

바이너리 XML 기술 동향

변일수¹ 김성운² 안창원³ 박종근⁴ 조희남⁵ Desmons Frederic⁶

현재 XML은 가장 널리 받아들여지고 있는 문서 형식중의 하나로 그 사용분야는 점차 확대되어 가고 있다. 그러나 XML의 성공은 XML이 의도되지 않은 곳에까지 사용되는 결과를 낳았고, 뜻하지 않던 XML의 문제점이 거론되기 시작하였다. 문제점 중 하나는 XML이 너무 장황(verbose)하기 때문에 과부하가 크다는 점이다. 이는 소형기거나 네트워크 환경에서 치명적인 약점이 될 수 있다. 이 문제를 해결하기 위해 XML 문서의 바이너리 형식에 대한 연구가 진행되기 시작하였다. 하지만 이것은 민감한 문제이다. XML 문서의 이진화는 XML의 본래 취지에 역행하는 것이라고 생각하는 사람도 적지 않다. 이들은 XML의 이진화가 과거와 같이 독자적인 데이터 형식을 정의하게 해 폐쇄적인 환경으로 돌아가게 되는 것을 두려워한다. XML의 개방성을 유지하면서 내재된 문제점을 해결하기 위해서는 바이너리 XML에 대한 표준이 필요하다. 본 고에서는 바이너리 XML과 관련된 연구 동향과 표준화 추진 현황을 소개하고자 한다.

목 차

- I. 서 론
- II. 바이너리 XML 연구 동향
- III. 표준화 추진 현황
- IV. 결 론

I. 서 론

XML[1]은 운영 환경과는 독립적인 문서 형식을 정의할 수 있다는 장점으로 인해 많은 사용자층을 확보할 수 있었고, 현재 가장 주목 받는 문서 형식 중의 하나가 되었다.

그러나 양날의 검처럼 XML의 장점은 특정 분야에서는 단점으로 작용하고 있다. XML은 상호운용성(Interoperability)을 확보하고 보다 쉬운 접근이 가능하게 하기 위해 인간이 읽을 수 있는 텍스트 형식을 취하고 있다. 이는 실제로 데이터 처리과정에는 필요 없는 여분의 데이터를 포함하고 있다는 뜻이 되며, 텍스트 형식을 취하게 됨으로 생기게 되는 저장 공간의 손실도 존재한다. 일반적인 컴퓨팅 환경에서 이 정도의 과부하는 문제가 되지 않는다. 하지만 네트워크 환경이

1 ETRI 서버플랫폼연구팀/연구원
 2 ETRI 서버플랫폼연구팀/책임연구원
 3 ETRI 서버플랫폼연구팀/선임연구원
 4 ETRI 서버플랫폼연구팀/연구원
 5 ETRI 서버플랫폼연구팀/연구원
 6 ETRI 서버플랫폼연구팀/연구원

나 임베디드 시스템 같이 상대적으로 느린 매체에 자주 접근하거나 자원이 제한된 소형 기기의 경우 XML 문서의 과부하는 전체 응용 프로그램의 성능을 결정짓는 중요한 요소가 될 수 있다. 이에 대한 예로 웹서비스를 생각해 볼 수 있다. 웹서비스는 이기종간의 상호 운영성을 보장하기 위해 메시지 형식으로 XML 형식을 취하고 있다. 그러나 상호 운영성에 대한 대가는 적지 않다. XML 형식의 메시지 교환은 XML 의 내재된 단점으로 인해 전체 응용 프로그램의 성능을 저하시킬 정도로 큰 과부하를 일으키고 있다[14].

이 때문에 업계에서는 XML 문서의 바이너리 형식에 대해 요구하기 시작했다. 이를 받아들여 W3C[6]에서는 바이너리 XML 에 대한 정당성을 확인하기 위해 바이너리 XML 에 대한 쓰임새 (Usecase)를 분석하고 취합하는 워킹 그룹(XML Binary Characterization Working Group)을 운영하고 있다[2]. 바이너리 XML 의 이점은 분명하다. 데이터를 네이티브(native) 형식으로 저장함으로써 문서의 크기를 줄이고 XML 문서의 파싱(parsing) 과정에 따른 과부하를 줄임으로 성능을 향상시킬 수 있다. 그러나 논란의 여지는 적지 않다. XML 의 최대 장점은 상호운용성(Interoperability)인데 이를 포기하고 바이너리 형식을 취한다는 것은 과거로의 회귀를 뜻하기 때문이다. 현재로서 이에 대한 가장 현실적인 접근 방법은 XML 문서의 바이너리 인코딩 표준을 마련하는 것이다.

본 논문에서는 현재 여러 분야에서 동시 다발적으로 일어나고 있는 바이너리 XML 연구 동향과 표준화 추진 현황을 다룬다.

II. 바이너리 XML 연구 동향

XML 을 이진화(binary)함으로 성능을 향상시키고자 하는 노력이 3 가지 측면에서 기울여지고 있다. 그것들은 압축, 문서 구조의 분리, 인코딩 등 총 3 가지로 분류될 수 있다.

1. 압축

XML 문서의 단점을 보완하기 위한 한가지 방법은 XML 문서를 압축하는 것이다. 서론에서 지적되었듯이 XML 문서는 사람이 읽을 수 있는 문서 형식을 지향하기 때문에 모든 데이터가 텍스트 형식으로 저장된다. 이것은 저장 공간을 낭비하는 원인이 된다. 예를 들어 숫자 100 을 저장한다고 가정해보자. 이것을 네이티브 형식으로 저장하고자 하면 단지 한 바이트(byte)면 된다. 하지만 텍스트 형식으로 저장하기 위해서는 문자 하나당 2 바이트씩 총 3 개의 문자로 6 바이트를 필요로 한다. 즉 네이티브 형식보다 무려 6 배의 저장 공간을 차지한다는 뜻이다. 게다가

XML 문서의 각 요소들은 한 쌍의 태그로 둘러싸인다. 결국 XML 문서의 크기는 불필요하게 커질 수 밖에 없다. 문서가 디스크나 네트워크와 같이 상대적으로 느린 매체를 통해 전송되어야 할 경우, 문서의 크기는 성능에 치명적인 요소일 수 밖에 없다. 따라서 문서의 크기를 줄이는 것은 상당히 도움이 된다. 과부하를 느린 매체에서 상대적으로 빠른 CPU 로 옮김으로 속도를 향상시키기 때문이다. 다시 말해서, 느린 매체에 다량의 정보를 넘겨주고 처리가 끝나기를 기다리는 대신, 상대적으로 처리 속도가 빠른 CPU 가 넘겨 주어야 할 정보의 양을 줄이도록 함으로 성능 상의 이득을 볼 수 있다는 뜻이다. 그래서 일부 웹서비스에서는 XML 문서를 송신하기 전에 압축 과정을 거친다. 단 수신측에서는 메시지를 읽기 위해 반드시 해제 과정을 거쳐야 한다.

가. Gzip

Gzip[3]은 GNU 프로젝트의 일환으로서 compress 를 대체하면서, 더 나은 성능을 보이며 동시에 특허에 의해 제한되지 않는 압축 알고리즘을 제공하고 있다. Gzip 알고리즘은 여러 언어로 이식되었으며 대부분의 라이브러리들이 오픈 소스로 제공되기 때문에 별다른 노력 없이 적용할 수 있다는 장점이 있다. 그러나 매 연산 시 압축과 해제 과정이 수반되어야 한다는 것이 단점으로 작용한다. 이것은 비단 Gzip 만의 문제라고는 할 수 없으며 압축에 의한 접근 방식이 어쩔 수 없이 가지게 되는 단점이다.

나. XMill

XMill[4]은 XML 문서에 특화된 압축 알고리즘이다. XMill 은 엔트로피(Entropy)기반의 압축 방식을 취하고 있다. 엔트로피 부호화의 기본 원리는 데이터 심볼들의 통계적 발생 빈도에 따라 각각의 심볼들을 적절한 길이의 부호로 표현하는 것이다. 데이터 심볼들의 발생 확률에 따라 엔트로피라고 불리는 심볼당 평균 정보량이 결정되는데, 엔트로피 부호화의 목표는 심볼당 평균 부호 길이가 엔트로피에 가까이 가도록 한다.

엔트로피 부호화에는 허프만(Huffman) 방식, 산술 방식, LZW 방식이 있다. 이 중에서 XMill 에 사용된 방식은 허프만 방식이다. 이것은 고정길이 부호를 가변길이 부호로 바꾸는 것으로서 빈도수가 높은 심볼에는 짧은 부호를, 빈도수가 낮은 심볼에는 긴 부호를 할당하여 평균 부호 길이를 원래 심볼의 고정 길이보다 짧게 하는 것이다.

영문 텍스트의 경우, 압축하지 않은 원문은 글자당 7 비트를 차지하는데 이를 허프만 방식을 이용해 부호화하면 자주 발생하는 글자들은 대략 3 비트 정도의 낮은 비트로, 빈도수가 낮은 기

호들은 10 비트 이상의 높은 비트로 부호화시켜 심볼당 평균 비트수를 원래의 그것보다 낮추는 것이다. XML 문서는 구조상 반복되는 구문이 많기 때문에 허프만 방식은 큰 효과를 나타낼 수 있다. 실제로 XML은 Gzip 보다 두 배 이상의 나은 성능을 보이고 있다.

2. 문서 구조의 분리

앞에서 우리는 압축 방식의 가장 큰 문제점으로 압축과 해제 과정에 따른 과부하를 지적했다. 만약 해당 문서의 모든 요소가 사용된다면 효과를 기대할 수 있지만 문서의 일부 요소만을 필요로 한다면 과부하는 더 커진다고 볼 수 있다. 실제 필요로 하는 정보 대비 요구되는 처리 능력을 고려해 볼 때 확실히 압축에 의한 접근 방식은 비효율적이다. 이에 대한 해결책으로 대두된 것이 ‘문서 구조의 분리’(Separation of Structure and Content)다. 이 방식에서는 데이터를 압축하기 전에 문서 구조(Structure)를 추출하고 콘텐츠(Content)에 대한 인덱스를 유지한다. 그리고 콘텐츠는 용량을 줄이기 위해 압축된다. 압축에는 gzip 이나 deflate[19]같은 일반적인 압축 기법들이 사용된다.

문서 구조를 추출함으로써 갖게 되는 장점은 압축 해제 과정을 선택적으로 할 수 있다는 데에 있다. 먼저 원하는 콘텐츠의 위치를 문서 구조를 통해 파악하고, 그 다음에 해당 위치의 콘텐츠만을 선택적으로 압축 해제함으로써 압축 해제에 따른 과부하를 최소한으로 유지할 수 있다. 그러나 XML 문서 전체의 요소(Element)가 전부 사용되어야만 하는 상황이라면 문서 구조 추출에 의한 이점은 대부분 상쇄되어 압축 기법과 별로 차이가 없어진다는 것이 단점이다.

다음은 ‘문서 구조의 분리’ 접근 방식을 취한 몇 가지 예들에 대한 설명이다.

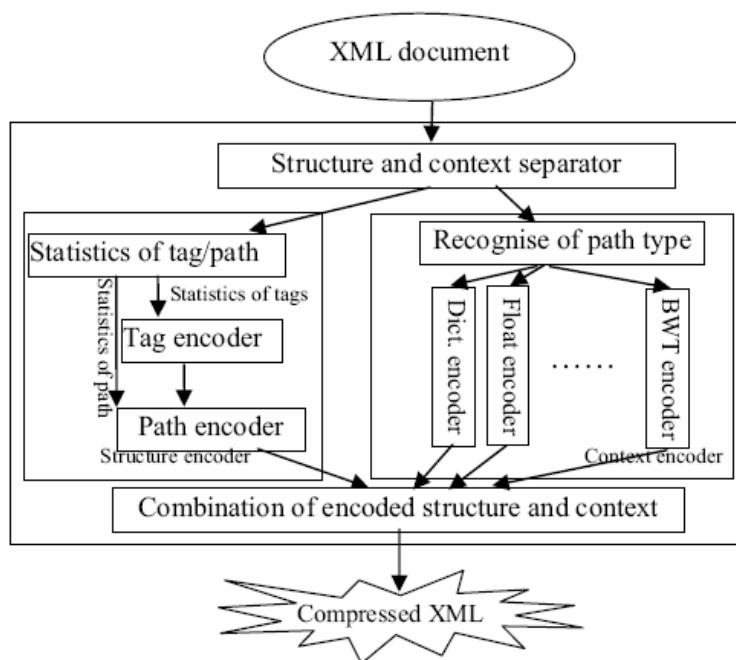
가. Millau

Millau[5]인코딩 형식은 WAP(Wireless Application Protocol)[11] 바이너리 XML 형식에 대한 확장이다. 이 형식은 XML 문서의 크기를 줄이기 위해 고안되었다. 태그와 속성은 토큰(Token)으로 변환된다. 그리고 문서 구조와 콘텐츠를 분리해 각각의 데이터를 서로 다른 스트림으로 전송한다. 문서 구조는 WBXML 인코딩 형식에 의해 변환된다. 반면 콘텐츠는 일반적인 압축 알고리즘에 의해 압축된다. 사용자는 모든 문서를 전송받을 필요가 없다. 문서 구조와 콘텐츠가 서로 다른 스트림으로 전송되기 때문에, 먼저 전송 받은 문서 구조를 조사한 후, 원하는 부분만을 전송받으면 된다. 이것은 데이터 전환에 의한 과부하를 없애는 역할을 한다. 이 형식은 네트워크 상에서 XML 문서를 주고 받을 때, 특히 문서의 일부분만을 참조하고자 할 때 유용하

다.

나. XCpaqs

XCpaqs[15]는 XML 문서의 압축 방식이면서 XPath 를 지원한다는 점이 특징이다. XCpaqs 의 전체적인 구조는 다음과 같다.



[그림 1] XCpaqs 의 아키텍처

그림 1 에서 볼 수 있듯이 ‘Structure and context separator’는 문서 구조와 콘텐츠를 분리한다. 문서 구조를 통해서 Xpath 경로와 해당 경로의 데이터 타입이 결정된다(좌측). 콘텐츠는 문서 구조 분석기에 의해 결정된 데이터 타입에 따라 적절한 압축기로 압축된다(우측). 그리고 두 결과를 조합하는 것으로 모든 과정을 마무리한다. XCpaqs 는 데이터 타입에 따라 압축 방식을 바꿀 수 있다는 특징을 가지고 있다.

3. 인코딩(Encoding)

본래의 XML 문서의 의미와 구조가 보존될 수 있도록 고유의 인코딩 포맷을 정의하는 방법도

있다. XML 문서의 데이터들은 각각의 데이터 타입에 맞는 네이티브 형식으로 인코딩되기 때문에 공간이 절약된다. 하지만 일반 압축 기법의 압축률과 비교할 때 인코딩으로 인해 절약되는 공간의 크기는 미약하다. 그러나 이것이 곧 인코딩 기법의 성능이 압축 기법보다 낫다는 뜻은 결코 아니다. 인코딩 기법의 장점은 단순히 공간을 절약하는 데에 있지 않다. 인코딩된 바이너리 XML 문서를 Pre-parsed XML 이라고도 부르는데, 여기에 함축된 의미는 XML 문서의 파싱 과정이 더는 필요하지 않다는 것이다. 미리 파싱된 형태로 데이터를 저장함으로써 매 접근 시마다 텍스트 문자열들을 파싱하고 데이터를 추출하는 과정을 거칠 필요가 없다는 말이다. XML 문서가 가지는 단점 중의 하나는 데이터가 인간 친화적인(human-friendly) 형태로 기록되었기 때문에 컴퓨터가 이해하기 위해서는 파싱과정을 통해 데이터를 추출하고 변환해야만 한다는 데에 있다. 이것은 가볍지 않은 처리 과정이며, 임베디드 기기처럼 자원이 제한된 기기의 경우에는 특히나 더 그렇다. Pre-parsed XML 에서는 파싱과정이 제거되었기 때문에 CPU Usage 나 Memory footprint 측면에서 상당한 이득이 있으며, 문서에 대한 접근 속도는 비약적으로 향상된다. 이러한 장점들은 단순히 크기를 줄임으로 얻게 되는 성능 상의 이득을 충분히 능가할 수 있으리라 여겨진다. 특히 한 XML 문서에 대해 반복적인 접근이 필요한 경우 파싱 과정을 제거함으로써 인해 생기는 이득은 훨씬 커진다.

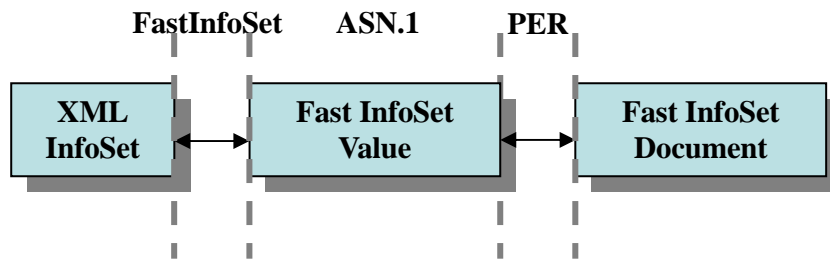
다음은 인코딩 방법을 적용한 예들이다.

가. Fast Infoset

Fast Infoset[8]을 이해하기 위해서는 XML 스펙에 대해 먼저 살펴보는 것이 도움이 된다. XML 스펙에는 XML Information Set[17] 스펙이 존재한다. 이는 XML 이 표현할 수 있는 정보를 추상적으로 정의한 것으로서 XML 이 어떤 정보를 가질 수 있는 지에 대한 내용만 있을 뿐 어떻게 표현할지에 대해서는 전혀 언급이 없다. 일종의 관념적인 모델이다. 이에 대한 실제화를 XML[16]스펙에서 담당하고 있다. 달리 생각하자면 우리가 일상적으로 보게 되는 각괄호(<,>)로 이루어진 XML 문서는 XML Information Set 의 구현물이라고 생각할 수 있다. Fast Infoset 은 XML Information Set 의 또 다른 구현물이며 이진 표현이다.

그렇다면 이진 코드는 어떻게 생성되는가? Fast Infoset 은 거의 20년동안 통신업계에서 사용되어 안정성과 실효성이 증명된 ASN.1(Abstract Syntax Notation One)[7]이라는 추상화 구문을 이용하고 있다. ASN.1 은 프로토콜의 설계를 위해 고안된 언어로서 프로토콜을 정의하는 추상화 구문과 실제 전송 라인에서 어떤 비트 패턴으로 표현될 지에 관한 전송 구문을 완전히 분

리시켜 놓았다. ASN.1 에 의해 표현된 추상적인 프로토콜은 인코딩룰(Encoding rule)에 의해 실질적인 비트 패턴으로 변환된다. BER(Binary Encoding Rule), PER(Packed Encoding Rule), XER(XML Encoding Rule)등 다양한 인코딩 룰이 정의되어 있어 적합한 룰을 가져다 쓰기만 하면 된다. 고유의 비트 패턴을 정의하고자 한다면 ECN(Encoding Control Notation)을 통해 새로운 룰을 정의할 수도 있다. 다음은 Fast Infoset 을 통한 XML Information Set 의 인코딩 과정을 관념적으로 표현하고 있다[9].



[그림 2] Fast Infoset

Fast Infoset 은 문서의 크기를 줄이고자 테이블과 인덱싱 기법을 사용하고 있다. 처음 나타나는 문자열을 테이블에 저장해 놓고 동일한 문자열이 나타날 경우 인덱스로 대체함으로써 XML 문서의 크기를 줄인다.

Fast Infoset 을 적용했을 경우 문서의 압축률은 압축 기법에 비해 별로 좋지는 않다. 하지만 성능상의 향상을 체감할 수 있을 정도는 되며, 부가적으로 일반 XML 문서나 압축된 XML 문서에 비해 접근 성능이 비약적으로 향상되는 것을 확인할 수 있다[12].

나. XBIS(XML Binary Information Set)

XBIS[10]는 본래 텍스트로 이루어진 XML 문서로의 완전한 전환을 지원하는 고유의 인코딩 포맷을 정의하고 있다. XBIS 의 장점은 간결함(Compactness)에 있는데, 그것은 일반 XML 문서의 반복적인 구문을 활용하는 데에 기인한다. XBIS 는 반복되는 구문을 단 한번만 정의하고 반복될 때마다 원래의 구문을 참조하는 핸들을 사용함으로써 문서의 크기를 줄인다. 게다가 XBIS 는 데이터를 사용하기 쉬운 형식(Predigested form)으로 변환시켜 놓음으로 문서에 대한 접근 속도를 향상시켰다. XBIS 의 접근 방식은 Fast Infoset 과 상당히 유사하다. 하지만 XBIS 는 표준

에 근거하지 않은 독자적인 형식이라는 데에 그 한계점이 있다. 참조 구현은 자바로 작성되었으며 SAX2(Simple API for XML2)[18]인터페이스를 구현하고 있다.

III. 표준화 추진 현황

1. W3C

W3C에서는 업계의 요구를 받아들여 XBC-WG(XML Binary Characterization Working Group)를 구성하였다. 현재 이 워킹 그룹의 역할은 바이너리 XML의 쓰임새를 파악하여 바이너리 XML 관련 권고안(Recommendation)의 기초를 세우는 작업을 담당하고 있다. 이 말을 달리 표현하자면, 바이너리 XML과 관련된 스펙은 W3C 내에 아직 존재하지 않는다는 뜻이다. 다만 바이너리 XML의 필요성에 주의를 기울이기 시작했다는 점에 유의할 수 있다.

XBC-WG에서는 Binary XML이 반드시 만족해야 하는 최소한의 요구사항을 결정하기 위해 현재 XML이 잘 사용되지 않는 영역의 쓰임새를 분석하였다. 이를 위해 각종 속성을 정의하고, 각 쓰임새별 요구사항을 이들 속성으로 표현하였으며, 이를 XML에서 제공하는 속성과 비교함으로써 바이너리 XML에서 제공해야 할 최소한의 요구사항으로 도출하였다. 앞으로 규정될 바이너리 XML에 대한 어렵פות한 윤곽을 잡을 수 있도록 XBC-WG가 정의한 최소한 요구 사항들을 다음에 열거하였다[13].

- 전송 독립성 (Transport Independence)
- 인간 언어와의 중립성 (Human Language Neutral)
- 기술료 무료 (Royalty Free)
- 플랫폼 중립성 (Platform Neutrality)
- 콘텐츠 형식 관리 (Content Type Management)
- XML 스택으로의 통합 가능성 (Integratable into XML Stack)

결론적으로 말해서 W3C에서도 바이너리 XML에 대한 필요성을 인지하고 있으며 그 기술적 가능성과 요구사항에 대한 조사를 수행했고, 바이너리 XML에 대한 표준에 W3C가 적극적으로 참여하여 상호 호환성을 확보해야 함에 동의하고 있다.

2. Fast Infoset

현재 Fast Infoset 에 대한 스펙(ISO/IEC24824-1)은 초안 단계에 있다. 안타깝게도 아직 공개되지 않았으며 TIES(Telecom Information Exchange Services)사용자만이 스펙을 볼 수 있도록 제한되어 있다. 로드맵에 의하면 Fast Infoset 은 FDIS(Final Draft International Standard)를 위한 무기명 투표 중에 있으며, 2005년 6월 중순경에 권고안이 발표될 것으로 예정되어 있다.

IV. 결론

XML은 지금까지 대단히 성공적이었다고 할 수 있다. 어떤 플랫폼에도 종속되지 않은 독립적인 데이터 표현 방식은 기기종 간의 데이터 전송 수단으로 매우 유용함이 증명되어 왔다. 하지만 성공의 이면에는 XML 문서에 내재된 단점의 그늘이 드리워지고 있었다. XML의 성공은 XML의 사용이 의도되지 않은 분야로까지 확대되게 하였고 결국 성능과 관련된 이슈를 문제로 부각시켰다. 이를 해결하기 위해서는 XML 문서를 이진화해야 한다. 그리고 XML의 장점인 상호운용성(Interoperability)을 유지하기 위해 인코딩 형식에 대해 확립된 표준이 필요하다.

W3C에서는 XML의 이진화에 대한 요구사항을 파악하고 정당성을 파악하기 위한 절차에 들어갔다. 하지만 결과가 언제 나올지는 미지수다. 하지만 여기에서 우리는 하나의 중요한 움직임에 주목할 필요가 있다. ITU-T에서 ISO/IEC와 함께 XML의 이진화에 대한 스펙을 정의하고 있다는 점이다. Fast Infoset이 바로 그것이다. 아직은 공개되지 않은 초안의 형태이지만, 올해 중반기 쯤에는 그 모습을 볼 수 있을 것으로 예상된다. 게다가 관련 구현물도 이미 존재하며 성능상의 이점은 확인되었다. ASN.1이라는 이미 확립된 표준을 기반으로 한 응용 표준이기 때문에 적용 또한 어렵지 않을 것이다. 아직 확정된 것이 없어 설불리 판단을 내리긴 힘들겠지만, Fast Infoset이 바이너리 XML 표준의 강력한 후보 중에 하나가 되리라는 것만은 틀림 없다.

<참 고 문 헌>

- [1] Extensible Markup Language, <http://www.w3.org/XML/>
- [2] XML Binary Characterization Working Group, <http://www.w3.org/XML/Binary/>
- [3] Gnu ZIP, <http://www.gzip.org/>
- [4] XMill, <http://sourceforge.net/projects/xmill/>

- [5] Girardot, M., Sundaresan, N., “ Efficient representation and streaming of XML content over the Internet medium” , Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on Volume 1, 30 July-2 Aug. 2000 Page(s):67 - 70 vol.1
- [6] World Wide Web Consortium, <http://www.w3c.org/>
- [7] Abstrax Syntax Notation One, <http://asn1.elibel.tm.fr/en/index.htm>
- [8] Fast Infoset, <http://asn1.elibel.tm.fr/en/xml/#fast-infoset>
- [9] John, Larmouth., “ASN.1 Complete” , <http://www.oss.com/asn1/larmouth.html>
- [10] XML Binary Information Set, <http://sourceforge.net/projects/xmill/>
- [11] WAP, <http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html>
- [12] Paul, Sasndoz., Alessando, Triglia., Santiago, Pericas-Geertsen., “Fast Infoset”, <http://java.sun.com/developer/technicalArticles/xml/fastinfoset/>
- [13] 박종근, “XML Binary 특성화 분석,” 한국전자통신연구원, 관리번호 TM031020050734, 2005, 5. 30
- [14] Fast Web Service, <http://java.sun.com/developer/technicalArticles/WebServices/fastWS/>
- [15] Hongzhi Wang., Jianzhong Li., Jizhou Luo; Zhenying He.,” XCpaqs: compression of XML document with XPath query support” , Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on Volume 1, 5-7 April 2004 Page(s):354 - 358 Vol.1
- [16] Extensible Markup Language(XML) 1.0, <http://www.w3.org/TR/1998/REC-xml-19980210.html>
- [17] XML Information Set(Second Edition), <http://www.w3.org/TR/xml-infoset/>
- [18] Simple API for XML, <http://www.saxproject.org/>
- [19] P. Deutsch, "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, Aladdin Enterprises, May 1996, <http://www.ietf.org/rfc/rfc1951.txt>