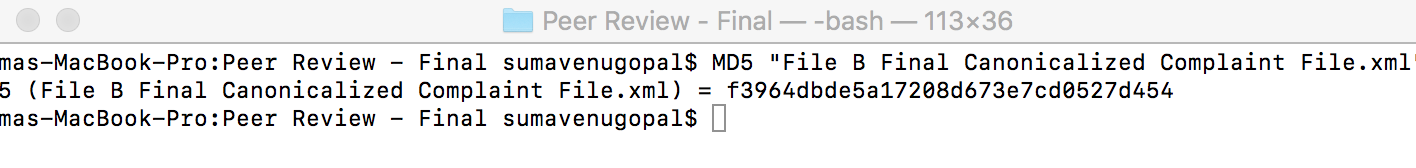
1. Ensured that the XML document is in [UTF-8](https://www.w3.org/TR/xml-c14n/#UTF-8) format by selecting UTF-8 formatting while creating the file on desktop. Screenshot is provided below. Text editor used is ATOM.
2. Ensured that the Line breaks are normalized. No additional step was needed to achieve this as the XML provided when opened in Mac OS with the text editor ATOM seemed to already have LF line endings indicating line breaks are already normalized.
3. Checked for white spaces outside the XML root element and within start and end tags so that they could be normalized. But this was not applicable for File B XML.
4. Checked to see if the XML declaration part in the XML specifying encoding and version had to be removed. Such details are not present in File B XML and hence this step was not needed.
5. Checked to see if the XML has any DTD or a reference to a DTD so that that it could be removed. Although it is mentioned in the instructions that File B has minimal internal DTD, could not locate any. Concluded that DTD details are not present in File B XML and hence this step is not needed.
6. Made sure that lexicographic order is imposed on the attribute order for each element. Order of the attributes of <response> and <event> was changed in this step.
7. Also made sure that attribute values are devoid of any unnecessary white spaces. Normalized ‘type’ attribute value in some cases as part of this step by removing unwanted white space before the closing quotes.
8. Checked to make sure that attribute values were enclosed within double quotes.
9. Checked to see if there was a need to add any default attributes to elements. This step too was not applicable.
10. Checked if any character or parsed entity references had to be replaced. No entity references seem to be present and this step was not applicable for File B
11. Checked to see if any special characters had to be replaced with character reference. Could not find any.
12. Checked if any CDATA had to be replaced with character content and this too was not applicable as there seem to be no CDATA content.
13. Added ending tags to some of the empty elements which were missing those – Ex: <event>,< submitted>
14. Checked to see if there is any white space within content that had to be retained, but this was not applicable for File B
15. Checked if any superfluous namespace declarations had to be removed , but this step too was not applicable for the XML as there are no namespaces URIs used.
16. Once the above steps are complete, calculated the **MD5 checksum -** 530466188e24a7869850d9324f573379



**Applying Additional Data Curation Steps in Creating Proposed Final Canonicalized XML**

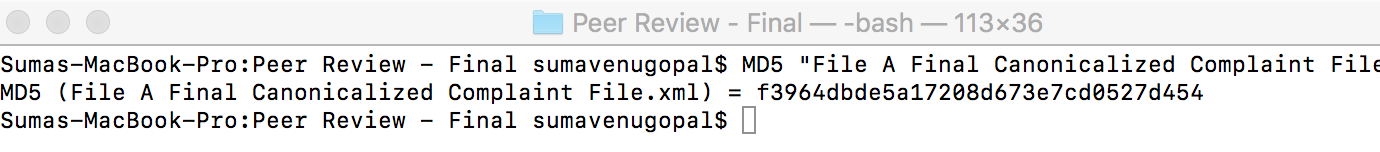
1. After creating canonicalized versions of File A and File B XMLs following C14N in the above steps, compared these two canonical files and saw that there were still slight differences. Also the MD5 checksums do not match. Hence decided to create final data curated canonicalized XMLs with matching checksums which will have all the details from File A and File B XMLs thus supporting the overarching data curation objectives of data integration, preservation, completeness ,modification and also to prevent any information loss.
2. Started the process with File\_B\_Canonical.xml, the C14N canonicalized version of File B.
3. Decided to retain the tag names as they are as they seemed to be almost the same as File A for most elements and attributes and have really good semantic value and can aid metadata creation and reproducibility.
4. In File B XML, submission mode seems to be not present for some complaints. For each <complaint> that did not have a ‘submissionType ‘attribute, fetched it from the <submitted> ‘via’ attribute of the respective complaint in File A XML and added it along with corresponding value to the final canonicalized XML ensuring data preservation and completeness.
5. <submitted> element was missing in File B XML for the most part, apart from an unpopulated instance , but decided to not explicitly address it as same information was added as submissionType on all records in File B XML as part of the above step ensuring data completeness. This ensures data is organized and avoids redundancy.
6. However <submitted> tag was not removed and was retained in cases where it was present, to avoid any possible system or data model compatibility issues and to ensure reproducibility. In real world scenario, availability of more details about new system would have helped take a decision about whether to retain this element or not.
7. Also noticed that the <response> element for a particular complaint (id = 837784 ) was missing a ‘timely’ attribute in File B XML. It was added based on corresponding value from File A for the same complaint ensuring data preservation and completeness.
8. Compared File A and File B XMLs from above step and figured that ‘timely’ Boolean attribute was referred to using ‘yes’ or ‘no’ in File B compared to ‘Y’ or ‘N’ in File A. Decided to stick to yes or no notation. This can be considered as reformatting from file A perspective.
9. Also decided to retain the available comment within the File B XML, as it documents an update that was done to the data set and thus supports provenance.
10. Decided to replace the ‘&’ in the final XML with ‘&amp’; to escape & and ensure successful access and reproducibility with respect to the DTD XML parsing of PCDATA element
11. Sorting of elements is NOT handled as part of C14N , although sorting of attributes is a step in C14N. Last complaint (id = 14038) has its elements ordered differently compared to other complaints. So decided to re-order these elements in the same way as other complaints to ensure more readability and ease in any kind of workflow processing.
12. Creation of final Canonical XML from File B is complete.
13. Now, took the File A Canonical XML created earlier following C14N and did a comparison with the final canonicalized XML from the above step
14. Incorporated all the differences from the final canonical XML from step 12 back to the final File A XML , so the files would look identical. Followed more or less similar process that were adopted in the steps detailed above.
15. Creation of final Canonical XML from File A is complete.
16. Generated MD5 checksums of the final files from Step 12 and Step 15 and calculated the MD5 checksum for each .Confirmed that both values match as shown below.
17. Once creation of the final canonical XML from File B is complete, calculated the MD5 checksum which is provided below:

**MD5 checksum -**  f3964dbde5a17208d673e7cd0527d454



1. Once creation of the final canonical XML from File A is complete, calculated the MD5 checksum which is provided below:

**MD5 checksum -**  f3964dbde5a17208d673e7cd0527d454

****

# **REPRODUCIBILITY**

Data provided in both File A XML and File B XML has been analyzed is detail and respective DTD defining the structure of data was created during the initial steps in this curation process / assignment. Post canonicalization , both the XML files were integrated in such a way that no information from both the XMLs is lost. A DTD was created to define this final canonicalized XML and made sure that this final XML validates against the DTD. Each and every step of analysis and processing methodologies adopted have been documented in detail in the previous steps to support reproducibility.

Data curation steps followed also ensure that we have a final canonicalized, well-formed and valid XML -DTD combination ensuring no information loss and with the capability to reproduce valid, reliable, structured, machine and human readable data records whenever needed.

# **CANONICALIZATION & DATA CURATION GOALS**

By following the below steps in the canonicalization process, it was ensured that all the important data curation activities were addressed :

* Conducted thorough analysis of the data files provided to make sure that a good understanding on the files A and B, their formats, data composition , completeness, purpose, metadata, content, short comings etc. were understood
* Documented each and every detail about the files as well as about the entity and attribute details from the analysis step above, so that those would be useful in the upcoming steps, in DTD creation, canonicalization as well as in the future.
* Designed the canonicalization process around the C14N XML canonicalization algorithm and implemented it step by step along with documentation for each step to ensure proper implementation and tracking.
* Ensured that a final canonicalized, well-formed and valid XML along with its DTD was created and that the XML validates against the DTD. This step ensured that the result to the data curation process adhered to the correct procedures, formats and policies and thus meets the organization’s data policy as interpreted from the XMLs
* Each and every step of canonicalization was followed as perfectly as it should be, especially during final integration, ensuring no information is lost thus making sure that the data curation process is followed as expected.

Listed below are the details of the data curation objectives that were in consideration during the C14N canonicalization process and final XML file creation.

* Appropriate DTD were defined for the initial XML files A and B, as well as for the final canonicalized file to make sure that the data accessibility and modification are supported at all levels.
* Contents and format of the XML also ensures data identification especially due to the structured content , meaningful tag names and data records that are complete end-to-end post canonicalization and integration.
* XML file in the final standard format along with a defined DTD will facilitate the sharing of data across varies parties and systems that might benefit from it.
* A well-formed, valid and machine readable final XML has been created post canonicalization that will also support systematic data workflows thus making automated processing much easier.
* During the entire process, utmost care has been taken in ensuring that data or the data tags in the source and destination systems are preserved for current and future use and are not tampered with in any manner.
* The above step also ensures discoverability, making sure that anyone looking for a certain piece of information can access it and process it correctly.
* Data curation steps followed also ensure that we have a final canonicalized, well-formed and valid XML -DTD combination ensuring no information loss and with the capability to reproduce valid, reliable, structured, machine and human readable data records whenever needed.
* Slight data model differences between files A and B were addressed effectively during integration in such a manner that doesn’t compromise the quality of content, its readability, accessibility or completeness
* As part of canonicalization and optimization reformatting was also done that involved space removal, adding XML tags if needed and sorting attributes if needed to make sure that the final file was more streamlined, structured and readable.

# **FUTURE CONSIDERATIONS**

Below are some of the additional curation activities / objectives that can be recommended to enhance the data set for future discovery and use:

* The analysis and canonicalization done here was based on certain assumptions about the data based on what we got see in the File A and B XMLs. It is important that adequate details are made available to the curation team in a real-word scenario , providing them more details about the processes, scenarios, calculations, real world input data etc. that led to the data set creation. Thus provenance is an important data curation objective that should be addressed.
* It is important that the data adheres to the compliance requirements. XMLs seem to have finance related data. There might be compliance requirements like PCI, HIPAA etc. depending on the industry the data set is related to and that needs to be met within the data set.
* It is also important that the dataset is secured physically with respect to where it resides and programmatically by using proper authentication mechanisms, thus avoiding unauthorized access.
* Continuing support is needed for the ongoing collection of data, as the scope and or format can change anytime to meet system or user level requirements
* It is also important that the data is organized and stored effectively safe and long-term, meeting the archival requirements of the data source and the data user.
* Storage requirements need to be checked and revised effectively to meet the needs as the data grows over a period of time.
* Organization that owns the data also is expected to periodically revise and audit its policies related to the data set and the related processing and curation activities, thus ensuring that all these aspects are up to date and outdated in any way.