BEX – Binary Encoded XML (Document)

Binary Encoded XML (Document) beschreibt das binäre Datenformat zur redundanzarmen Abbildung eines XML Dokuments. Ziel dieses Formats ist es, eine leichtgewichtige, nur lesende DOM-Implementation darauf aufsetzen zu können, welche signifikant weniger Arbeitsspeicher verbraucht, als eine zumeist auch schreiben könnende Implementation einer Standard XML Softwarebibliothek. Diese leichtgewichtige DOM-Implementation kann die Binärdatei Blockweise in den Arbeitsspeicher laden und dort zur Wiederverwendung gemäß einer most-recently-used Strategie vorhalten. Die offset-Listen sollten dazu vollständig im Arbeitsspeicher vorgehalten werden.

# Notation

Die folgenden, aufeinander aufbauenden Datenstrukturen werden jeweils als Auflistung ihrer Datenfeldern angegeben, wobei zu jedem der Felder dessen Datenformat, Anzahl und Beschreibung angegeben sind. Virtuelle Datenfelder, die nicht explizit in der Datei gespeichert werden, werden ohne Format und Anzahl angegeben. Das Datenformat ***INT(****X****)*** steht für eine positive, ganze Zahl mit *X* Byte Länge in Big-Endian. Die in den Beschreibungen verwendete Funktion ***lengthOf(****V****)*** gibt die kleinste Anzahl an Byte an, die für die Abbildung des Zahlenwerts *V* notwendig sind (für gilt und für gilt ).

# Datenstruktur: BEX

***BEX*** Kodiert einen DOM Dokumentknoten mit seinen enthaltenen Text-, Element- und Attributknoten als binäre Datenstruktur zum wahlfreien Zugriff.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| attrUriPool | *TextValuePool* | 1 | Auflistung der Zeichenketten für die URIs der Attributknoten. |
| attrNamePool | *TextValuePool* | 1 | Auflistung der Zeichenketten für die Namen der Attributknoten. |
| attrValuePool | *TextValuePool* | 1 | Auflistung der Zeichenketten für die Werte der Attributknoten. |
| attrGroupPool | *AttrGroupPool* | 1 | Auflistung der Attributknotenlisten. |
| elemUriPool | *TextValuePool* | 1 | Auflistung der Zeichenketten für die URIs der Elementknoten. |
| elemNamePool | *TextValuePool* | 1 | Auflistung der Zeichenketten für die Namen der Elementknoten. |
| elemValuePool | *TextValuePool* | 1 | Auflistung der Zeichenketten für die Werte in Elementknoten. |
| elemGroupPool | *ElemGroupPool* | 1 | Auflistung der Kindknotenlisten. |
| documentGroupRef | *INT(contentLength)* | 1 | Kindknotenliste des Dokumentknoten. Ist (i+elemValuePool.size+1) als Referenz auf die i-te Kindknotenliste in elemGroupPool. Die *contentLength* ist ***lengthOf(***elemValuePool.size+ elemGroupPool.size***)***. |

# Datenstruktur: TextValuePool

Kodiert eine Auflistung von Zeichenketten.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| size | *INT(4)* | 1 | Anzahl der Zeichenketten. |
| length | *INT(1)* | 1 | Anzahl der Byte je Startposition. |
| offset | *INT(length)* | size | Startpositionen der Zeichenketten im Speicherbereich item. Das erste Byte der i-ten Zeichenkette liegt bei Position offset[i] und das nach dem letzten bei offset[i+1]. Die Startposition offset[0] ist implizit 0. |
| item | *TextValueItem* | size | Speicherbereich mit den Bytes der Zeichenketten. |

# Datenstruktur: TextValueItem

Kodiert eine Zeichenkette.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| size | *-* | - | Anzahl der Bytes in data. |
| data | *INT(1)* | size | Bytes der UTF-8-kodierten Zeichenkette. |

# Datenstruktur: AttrGroupPool

Kodiert eine Auflistung von Attributknotenlisten.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| size | *INT(4)* | 1 | Anzahl der Attributknotenlisten. |
| length | *INT(1)* | 1 | Anzahl der Byte je Startposition. |
| offset | *INT(length)* | size | Startpositionen der Attributknotenlisten im Speicherbereich item. Der erste Attributknoten der i-ten Attributknotenliste liegt bei Position offset[i] und der nach dem letzten bei offset[i+1]. Die Startposition offset[0] ist implizit 0. |
| item | *AttrGroupItem* | size | Speicherbereich mit den Attributknoten aller Attributknotenlisten. |

# Datenstruktur: AttrGroupItem

Kodiert eine Attributknotenliste.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| size | *-* | - | Länge der Liste. |
| node | *AttrGrpupNode* | size | Attributknoten der Liste. |

# Datenstruktur: AttrGroupNode

Kodiert einen Attributknoten.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| uriRef | *INT(uriLength)* | 1 | Ist abwesend, wenn der attrUriPool leer ist. Ist (0), wenn der Attributknoten keinen URI besitzt. Ist (i+1) als Referenz auf den i-ten URI im attrUriPool. Die *uriLength* ist ***lengthOf(***attrUriPool.size***)***. |
| nameRef | *INT(nameLength)* | 1 | Ist (i) als Referenz auf den i-ten Namen im attrNamePool. Die *nameLength* ist ***lengthOf(***attrNamePool.size-1***)***. |
| valueRef | *INT(valueLength)* | 1 | Ist (i) als Referenz auf den i-ten Wert im attrValuePool. Die *valueLength* ist ***lengthOf(***attrValuePool.size-1***)***. |

# Datenstruktur: ElemGroupPool

Kodiert eine Auflistung von Kindknotenlisten.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| size | *INT(4)* | 1 | Anzahl der Kindknotenlisten. |
| length | *INT(1)* | 1 | Anzahl der Byte je Startposition. |
| offset | *INT(length)* | size | Startpositionen der Kindknotenlisten im Speicherbereich item. Der erste Kindknoten der i-ten Kindknotenliste liegt bei Position offset[i] und der nach dem letzten bei offset[i+1]. Die Startposition offset[0] ist implizit 0. |
| item | *ElemGroupItem* | size | Speicherbereich mit den Kindknoten der Kindknotenlisten. |

# Datenstruktur: ElemGroupItem

Kodiert eine Auflistung von Kindknotenlisten.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| size | *-* | - | Länge der Liste. |
| node | *ElemGrpupNode* | size | Kindknoten der Liste. |

# Datenstruktur: ElemGroupNode

Kodiert einen Kindknoten meist als Element- und sehr selten als Textknoten.

| Feld | Format | Anzahl | Beschreibung |
| --- | --- | --- | --- |
| uriRef | *INT(uriLength)* | 1 | Ist abwesend, wenn der elemUriPool leer ist. Ist (0), wenn der Kindknoten ein Textknoten oder ein Elementknoten ohne URI ist. Ist (i+1) als Referenz auf den i-ten URI im elemUriPool, wenn der Kindknoten ein Elementknoten ist. Die *uriLength* ist ***lengthOf(***elemUriPool.size***)***. |
| nameRef | *INT(nameLength)* | 1 | Ist (0), wenn der Kindknoten ein Textknoten ist. Ist (i+1) als Referenz auf den i-ten Namen im elemNamePool, wenn der Kindknoten ein Elementknoten ist. Die *nameLength* ist ***lengthOf(***elemNamePool.size***)***. |
| contentRef | *INT(contentLength)* | 1 | Ist (0), wenn der Kindknoten ein Elementknoten ohne Kindknoten ist. Ist (i+1) als Referenz auf den i-ten Wert im elemValuePool, wenn der Kindknoten ein Textknoten oder ein kindelementloser Elementknoten ist. Ist (i+1+elemValuePool.size) als Referenz auf die i-te Kindknotenliste im elemGroupPool, wenn der Kindknoten ein Elementknoten mit Kindelementen ist. Die *contentLength* ist ***lengthOf(***elemValuePool.size+ elemGroupPool.size***)***. |
| attributesRef | *INT(attributesLength)* | 1 | Ist abwesend, wenn der attrGroupPool leer ist. Ist (0), wenn der Kindknoten ein Textknoten oder ein Elementknoten ohne Attributknoten ist. Ist (i+1) als Referenz auf die i-te Attributknotenliste im attrGroupPool, wenn der Kindknoten ein Elementknoten mit Attributknoten ist. Die *attributesLength* ist ***lengthOf(***attrGroupPool.size***)***. |