

Serializing DOM Method

Position Paper for the W3C Workshop on Binary Interchange of XML Information Item Sets

24-26th September, 2003

Atsushi Sakakibara

Media Fusion Co., Ltd.

info@mediafusion.co.jp

<Abstract>

Cellular phone has spread throughout rapidly in these years. To say nothing of e-mail and internet, its application domain is expected to become widespread. However, CPU and memory which cellular phone can be equipped is limited because of its size of the body. This limits data quantity it can handle.

It is not comfortable to handle XML documents on cellular phone. This is because one needs a considerable amount of processing and memory usage for parsing and other subordinate operations to handle them. Embedded home electric appliances which can be equipped with limited CPU and memory are the same. This must make application domain of XML narrower and prevent its spread. We aimed to reduce CPU consumption, memory usage and bandwidth with our proposal technology to solve these problems. We have developed a space-efficient pre-parsed XML in binary format which does not require parsing or decompression on reference at all, and implemented in our products. This research and development was funded by Telecommunications Advancement Organization of Japan. We believe that this binary format transmission enable to load applications using XML onto cellular phone and embedded home electric appliances, and wider spread of XML.

<Essence of Our Proposing Idea>

We targeted a space-efficient pre-parsed XML in binary format which does not require parsing or decompression on reference at all in our proposing technology.

Our pre-parsed XML format consists of two separate parts, Structural part and Data part., and each part is compressed in different ways. In the structural part, the DOM which is serialized for byte stream with consideration for data transfer is used (Serialized DOM). Data part is a one string which has all the strings appearing in a document as partial string. These two serial data are merged into one. The structural part is compressed in a mechanism in which minimum bytes which will be needed to express the data will be sent and received (byte level compression) for the efficient usage of space resources in structural part. The data part is compressed in a way to minimize the occurrence of the same sequence of characters.
(Refer to "structure of our pre-parsed XML format" for more details.)

<What we have done in this area>

We have focused on SVG and SMIL, and succeeded to develop middleware using XML for cellular phone and PDA. Our middleware are equipped with browsers to display SVG data for cellular phone/PDA, and will be equipped with a tool to render SMIL data soon. Our proposing technology has been already implemented in the middleware. We could manage to display map data in SVG format on cellular phone, which had been impossible before, using this technology. High-speed display is realized because receiving side (cellular phone) does not need to parse data. Large-sized map data can also be displayed as the contents of the data can be accessed without decompression. Of course, bandwidth usage is reduced as data is transferred in compressed form.

<Our focus in this area>

We have researched efficient ways to transfer and render SVG/SMIL data for devices with limited CPU and memory resources, such as cellular phone and PDA, preparing for the development of our middleware mentioned above. We mainly researched methods for SVG/SMIL data, especially for map data, handled on cellular phone and PDA.

Our aim was to transfer and render information for devices with limited resource efficiently. Therefore, our most important goal in this area was to access to the contents of compressed data without parsing or decompression at receiving side and to reduce bandwidth usage; High-speed rendering can be

realized if receiving side does not need to parse data. Large-sized data can be rendered on devices with limited resource if the contents of the data can be accessed without decompression. Bandwidth usage is reduced if data is transferred in compressed form.

<Internationalization and Accessibility>

We developed the above mentioned middleware in the Japanese environment, which is two-byte language environment, so that the technology supports not only one-byte languages but also two-byte languages. This means we ensure that internationalization was not compromised.

Also one can access to contents of pre-parsed XML documents without parsing or decompression by utilizing DOM-like method (refer to 1.1. Structural Part in <Structure of our pre-parsed XML format> for the details) with this technology.

<Comparison with using gzip on raw XML>

When you compress raw XML with gzip, you can reduce the data size. However, you need to decompress the compressed data at receiving side. In other words, it requires processing time for decompression, and the process takes a considerable amount of time. The receiving side cannot estimate required memory consumption for decompression of the compressed data in gzip format without doing so. This means, even if you can receive compressed data, you may not be able to access to its contents when your device does not have enough memory for decompression. To sum up, data compression with gzip can reduce data transfer time, but there are problems when accessing to the data contents after receiving it.

Our proposing compression technology can solve these problems. This forms pre-parsed XML in space-efficient binary format, so that you can access to each element of a compressed XML document, and you are not required to parse or decompress the document when referencing its contents. This enables you to use limited memory resource at receiving side efficiently.

<Interoperability between Vocabularies>

This technology can be applied not only to XML but also to other of “structured documents”. It goes without saying that this can be applied to any kind of XML documents.

As we will mention below, Dynamic Update is not important to our technology, and we do not need to consider about schema. Therefore, this technology is independent from any schemas.

<Dynamic Update, Streaming, Random Access>

The target field of this technology is devices which can be equipped with limited CPU or memory, such as cellular phone and home electric appliances. Currently, we do not consider Dynamic Update is important for the field. Therefore, we have not taken Dynamic Update into our consideration in our technology.

We consider Streaming and Random Access are very important for the field to achieve our above mentioned goal, and have already implemented.

<Structure of our pre-parsed XML format>

1. Separation of Structures and Data.

The format of pre-parsed XML we propose in this paper is as follows.

Our pre-parsed XML format consists of two separate parts, Structural part and Data part., and each part is compressed in different ways. In the structural part, the DOM which is serialized for byte stream with consideration for data transfer is used (Serialized DOM). Data part is a one string which has all the strings appearing in a document as partial string. These two serial data are merged into one. The structural part is compressed in a mechanism in which minimum bytes which will be needed to express the data will be sent and received (byte level compression) for the efficient usage of space resources in structural part. The data part is compressed in a way to minimize the occurrence of the same sequence of characters.

1.1. Structural Part (The Serialized DOM).

DOM-like structure in which set of pointers point the positions of individual element on the stream will be constructed. (refer to Figure-2)

As every element holds the positions of related elements on stream, we will be able to access the element directly (traverse between nodes) without any reparsing and deserialization.

The element has no real value. The real value will have to be found from the position and the length on stream.

1.2. Data Part (compression of string dictionary)

The values of elements (Node Name and Node Value) will be stored as one sequence of strings.

From the structural part, this sequence will be referred as partial strings derived from its position on the stream, and then assigned as to be the value of the element.

In serialized-DOM, same sequence of characters will be referred from many different elements so whole element that has same element name will not be necessary to be sent.

This method is suitable for documents which contain a lot of same sequence of characters, such as SVG and SMIL documents.

1.3 The efficient usage of spaces of resources in Structural Part (byte-level compression).

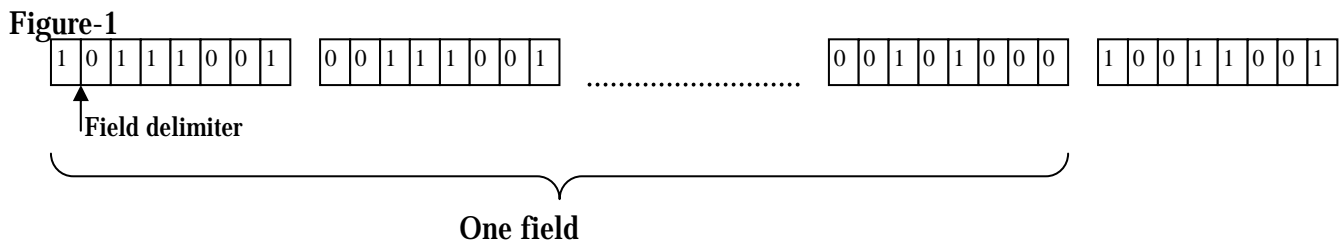
The structural part will be binaries because it will be consisted of mainly aggregations of DOM-like containers (explained in section 1.1.), which contain plural positions on the stream.

Now we consider the encoding method of these binaries which retain the capability of streaming and random-accessibility.

The encoding will be achieved as follows.

The most significant bit in the 1 byte represents "field delimiter" and the following 7 bits will represent data. This will describe variable-length fields in which each field is starting from the byte containing "field delimiter" ending at right before the byte containing "field delimiter".

Using the above representation, variable field will be defined as follows.



In this way, data which size is less than 7 bits will be stored in 1 byte and one which size is larger than 7 bits will be stored in two or more byte blocks.

Therefore, the size of field is growing gradually as it requires. Then the efficiency of the space of the resources will be achieved. (refer to Figure-3)

1.4 The improvement of the efficient usage of the space of resources in Data Part.

The Data Part will be described as one long sequence of strings. Naturally, the same sub-string will be occurred frequently. Such an occurrence will be suppressed as much as possible. This improves the efficient usage of the space of resources. Exactly same elements, elements containing partial strings, and then overlapping part at the beginning and the end of words will be put together step by step. (refer to Figure-4)

2. Access to compressed data

Each element of the compressed XML document can be accessed as it is without decompression. We suppose that subset of XML-DOM will be suitable for this purpose at accessing to each element of such a pre-parsed XML.

While original DOM retains reference to objects representing each element and traverse the data, in

pre-parsed XML, we translate the position on the stream as reference and seeking on the stream as traversing the data. In other words, reference to an object is replaced by a seek pointer and reference to a string by stream position.

The possible functions to implement are as follows:

- * Reconstruction of an XML document
- * Reconstruction of the element contained in XML documents
- * The traverse between nodes (parents, children and siblings)
- * Refer to element name.
- * Refer to element value.

<Conclusion>

Currently, handling XML documents, which has redundant tendency, on devices equipped with limited CPU and memory resources, including cellular phone embedded home electric appliances, is troublesome. A considerable amount of processing and memory usage for parsing and other subordinate operations is required to handle XML documents, and this is problematic to those devices. We have developed a space-efficient pre-parsed XML in binary format which does not require parsing or decompression on reference at all, and implemented in our products. This technology enables to reduce CPU consumption, memory usage and bandwidth, and can solve the problems mentioned above. We believe this technology is brilliant to transfer and render XML document to such devices.

Figure-2 (Structure part)

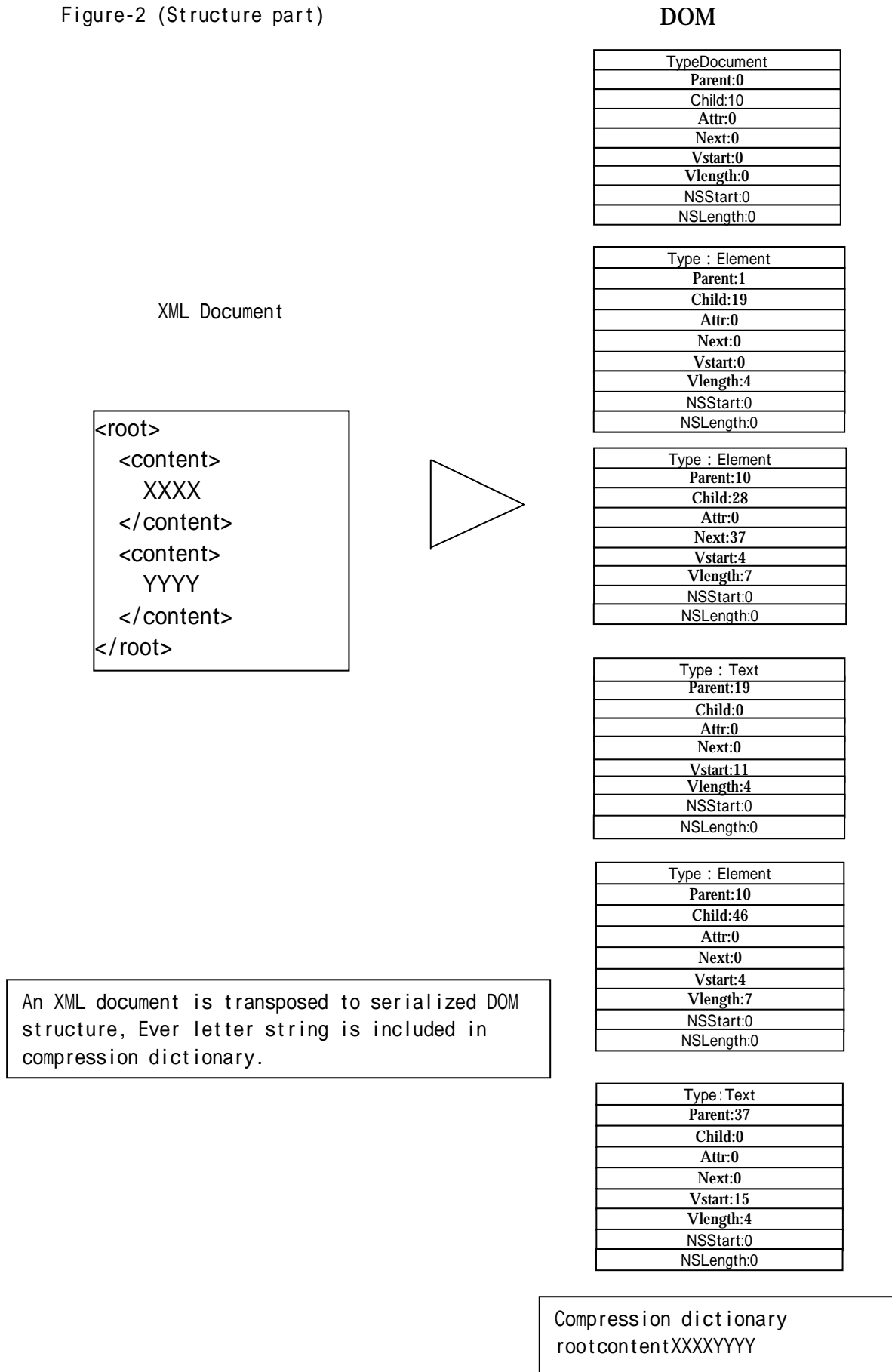
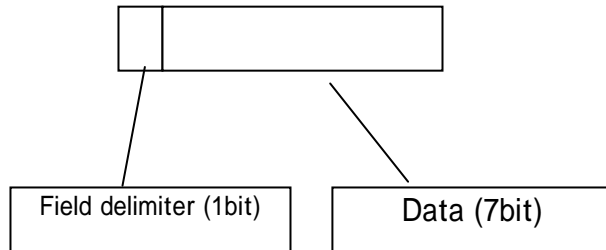


Figure-3
Structure of byte-data



Example:

When the value of a field is 1

1	0000001
---	---------

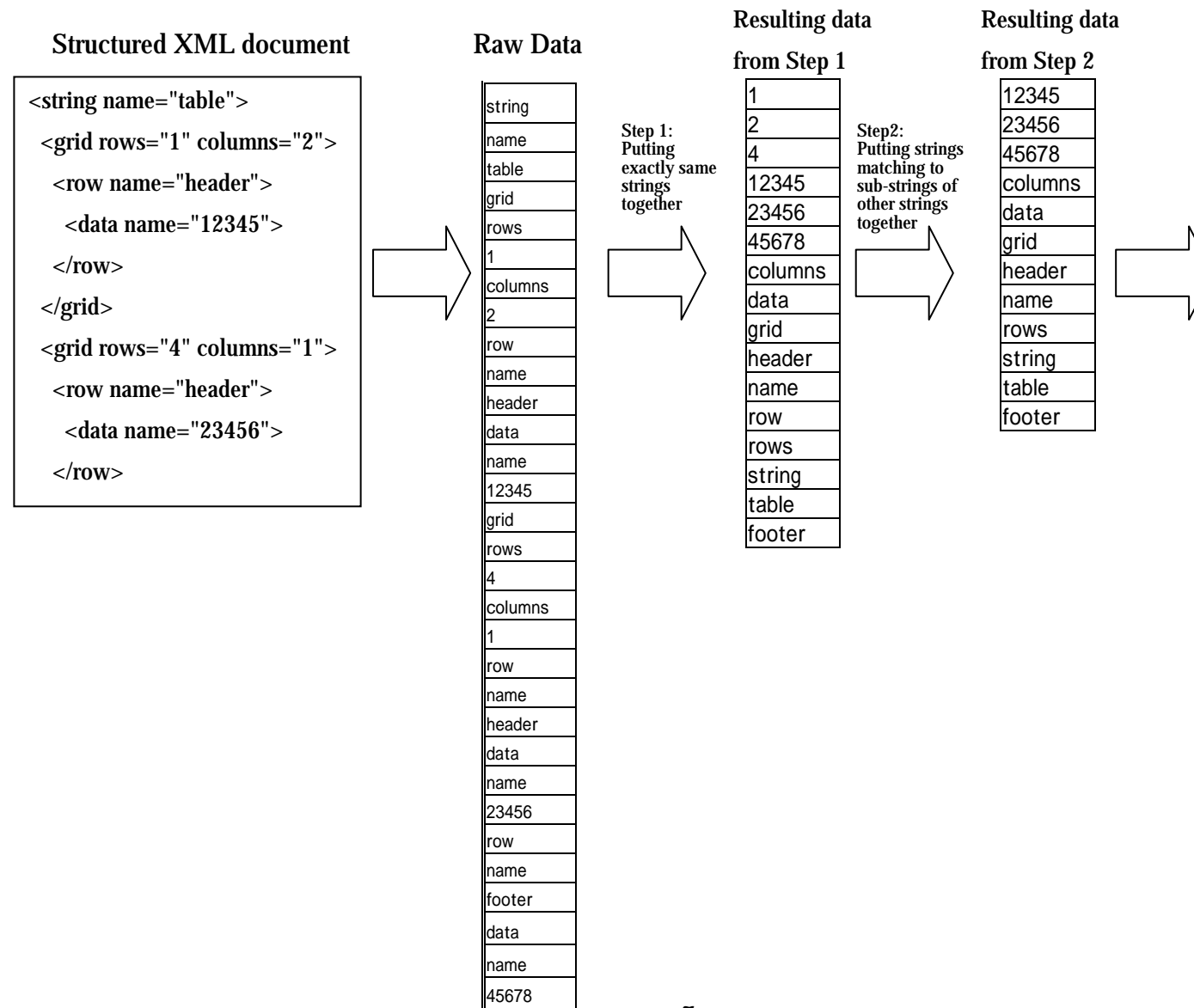
When the value of a field is 1200

1	1001000	0	0000001
---	---------	---	---------

When the value of a field is 1100000

1	0100000	0	00011001	0	0000110
---	---------	---	----------	---	---------

Figure 4 (The improvement of the efficient usage of the space of resources in Data Part)



Resulting data from Step 3

Step3:
Putting
matching prefix
and suffix
together

[illegible]

Final Result

[illegible]