INTERNATIONAL STANDARD

# ISO/IEC 21000-18

First edition
2007-06-15

# Information technology — Multimedia framework (MPEG-21) —

## Part 18:
## Digital Item Streaming

*Technologies de l'information — Cadre multimédia (MPEG-21) —*

*Partie 18: Transmission d'élément numérique*

Reference number
ISO/IEC 21000-18:2007(E)

**PDF disclaimer**

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

## Contents

Page

# Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 21000-18 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 21000 consists of the following parts, under the general title *Information technology — Multimedia framework (MPEG-21)*:

- *Part 1: Vision, Technologies and Strategy* [Technical Report]
- *Part 2: Digital Item Declaration*
- *Part 3: Digital Item Identification*
- *Part 4: Intellectual Property Management and Protection Components*
- *Part 5: Rights Expression Language*
- *Part 6: Rights Data Dictionary*
- *Part 7: Digital Item Adaptation*
- *Part 8: Reference Software*
- *Part 9: File Format*
- *Part 10: Digital Item Processing*
- *Part 11: Evaluation Tools for Persistent Association Technologies* [Technical Report]
- *Part 12: Test Bed for MPEG-21 Resource Delivery* [Technical Report]
- *Part 14: Conformance Testing*
- *Part 15: Event Reporting*
- *Part 16: Binary Format*
- *Part 17: Fragment Identification of MPEG Resources*
- *Part 18: Digital Item Streaming*

# Introduction

Today, many elements exist to build an infrastructure for the delivery and consumption of multimedia content. There is, however, no “big picture” to describe how these elements, either in existence or under development, relate to each other. The aim for MPEG-21 is to describe how these various elements fit together. Where gaps exist, MPEG-21 will recommend which new standards are required. ISO/IEC JTC 1/SC 29/WG 11 (MPEG) will then develop new standards as appropriate while other relevant standards may be developed by other bodies. These specifications will be integrated into the multimedia framework through collaboration between MPEG and these bodies.

The result is an open framework for multimedia delivery and consumption, with both the content creator and content consumer as focal points. This open framework provides content creators and service providers with equal opportunities in the MPEG-21 enabled open market. This will also be to the benefit of the content consumer providing them access to a large variety of content in an interoperable manner.

The vision for MPEG-21 is to define a multimedia framework *to enable transparent and augmented use of multimedia resources across a wide range of networks and devices* used by different communities.

Within MPEG-21, a Digital Item is defined as a structured digital object with a standard representation, identification and description. This entity is also the fundamental unit of distribution and transaction within this framework.

Digital Item Streaming enables the incremental delivery of a DI (DID, metadata, resources) in a piece-wise fashion and with temporal constraints in such a way that a receiving User may incrementally consume the DI. What is delivered is potentially a derived DI, i.e. one in which some parts may have been removed or transformed.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of a patent.

ISO and IEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO and IEC that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO and IEC. Information may be obtained from:

enikos pty ltd
c/o 18 Bulletin Place
Sydney 2001
Australia

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

# Information technology — Multimedia framework (MPEG-21) —

# Part 18: **Digital Item Streaming**

## 1 Scope

This part of ISO/IEC 21000 specifies tools for Digital Item Streaming. The first tool is the Bitstream Binding Language, which describes how Digital Items (comprising the Digital Item Declaration, metadata and resources) can be mapped to delivery channels such as MPEG-2 Transport Streams or the Real-time Transport Protocol.

## 2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including and amendments) applies.

ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 13818-1 | ITU-T Recommendation H.222.0, *Information technology — Generic coding of moving pictures and associated audio information: Systems*

ISO/IEC 13818-6, *Information technology — Generic coding of moving pictures and associated audio information — Part 6: Extensions for DSM-CC*

ISO/IEC 15938-5, *Information technology — Multimedia content description interface — Part 5: Multimedia description schemes*

ISO/IEC 21000 (all parts), *Information technology — Multimedia framework (MPEG-21)*

IETF RFC 3350, *RTP: A Transport Protocol for Real-Time Applications*, IETF Request For Comments: 3550, July 2003

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, IETF Request For Comments: 3986, January 2005

SMPTE 12M-1999, *Television, Audio and Film — Time and Control Code*

W3C DOM, *Document Object Model (DOM) Level 3 Core Specification*, W3C Recommendation, 7 April 2004

W3C XINCLUDE, *XML Inclusions (XInclude) Version 1.0*, W3C Recommendation, 20 December 2004

W3C XML, *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, W3C Recommendation, 16 August 2006

W3C XMLNAMES, *Namespaces in XML 1.0 (Second Edition)*, W3C Recommendation, 16 August 2006

W3C XPATH, *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 16 November 1999

## 3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

**3.1**
**Binding**
set of abstract Bitstream Binding Language processing instructions which map content of a particular arrangement (e.g. a particular content format, or a set of XML documents with similar structure) into a collection of Packets to be output to one or more Handlers

**3.2**
**Binary Document**
binary resource which can be processed using the Bitstream Binding Language via a BS Schema to describe the syntactical structure of the resource

**3.3**
**Delivery Time**
point in time when a Packet becomes available to an MPEG-21 Peer for consumption

**3.4**
**Document**
either an XML Document, as defined by W3C XML, or a Binary Document

**3.5**
**Document Fragment**
XML Document Fragment, as defined by W3C DOM

**3.6**
**Duration**
length of time for which an object is active

NOTE In BBL, individual Elements may be assigned Duration, which aggregate to provide the Duration of a Packet. If an Element has no explicitly associated Duration, it is assumed to have a Duration of zero.

**3.7**
**Encoder**
plug-in object to a BBL processor that may be used to include encoded content in a Packet

**3.8**
**Element**
XML element, as defined by W3C XML

**3.9**
**Fragmentation**
authoring process by which a Digital Item is split into Packets meaningful for consumption purposes

NOTE This process can attach time information to the output Packets indicating the point in time when they become available to the MPEG-21 Peer for consumption.

**3.10**
**Fragmentation Constraint**
rule which constrains the way in which content is fragmented

EXAMPLE One possible constraint is that a particular Element and its subtree should be “unbroken” — i.e. transmitted in a single Packet without Fragmentation.

**3.11**
**Fragmentation Rule**
criteria which defines conditions that a content fragment shall attempt to fulfil

EXAMPLE An example of this is a maximum size (in bytes) for any fragment.

NOTE See 7.5.2 for situations when a Fragmentation Rule cannot be fulfilled.

**3.12**
**Handler**
plug-in object to a BBL processor which receives a byte array representing the content of a Packet, along with the Delivery Time and Repetition Period for the Packet, and outputs the Packet to a delivery channel

NOTE The process can be guided by parameters provided either at a global or local level.

**3.13**
**Instance**
BBL document which processes one or more concrete Source Documents to form a collection of Packets which are output to one or more Handlers

NOTE If required, BBL Instances refer to BBL Bindings to facilitate this processing.

**3.14**
**map**
process of fragmenting a Source Document, assigning temporal and other parameters to fragments, and inserting the fragments into a specific location of an outgoing stream

**3.15**
**Packet**
fragment of a Digital Item (DID, metadata and/or resources), which is delivered to an MPEG-21 Peer, and to which temporal information (e.g. Delivery Time) may be attached

**3.16**
**Packet Stream**
sequence of Packets which possess time-invariant properties such that BBL is able to operate upon the sequence as a whole

**3.17**
**Processable**
⟨Packet⟩ not reliant upon Packets which have not yet been received

**3.18**
**Repetition Period**
period of time elapsing between repeated availability of a Packet to an MPEG-21 Peer for consumption (starting at Delivery Time)

**3.19**
**Source Document**
XML Document or Binary Document against which W3C XPATH expressions in BBL attributes are evaluated

NOTE In the case of a Binary Document, W3C XPATH expressions are evaluated via a BS Schema which has been associated with the namespace(s) used in the W3C XPATH expression.

## 4 Abbreviated terms

For the purposes of this document, the following abbreviations apply.

| | |
|---|---|
| **BBL** | Bitstream Binding Language |
| **BiM** | Binary MPEG format for XML (ISO/IEC 23001-1 [4]) |
| **BS Schema** | Bitstream Syntax Schema |
| **BSD** | Bitstream Syntax Description |
| **BSDL** | Bitstream Syntax Description Language |
| **DI** | Digital Item |
| **DID** | Digital Item Declaration |
| **DIS** | Digital Item Streaming |
| **DIDL** | Digital Item Declaration Language |
| **IPMP** | Intellectual Property Management and Protection |
| **MIME** | Multipurpose Internet Mail Extensions (IETF RFC 2045) |
| **MPEG** | Moving Picture Experts Group |
| **MPEG-2 TS** | MPEG-2 Transport Stream |
| **MPEG-2** | ISO/IEC 13818 |
| **MPEG-21** | ISO/IEC 21000 |
| **MPEG-4** | ISO/IEC 14496 |
| **MPEG-7** | ISO/IEC 15938 |
| **RTP** | Real-time Transport Protocol (IETF RFC 3550) |
| **SMPTE** | Society of Motion Picture and Television Engineers |
| **TCP** | Transmission Control Protocol (IETF RFC 793) |
| **UDP** | User Datagram Protocol (IETF RFC 768) |
| **UMA** | Universal Multimedia Access |
| **URI** | Uniform Resource Identifier (IETF RFC 3986) |
| **URL** | Uniform Resource Locator (IETF RFC 3986) |
| **URN** | Uniform Resource Name (IETF RFC 3986) |
| **W3C** | World Wide Web Consortium |
| **XML** | Extensible Markup Language (W3C XML) |

# 5 Conventions

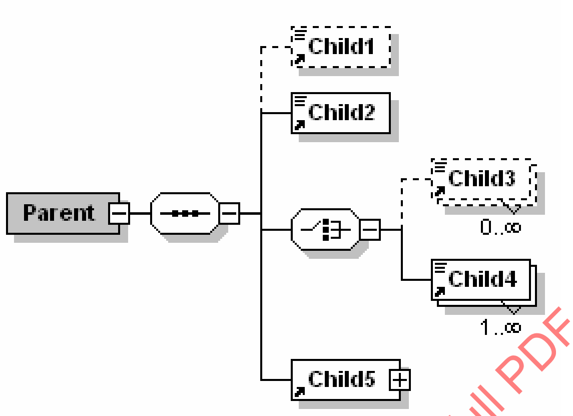
## 5.1 Documentation conventions

The syntax of each element in the Bitstream Binding Language is specified using the constructs provided by W3C XMLSCHEMA.

Element names and attribute names in the representation are in `this typeface`.

The syntax of each element in the Bitstream Binding Language is specified using the following format.

**Table 1 — Example element specification**

| | | | |
|---|---|---|---|
| **Diagram** |  Parent<br>Child1<br>Child2<br>Child3<br>0..∞<br>Child4<br>1..∞<br>Child5 | | |
| **Children** | <Child1> <Child2> <Child3> <Child4> <Child5> | | |
| **Used by** | <GrandParent1> <GrandParent2> | | |
| **Attributes** | **Name** | **Type** | **Description** |
| | ID | ID | A unique ID value, which can be referenced by another element. |
| **Source** | `<xs:element name="Parent">`<br>`  <xs:complexType>`<br>`    <xs:sequence>`<br>`      <xs:element ref="Child1" minOccurs="0"/>`<br>`      <xs:element ref="Child2"/>`<br>`      <xs:choice>`<br>`        <xs:element ref="Child3" minOccurs="0" maxOccurs="unbounded"/>`<br>`        <xs:element ref="Child4" maxOccurs="unbounded"/>`<br>`      </xs:choice>`<br>`      <xs:element ref="Child5"/>`<br>`    </xs:sequence>`<br>`    <xs:attribute name="ID" type="xs:ID"/>`<br>`  </xs:complexType>`<br>`</xs:element>` | | |

The language definition clause contains syntax diagrams for each element. Here is an example syntax diagram with annotations.

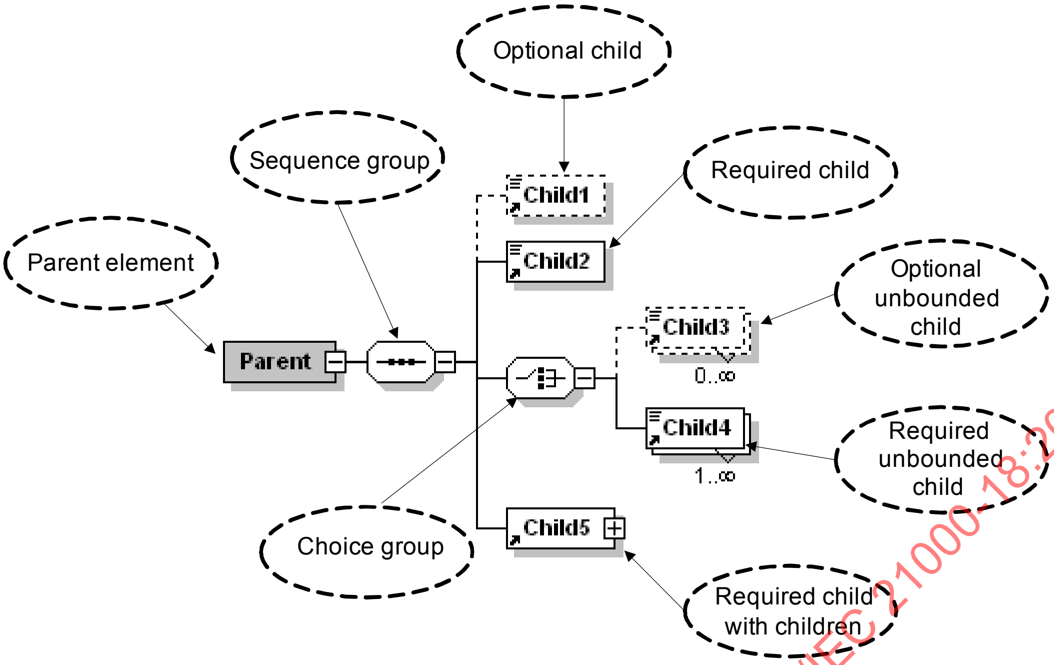


**Figure 1 — Example element syntax diagram**

Non-normative examples are shown in this document using a separate font and background.

EXAMPLE

```
<Example attribute1="example attribute value">
            <Element1>example element content</Element1>
</Example>
```

## 5.2 Namespace prefix conventions

This document makes use of vocabularies from several XML namespaces (where the definition of an XML namespace is as specified in W3C XMLNAMES). Qualified Names are written with a namespace prefix followed by a colon followed by the local part of the Qualified Name as shown in the following example.

EXAMPLE `bbl:BBL`

For the purposes of this document Table 2 gives the namespace prefixes associated with the identified namespaces. Where no prefix is used the default namespace is `urn:mpeg:mpeg21:2007:01-BBL-NS`.

**Table 2 — Namespace prefixes**

| Namespace prefix | Namespace |
|---|---|
| bbl | urn:mpeg:mpeg21:2007:01-BBL-NS |
| did | urn:mpeg:mpeg21:2002:02-DIDL-NS |
| dia | urn:mpeg:mpeg21:2003:01-DIA-NS |
| Hrtp | urn:mpeg:mpeg21:2007:01-BBL-NS:handler:RTP |
| Hmpg2 | urn:mpeg:mpeg21:2007:01-BBL-NS:handler:MPEG2TS |
| xml | http://www.w3.org/XML/1998/namespace |
| xmlns | http://www.w3.org/2000/xmlns |
| xi | http://www.w3.org/2001/XInclude |
| xs | http://www.w3.org/2001/XMLSchema |
| xsi | http://www.w3.org/2001/XMLSchema-instance |
| NOTE The prefixes `xml` and `xmlns` are normatively defined by W3C XMLNAMES. All other prefixes are not normative and are used by convention for consistency in this document. | |

## 6 Relationship to other ISO/IEC 21000 parts

The Digital Item is the fundamental unit of distribution and transaction in the Multimedia framework. While the different parts of ISO/IEC 21000 deal with the components and different aspects of Digital Items, together they form a complete integrated interoperable framework. This clause describes the relationship of this part of ISO/IEC 21000 with the other parts of ISO/IEC 21000 in addressing the specific function of Digital Item Streaming.

The Bitstream Binding Language is designed to enable Digital Item Streaming. Consequently, Digital Item Declarations declared according to the DIDL specified by part 2 of ISO/IE 21000 can be streamed using this tool. Equally IPMP Digital Item Declarations declared according to part 4 of ISO/IEC 21000 can be streamed using this tool. Both of the above Digital Item Declarations can contain or refer to XML conforming to other parts of ISO/IEC 21000 – such as DIA descriptions (part 7) and Rights Expressions (part 5). Such XML can be streamed using the Bitstream Binding Language, whether it is embedded in a DID, referenced by a DID, or used separately from a DID.

The Bitstream Binding Language uses BS Schema and BS Description from part 7 of ISO/IEC 21000 to specify the structure of binary resources which are to be streamed using BBL. This enables fragments of a binary resource to be specified using W3C XPATH references. See 7.2.3 for further information.

Part 7 of ISO/IEC 21000 specifies the XML streaming instructions and media resource streaming instructions that are optimized for use with (g)BSD, AdaptationQoS, and the described media. Hence the XML streaming and media resource streaming instructions should be used in these contexts.

## 7 Bitstream Binding Language

### 7.1 Bitstream Binding Language Introduction

The purpose of this clause is to describe the syntax and semantics of the Bitstream Binding Language (BBL). The syntax is defined using XML schema (as specified in W3C XMLSCHEMA). The goal is to be as flexible and general as possible, so that it may be used for the description of streaming instructions for a Digital Item no matter what its content. For the purposes of this document, the XML schema syntax descriptions are collectively referred to as BBL schema.

## 7.2 Overview

### 7.2.1 BBL

The Bitstream Binding Language (BBL) enables the description of how a Digital Item – comprising XML and binary content – can be fragmented and inserted (mapped) into one of several delivery channels (Figure 2). Just as a Digital Item is generic insofar as the XML that is stored in or referenced from the DID and the types of resources that may be attached, BBL is agnostic to the format of the data it is able to process, as well as the type of delivery channels to be output.

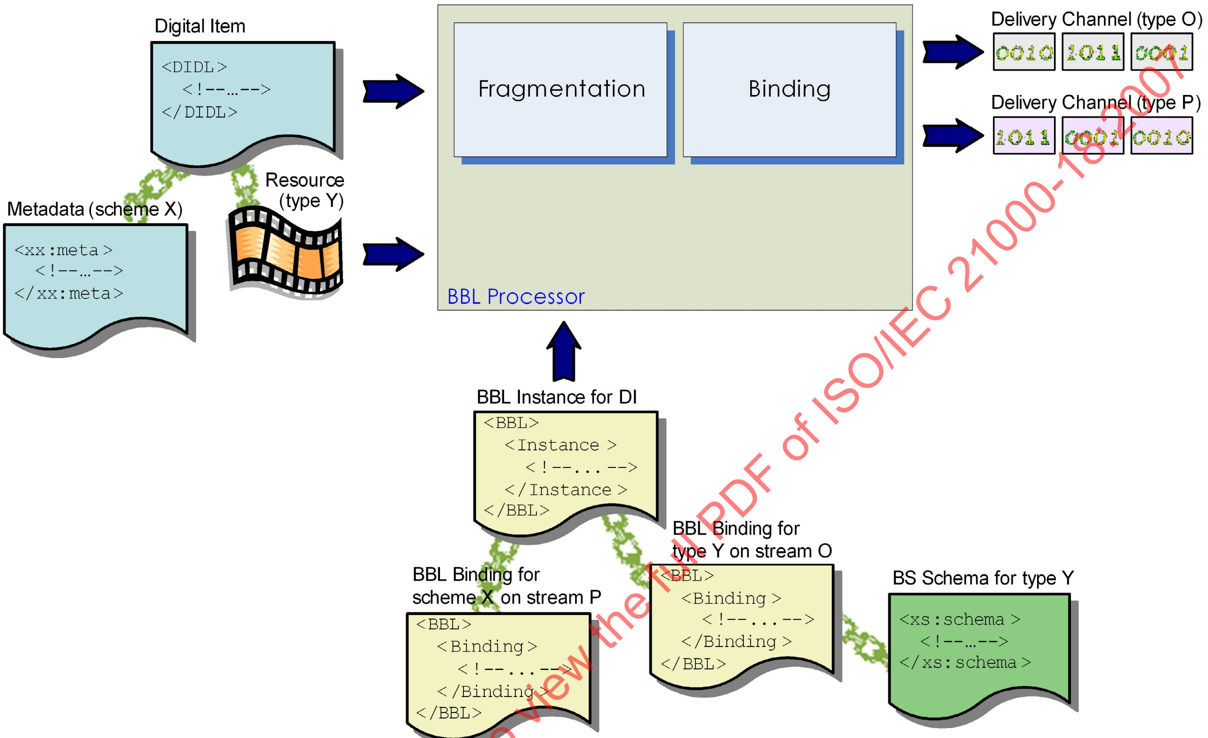


**Figure 2 — The BBL document model**

In this way, BBL provides an abstraction layer between a stored Digital Item and its representation in a specific channel. This enables different bitstreams to be created from a single DI to provide

— different views/subsets of the DI,

— different renderings of the content, and

— different bitstream formats,

facilitating a Universal Multimedia Access (UMA) solution to the serialization of Digital Items.

Different parts of a DI can be sent over separate channels.

EXAMPLE In the internet domain, portions of the DI requiring reliable delivery can be sent using, for example, TCP, while others which rely on timely delivery can be sent via RTP.

In BBL, each channel to be used is associated with a Handler (see 7.3.19). A BBL processor will provide in a Handler the functionality required for operating the delivery channel with which it is associated. Additionally, using a BS Schema, declarative behaviour for any particular channel (such as the contents of header fields) can be specified as part of the BBL (see C.3 for an example involving RTP headers).

BBL is designed for maximum reusability, so that (in the example, Figure 2) the binding of resources of type Y to delivery channels of type O can be specified once, and referenced by all DIs using Y and O. In the same way, when metadata is presented in a standard format (Scheme X) for a group of Digital Items, as single BBL Binding can be written for that metadata format and applied to all of the DIs.

### 7.2.2 BBL documents

There are two basic types of BBL document. The first is an Instance (see 7.3.10), which describes the streaming instructions for a specific Digital Item, and contains references to the DID and any other resources of the DI (these references are either URIs, or pointers to the location of the URI within the DID or other document).

The second type of BBL document is a Binding (see 7.3.10), which is an abstract mapping from some Digital Item or part thereof to a particular set of delivery channels. Bindings are instantiated as part of an Instance document, which provides the URI references for the concrete content to which the Binding(s) are to be applied.

As shown above (Figure 2), a Binding could be written which maps all resources of a particular format (e.g. AAC) to a particular delivery channel (such as RTP). While in this case such mappings might already exist (i.e. RFC 3640 [1]), a BBL Binding provides a programmatic method for specifying a mapping such that a BBL processor is able to directly transmit the desired content. In this way, as new mappings are developed, they can be created once and used by any BBL processor.

### 7.2.3 Processing of binary resources

The Bitstream Syntax Description Language (BSDL, see part 7 of ISO/IEC 21000) is used by BBL to provide an abstraction layer that allows binary resources to be handled in BBL in exactly the same way as XML.

EXAMPLE 1 If the author of a BBL document wishes to include an MPEG-4 Video Elementary Stream VOL object in a particular fragment, they do so using a W3C XPATH expression (as shown, Figure 3). The mp4 prefix is declared (as per W3C XMLNAMES) to be associated with a namespace for which the BBL processor has been previously informed is a binary namespace (and associated with a referenced BS Schema, see 7.3.23), and so invokes the BSDL parser on the currently active binary object to output the appropriate binary data.

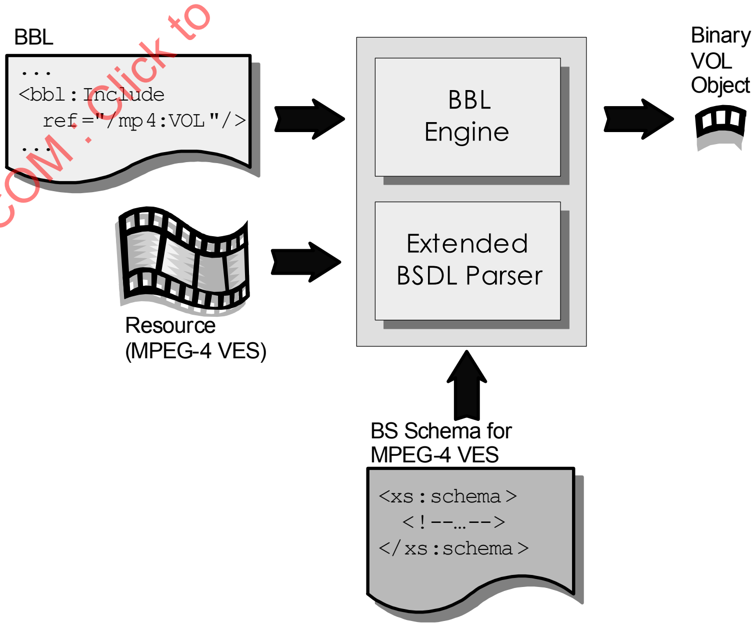


**Figure 3 — XML-based selection of binary content**

If required, pre-existing BS Descriptions can be treated in a similar manner, by declaring the namespace(s) used in the BSD as binary namespaces. This process is discussed in detail in 7.3.31.

EXAMPLE 2 A BSD can be used directly in case in which no BS Schema is available or possible, such as for a raw media stream consisting only of a concatenation of frames.

## 7.3 Bitstream Binding Language definition

### 7.3.1 Introduction

The following subclauses provide a formal definition of the syntax and semantics for the BBL Elements and Attributes, and provide requirements relating to the use of W3C XPATH expressions within BBL attributes.

### 7.3.2 Validation

Validating a document against the BBL schema (as specified in W3C XMLSCHEMA) is necessary, but not sufficient, to determine its validity with respect to BBL. After a document is validated against the BBL schema, it shall also be subjected to additional validation rules. These additional rules are given below in the descriptions of the elements to which they pertain.

NOTE While validation is required to ensure a valid BBL document is being used, an implementation could validate the document once and it can then use that document without needing to validate it. Re-validation would only be required if the document gets modified.

### 7.3.3 Document modularity

Modularity of BBL documents can be achieved by utilising XML Inclusions (XInclude) as specified in W3C XINCLUDE. XInclude can be used to reference elements within a document, or to elements in a different document. The former type of reference is known as an internal reference, the latter is known as an external reference. An internal reference allows a single source to be maintained for an element that occurs in more than one place in a BBL document. An external reference allows a BBL document to be split up into multiple linked discrete documents.

### 7.3.4 Use of "?" to delimit XPath expressions within attribute values

Where data is provided to a BBL description that already exists within another document – for example a URI reference to a binary or XML resource – a mechanism is required to allow the field within the BBL to contain a pointer to the actual value. For this purpose, W3C XPATH expressions delimited by "?" symbols may be used in certain attribute or element values which otherwise expect a literal value. These attributes are

— `xmlSource` (see 7.3.6),

— `binarySource` (see 7.3.6),

— `binding` (see 7.3.11),

— `bsSchemaLocation` (see 7.3.23), and

— the children of a `handler` or `handlerParams` element (see 7.3.19 and 7.3.25).

See 7.3.6 for an example of the use of delimited W3C XPATH expressions. In the example, the value of the `timeBase` parameter is found by evaluating the variable `$rtp:timeBase`.

### 7.3.5 Use of variables and functions within XPath expressions

W3C XPATH supports the use of variables and custom-defined functions. These functions can dramatically improve the efficiency of BBL processing, as commonly used values (for example a timebase) can be stored in a variable once, then subsequent accesses to the variable value are several orders of magnitude faster than repeated resolution of node-test expressions.

BBL provides a normative variable which is used to access data from the underlying processing model so that it may be included in output Packets. This variable is specified in the table below.

**Table 3 — BBL XPath variables**

| Variable name | Semantics |
|---|---|
| `$bbl:addr` | When used within an attribute, this variable shall evaluate to an absolute and unambiguous XPath address of the element to which the attribute is attached in the XML Source Document from which the element was extracted. This variable is designed to be used with the `bbl:context` attribute to provide an MPEG-21 receiving peer with the ability to reconstruct a DI from its fragments. |

The scope of any variable is always from the point at which it is defined to the end of the containing Instance/Binding element.

BBL also provides several normative functions for accessing data from the processing model. These are specified in the table below

**Table 4 — BBL XPath functions**

| Function name | Semantics |
|---|---|
| `bbl:deliveryTime(id)` | Returns an integer representing the Delivery Time (using the in-scope time scheme, see 7.3.7) of the `Packet` or `PacketStream` referenced by the id corresponding to the value of the id parameter. In the case of a `PacketStream`, the Delivery Time of the first Packet in the stream is returned. |
| `bbl:duration(id)` | Returns an integer representing the Duration (using the in-scope time scheme, see 7.3.7) of the `Packet` or `PacketStream` referenced by the id corresponding to the value of the id parameter. In the case of a `PacketStream` the Duration of the most recently transmitted Packet is returned. |
| `bbl:finalDeliveryTime(id)` | Returns an integer representing the Delivery Time (using the in-scope time scheme, see 7.3.7) of the `Packet` or `PacketStream` referenced by the id corresponding to the value of the id parameter. In the case of a `Packet`, this function returns a value identical to that returned by `bbl:deliveryTime()`. |

These functions are particularly useful to specify the Delivery Time of a Packet or Packet Stream relative to the Delivery Time of another Packet/Packet Stream.

### 7.3.6 `xmlSource` and `binarySource` attributes

BBL provides a hierarchical mechanism for declaring Source Documents identical to that used by W3C XMLNAMES. At any element, the in-scope XML or Binary Source Document shall be that referenced by the first `xmlSource` or `binarySource` attribute (respectively) encountered by searching the ancestor-or-self axis of the element (see W3C XPATH), in order of decreasing proximity to the element (i.e. first the element itself, then its parent, then its grandparent, and so on to the root node).

When W3C XPATH expressions are encountered within an `xmlSource` attribute, they shall be resolved as above with the exception that the search proceeds along the ancestor axis of the element, in order to prevent a circular reference.

When W3C XPATH expressions utilize a BSD namespace (as registered by a `BSD` element with value of true for its `fromBinarySource` attribute, see 7.3.23) they shall be resolved against the in-scope Binary Source Document. Otherwise they shall be resolved against the in-scope XML Source Document.

Because BBL Bindings (see 7.3.11) are abstract documents, they are not required to declare Source Documents (although they may). When a `Binding` is instantiated by a `Bind` element, the in-scope Source Documents shall be resolved as if the `Binding` element and its subtree were attached to the `Bind` as a child.

**Validation Rule:**

— Any attribute which takes as its value a W3C XPATH expression shall have an in-scope Source Document (XML or binary as appropriate), unless

  — the W3C XPATH expression does not contain any node tests, or

  — the attribute is part of an element which has a `Binding` ancestor.

EXAMPLE This example (see Figure 4) illustrates the behaviour of xmlSource and binarySource. Each W3C XPATH expression is indicated with a letter. The contexts of these expressions are listed below.

a) Because this W3C XPATH expression is contained within an `xmlSource` attribute, it is resolved against the XML Document referenced by the first `xmlSource` attribute encountered when searching the ancestors of this node. In this case this is did_foreman.xml which is declared by the `Instance` element.

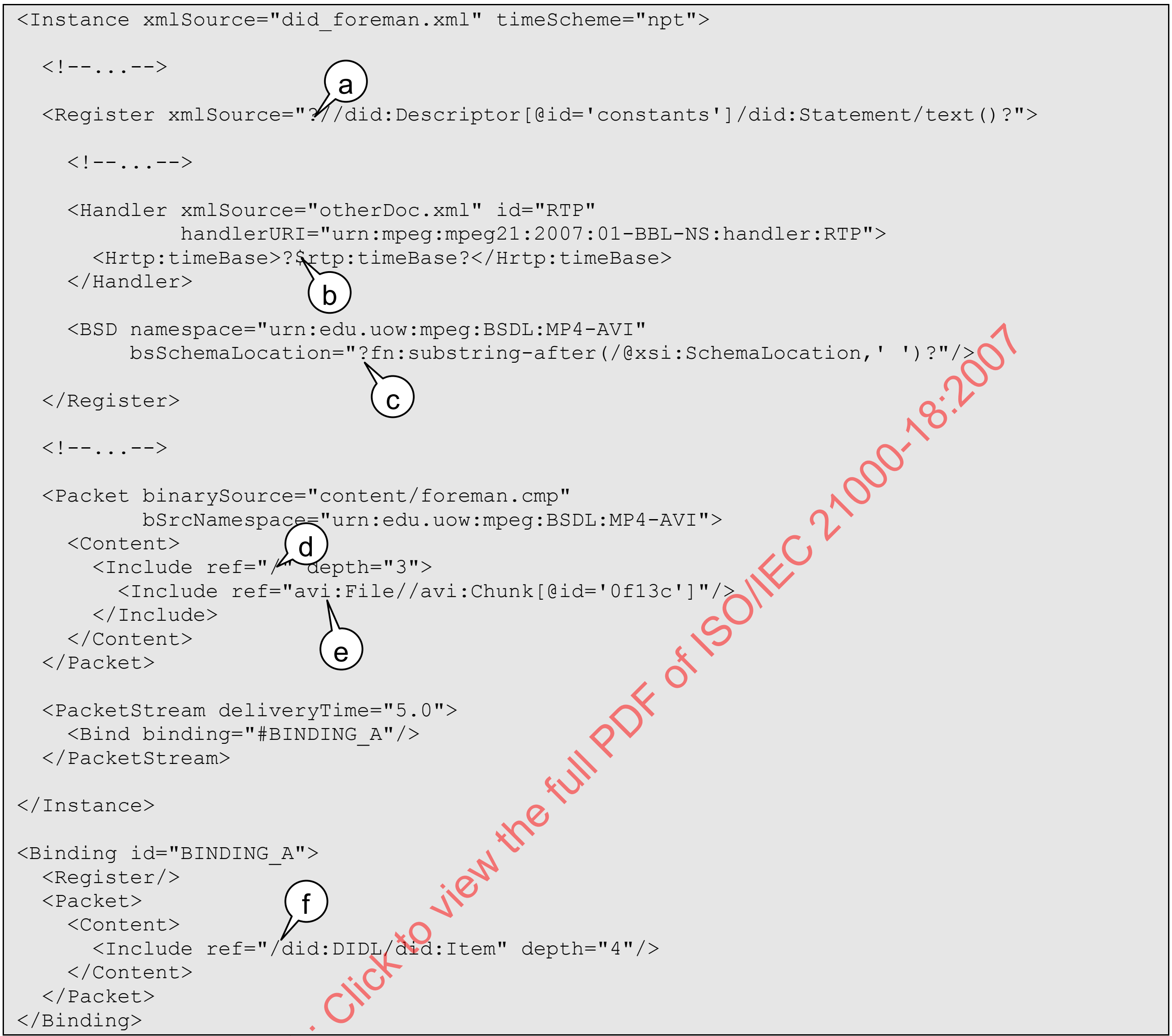
b) The XML Document against which this expression is resolved is that declared on the element (otherDoc.xml). However, the expression in fact makes no reference to any node-tests, so the context has no effect.

c) This expression is resolved against the XML Document referenced by the first `xmlSource` attribute encountered when searching first the node itself, and then the ancestors of the node. Here, this is the Document whose URI is the value returned by a).

d) The XML Document against which this expression is resolved is did_foreman.xml.

e) Because urn:edu.uow:mpeg:BSDL:MP4-AVI has been declared as a BSD namespace, this W3C XPATH expression is resolved against the Binary Document content/foreman.cmp. This is done by inferring the necessary BSD for the binary content using the declared schema (whose location was resolved by the W3C XPATH expression in c)).

f) The XML Document against which this expression is resolved is found by attaching the parent `Binding` element to the `Bind` from which it is referenced. It is thus did_foreman.xml.



```
<Instance xmlSource="did_foreman.xml" timeScheme="npt">

  <!--...-->

  <Register xmlSource="?//did:Descriptor[@id='constants']/did:Statement/text()?">

    <!--...-->

    <Handler xmlSource="otherDoc.xml" id="RTP"
             handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:RTP">
      <Hrtp:timeBase>?$rtp:timeBase?</Hrtp:timeBase>
    </Handler>

    <BSD namespace="urn:edu.uow:mpeg:BSDL:MP4-AVI"
         bsSchemaLocation="?fn:substring-after(/@xsi:SchemaLocation,' ')?"/>

  </Register>

  <!--...-->

  <Packet binarySource="content/foreman.cmp"
          bSrcNamespace="urn:edu.uow:mpeg:BSDL:MP4-AVI">
    <Content>
      <Include ref="/" depth="3">
        <Include ref="avi:File//avi:Chunk[@id='0f13c']"/>
      </Include>
    </Content>
  </Packet>

  <PacketStream deliveryTime="5.0">
    <Bind binding="#BINDING_A"/>
  </PacketStream>

</Instance>

<Binding id="BINDING_A">
  <Register/>
  <Packet>
    <Content>
      <Include ref="/did:DIDL/did:Item" depth="4"/>
    </Content>
  </Packet>
</Binding>
```

**Figure 4 — BBL fragment - `xmlSource`, `binarySource` example (informative)**

### 7.3.7 `timeScheme` attribute

BBL allows time values to be provided in a number of different time schemes (see Table below). The time schemes that may be used correspond to those defined in ISO/IEC 21000-17 (FID). Multiple time schemes may be used and a hierarchical mechanism for declaring the in-scope time scheme is provided. At any element or attribute that contains a time value (or variable or function that returns a time value), the in-scope time scheme shall be that referenced by the first `timeScheme` attribute encountered by searching the ancestor-or-self axis (see W3C XPATH), in order of decreasing proximity to the element or attribute (i.e. first the element or attribute itself, then its parent, then its grandparent, and so on to the root node).

If no declared in-scope time scheme is found, the default time scheme shall be `frac(1000)`.

**Table 5 — Time schemes**

| Time scheme | Description |
|---|---|
| `npt` | Time values are provided as Normal Play Time as defined in ISO/IEC 13818-6 (DSM-CC). |
| `smpte-24`<br>`smpte-24-drop`<br>`smpte-25`<br>`smpte-30`<br>`smpte-30-drop`<br>`smpte-50`<br>`smpte-60`<br>`smpte-60-drop` | Time values are provided as time codes as specified in the SMPTE time and control code standard SMPTE 12M-1999. |
| `mp7t` | Time values are provided conforming to MPEG-7 MediaTimePointType as defined in ISO/IEC 15938-5 (MDS). |
| `clock` | Time values are provided as Universal Time Codes which gives wall-clock time as defined by ISO 8601. |
| `frac(N)` | Time values are provided in fractions of a second where N is a positive integer indicating the number of fractions per second.<br>EXAMPLE A time scheme `frac(1000)` indicates time values are provided in 1/1000ths of a second (i.e. milliseconds). |

NOTE Elements and attributes containing time values to which the in-scope time scheme applies include those listed in the table below.

**Table 6 — Timing attributes to which the in-scope time scheme applies**

| Attribute | Parent element(s) |
|---|---|
| `deliveryTime` | Packet (7.3.26), PacketStream (7.3.27) |
| `repeat` | Packet (7.3.26), PacketStream (7.3.27) |
| `value` | DeliveryTimes (7.3.34), Durations (7.3.35), Duration (7.3.38) |

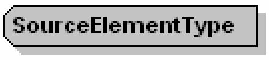
### 7.3.8 `SourceElementType` type

`xmlSource`, `binarySource`, and `timeScheme` attributes may be declared on most BBL elements (see 7.3.10 to 7.3.44). They are defined by the abstract complex type `SourceElementType`, from which the relevant BBL elements are extended.

**Validation Rule:**

— If the `binarySource` attribute is present, then `bSrcNamespace` attribute shall also be present, and vice versa.

**Table 7 — SourceElementType**

| | | | | |
|---|---|---|---|---|
| **Diagram** |  | | | |
| **Used by** | AbstractVar Bind Declarations IdentifiableElementType Include Register Timing Variables value-of | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | Specifies the XML Source Document to be used by W3C XPATH expressions on this element and its descendants. The value of this attribute shall be either a relative or absolute URI, or a "?" delimited W3C XPATH expression which resolves to a URI. See also 7.3.6 |
| | binarySource | xs:string | optional | Specifies the Binary Source Document to be used by elements of bs1:byteRange or bs1:bitstreamSegment type that are descendants of this element (whether declared within the BBL document or included), and by W3C XPATH expressions on this element and its descendants that utilize a BSD namespace. The value of this attribute shall be either a relative or absolute URI, or a "?" delimited W3C XPATH expression which resolves to a URI. See also 7.3.6. |
| | bSrcNamespace | xs:string | optional | Specifies the namespace with which the root element of the BS Schema describing the binarySource is associated. This namespace shall be registered as being associated with a BSD Namespace (see 7.3.23). If the binarySource attribute is present, this attribute shall also be present. |
| | timeScheme | xs:string | optional | Specifies the time scheme used for time values on this element and its descendants. See also 7.3.7. |

**Source**

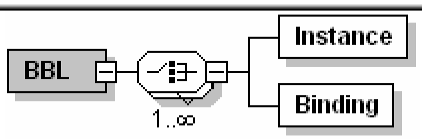
```
<xs:complexType name="SourceElementType" abstract="true">
  <xs:attribute name="xmlSource" type="xs:string" use="optional"/>
  <xs:attribute name="binarySource" type="xs:string" use="optional"/>
  <xs:attribute name="bSrcNamespace" type="xs:string" use="optional"/>
  <xs:attributeGroup ref="bbl:timeScheme"/>
</xs:complexType>
<xs:attributeGroup name="timeScheme">
  <xs:attribute name="timeScheme">
    <xs:simpleType>
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:NCName">
            <xs:enumeration value="npt"/>
            <xs:enumeration value="smpte-24"/>
            <xs:enumeration value="smpte-24-drop"/>
            <xs:enumeration value="smpte-25"/>
            <xs:enumeration value="smpte-30"/>
            <xs:enumeration value="smpte-30-drop"/>
            <xs:enumeration value="smpte-50"/>
            <xs:enumeration value="smpte-60"/>
            <xs:enumeration value="smpte-60-drop"/>
            <xs:enumeration value="mp7t"/>
            <xs:enumeration value="clock"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="frac\(\d+\)"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
  </xs:attribute>
</xs:attributeGroup>
```

### 7.3.9 BBL element

The BBL element is the root element of any BBL description. It contains one or more Instance and/or Binding elements.

The BBL element does not define any processing. All processing is defined by the children of the BBL element.

**Table 8 — BBL element**

| | |
|---|---|
| **Diagram** |  |
| **Children** | bbl:Instance bbl:Binding |
| **Source** | `<xs:element name="BBL">`<br>`  <xs:complexType>`<br>`    <xs:choice maxOccurs="unbounded">`<br>`      <xs:element name="Instance" type="bbl:InstanceType"/>`<br>`      <xs:element name="Binding" type="bbl:BindingType"/>`<br>`    </xs:choice>`<br>`  </xs:complexType>`<br>`</xs:element>` |

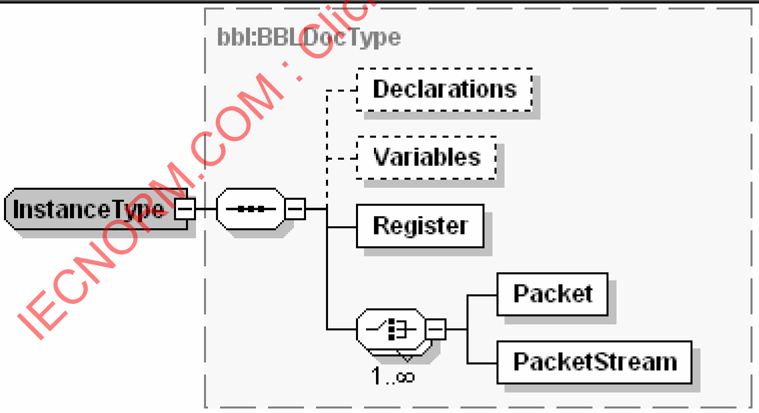
### 7.3.10 Instance element

The Instance element is of type InstanceType and declares a BBL Instance document. Instance documents describe the Fragmentation and binding of concrete Source Document(s), and may instantiate BBL Binding documents to do so.

Instance documents provide a scope for declared Variables and Handlers.

For an example of the use of this element, see C.2.

**Table 9 — InstanceType**

| | | | | |
|---|---|---|---|---|
| **Diagram** |  bbl:BBLDocType<br>Declarations<br>Variables<br>InstanceType<br>Register<br>Packet<br>PacketStream<br>1..∞ | | | |
| **Children** | bbl:Declarations bbl:Variables bbl:Register bbl:Packet bbl:PacketStream | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | id | xs:ID | optional | Optional attribute to identify this element. |

**Source**

```
<xs:complexType name="InstanceType">
  <xs:complexContent>
    <xs:extension base="bbl:BBLDocType"/>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BBLDocType">
  <xs:complexContent>
    <xs:extension base="bbl:IdentifiableElementType">
      <xs:sequence>
        <xs:element name="Declarations" minOccurs="0"
                    type="bbl:DeclarationsType"/>
        <xs:element name="Variables" minOccurs="0" type="bbl:VariablesType"/>
        <xs:element name="Register" type="bbl:RegisterType"/>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="Packet" type="bbl:PacketType"/>
          <xs:element name="PacketStream" type="bbl:PacketStreamType"/>
        </xs:choice>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="IdentifiableElementType">
  <xs:complexContent>
    <xs:extension base="bbl:SourceElementType">
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 7.3.11 `Binding` element

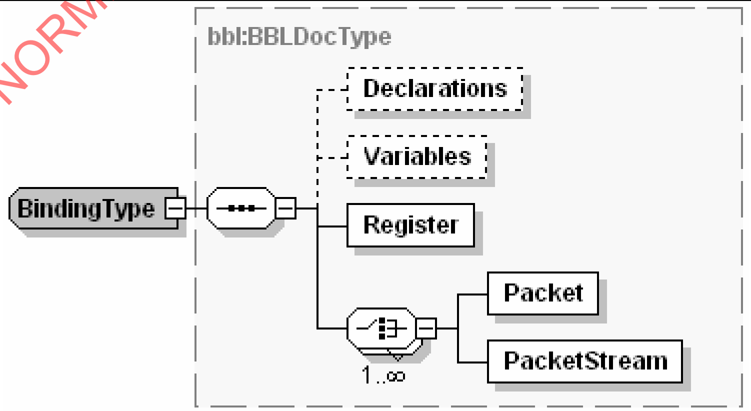
The `Binding` element is of type `BindingType` and declares a BBL Binding document. Binding documents describe an abstract Fragmentation and binding process which may be applied to multiple Source Documents.

A `Binding` is instantiated using a `Bind` element.

Binding documents provide a scope for declared variables and Handlers.

For an example of the use of this element, see 7.3.6.

**Table 10 — BindingType**

| | |
|---|---|
| **Diagram** |  bbl:BBLDocType<br>Declarations<br>Variables<br>BindingType<br>Register<br>Packet<br>PacketStream<br>1..∞ |
| **Children** | bbl:Declarations bbl:Variables bbl:Register bbl:Packet bbl:PacketStream |

| | Name | Type | Use | Description |
|---|---|---|---|---|
| Attributes | `xmlSource` | xs:string | optional | See `SourceElementType` (7.3.8) |
| | `binarySource` | xs:string | optional | |
| | `bSrcNamespace` | xs:string | optional | |
| | `id` | xs:ID | optional | See `InstanceType` (7.3.10) |

Source

```
<xs:complexType name="BindingType">
  <xs:complexContent>
    <xs:extension base="bbl:BBLDocType"/>
  </xs:complexContent>
</xs:complexType>
```

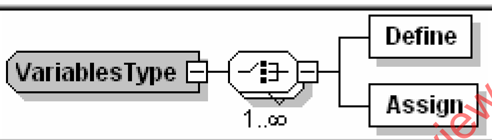
### 7.3.12 `Variables` element

The `Variables` element is of type `VariablesType` and contains variable definition and assignment instructions.

A `Variables` element may appear as a child of `Instance`, `Binding`, `Packet`, or `PacketStream`.

If the `Variables` element is the child of an `Instance` or `Binding` element, then it is processed at time 0. If, however, if it is the child of a `Packet` or `PacketStream`, the variable definitions and assignments are processed *at the same time as the packet is processed*.

**Table 11 — VariablesType**

| | | | | |
|---|---|---|---|---|
| Diagram |  | | | |
| Children | `bbl:Define bbl:Assign` | | | |
| Attributes | **Name** | **Type** | **Use** | **Description** |
| | `xmlSource` | xs:string | optional | See `SourceElementType` (7.3.8) |
| | `binarySource` | xs:string | optional | |
| | `bSrcNamespace` | xs:string | optional | |

Source

```
<xs:complexType name="VariablesType">
  <xs:complexContent>
    <xs:extension base="bbl:SourceElementType">
      <xs:choice maxOccurs="unbounded">
        <xs:element name="Define" type="bbl:DefineType"/>
        <xs:element name="Assign" type="bbl:AbstractVarType"/>
      </xs:choice>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 7.3.13 `AbstractVarType` type

The `AbstractVarType` type is a base type for elements concerned with variables such as the `Define` element (see 7.3.14) and the `Assign` element (see 7.3.15).

**Validation Rule:**

- The value of the name attribute shall not match any of those defined in 7.3.5.

**Table 12 — AbstractVarType**

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | bbl:name | xs:QName | required | The name of the variable to be defined. |
| | bbl:value | xs:string | required | A W3C XPATH expression which resolves to the value of the variable. The returned value shall match the variable type.<br>The context of the W3C XPATH expression varies according to the location of the containing Variables element. If the Variables element is declared by an Instance or Binding element, then the context of the W3C XPATH expression shall be the root node of the *in-scope XML Source Document*.<br>If the containing Variables element is declared by a Packet or PacketStream element, then the context shall be the root node of the content of *the current Packet*.<br>This is particularly important in the case of a PacketStream, where the content of the Packet will be a subset of that returned by the Include statement according to the declared Fragmentation Rules. See 7.3.31 for further information. |
| **Source** | `<xs:complexType name="AbstractVarType">`<br>`  <xs:complexContent>`<br>`    <xs:extension base="bbl:SourceElementType">`<br>`      <xs:attribute ref="bbl:name" use="required"/>`<br>`      <xs:attribute ref="bbl:value" use="required"/>`<br>`    </xs:extension>`<br>`  </xs:complexContent>`<br>`</xs:complexType>` | | | |

### 7.3.14 Define element

The Define element is of type DefineType and allows a variable to be declared so that it may be used in subsequent W3C XPATH expressions. Variables are declared by providing

- a variable name as a QName,
- an initial value, which will be equal to the result of the W3C XPATH expression contained by the value attribute, and
- optionally a type, which shall be one of the following:

**Table 13 — Allowed variable types**

| | | | |
|---|---|---|---|
| xs:integer | xs:int | xs:long | xs:short |
| xs:decimal | xs:float | xs:double | xs:Boolean |
| xs:byte | xs:QName | xs:dateTime | xs:base64Binary |
| xs:hexBinary | xs:unsingedInt | xs:unsignedShort | xs:unsignedByte |
| xs:time | xs:date | xs:anySimpleType | |

The location in time of a Define element is specified by its parent Variables element. See 7.3.12.

For an example of the use of this element, see C.2.

**Validation Rule:**

— For all variables other than those defined in 7.3.5, a Define element shall precede in time any reference to the variable by an Assign element or W3C XPATH expression.

**Table 14 — DefineType**

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | bbl:name | xs:QName | required | See AbstractVarType (7.3.13) |
| | bbl:value | xs:string | required | |
| | type | xs:QName | optional | The data type of the variable to be defined. The value of the type attribute shall match one of those listed in Table 13 — Allowed variable types. |
| **Source** | | | | |

```
  <xs:complexType name="DefineType">
    <xs:complexContent>
      <xs:extension base="bbl:AbstractVarType">
        <xs:attribute name="type" type="bbl:VarType" use="optional"
default="xs:int"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:simpleType name="VarType">
    <xs:restriction base="xs:QName">
      <xs:enumeration value="xs:integer"/>
      <xs:enumeration value="xs:int"/>
      <xs:enumeration value="xs:long"/>
      <xs:enumeration value="xs:short"/>
      <xs:enumeration value="xs:decimal"/>
      <xs:enumeration value="xs:float"/>
      <xs:enumeration value="xs:double"/>
      <xs:enumeration value="xs:boolean"/>
      <xs:enumeration value="xs:byte"/>
      <xs:enumeration value="xs:QName"/>
      <xs:enumeration value="xs:dateTime"/>
      <xs:enumeration value="xs:base64Binary"/>
      <xs:enumeration value="xs:hexBinary"/>
      <xs:enumeration value="xs:unsignedInt"/>
      <xs:enumeration value="xs:unsignedShort"/>
      <xs:enumeration value="xs:unsignedByte"/>
      <xs:enumeration value="xs:time"/>
      <xs:enumeration value="xs:date"/>
      <xs:enumeration value="xs:anySimpleType"/>
    </xs:restriction>
  </xs:simpleType>
```

### 7.3.15 **Assign** element

The Assign element is of type AbstractVarType and assigns a new value to any previously declared variable. This is done by providing the name of the variable and a W3C XPATH expression which resolves to the new value.

The location in time of an Assign element is specified by its parent Variables element. See 7.3.12.

For an example of the use of this element, see C.4.

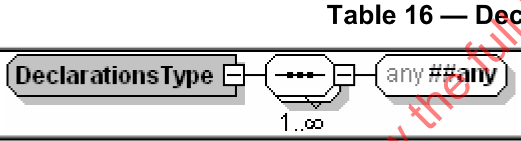
**Table 15 — Assign element**

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | bbl:name | xs:QName | required | See AbstractVarType (7.3.13) |
| | bbl:value | xs:string | required | |
| **Source** | See AbstractVarType (7.3.13) | | | |

### 7.3.16 Declarations element

The Declarations element is of type DeclarationsType and is used to define a set of BBL or other elements – without instantiating them – for later use in a document via an internal reference (see 7.3.3).

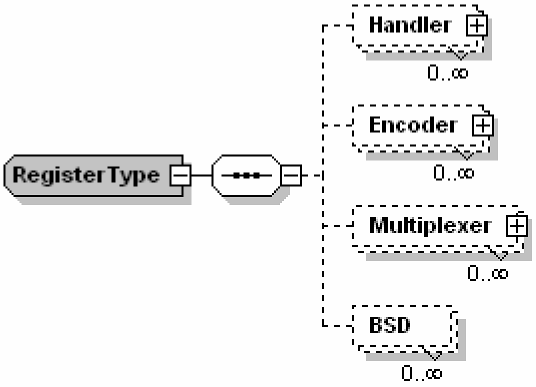
**Table 16 — DeclarationsType**

| | |
|---|---|
| **Diagram** |  |
| **Source** | `<xs:complexType name="DeclarationsType">`<br>`  <xs:sequence maxOccurs="unbounded">`<br>`    <xs:any namespace="##any" processContents="skip"/>`<br>`  </xs:sequence>`<br>`</xs:complexType>` |

### 7.3.17 Register element

The Register element is of type RegisterType and contains all of the Handler and BSD namespace declarations for an Instance or Binding.

**Table 17 — RegisterType**

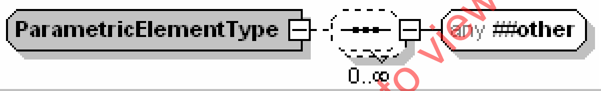
| | |
|---|---|
| **Diagram** |  RegisterType<br>Handler<br>0..∞<br>Encoder<br>0..∞<br>Multiplexer<br>0..∞<br>BSD<br>0..∞ |

| | | | | |
|---|---|---|---|---|
| **Children** | bbl:Handler bbl:Encoder bbl:Multiplexer bbl:Bsd | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| **Source** | `<xs:complexType name="RegisterType">`<br>`  <xs:complexContent>`<br>`    <xs:extension base="bbl:SourceElementType">`<br>`      <xs:sequence>`<br>`        <xs:element name="Handler" type="bbl:HandlerType"`<br>`                    minOccurs="0" maxOccurs="unbounded"/>`<br>`          <xs:element name="Encoder" type="bbl:EncoderType"`<br>`                    minOccurs="0" maxOccurs="unbounded"/>`<br>`          <xs:element name="Multiplexer" type="bbl:MultiplexerType"`<br>`                    minOccurs="0" maxOccurs="unbounded"/>`<br>`        <xs:element name="BSD" type="bbl:BSDType"`<br>`                    minOccurs="0" maxOccurs="unbounded"/>`<br>`      </xs:sequence>`<br>`    </xs:extension>`<br>`  </xs:complexContent>`<br>`</xs:complexType>` | | | |

### 7.3.18 `ParametricElementType` type

The ParametricElementType type is a base type for elements that allow child elements representing parameters such as the Handler element (see 7.3.19) and the HandlerParams element (see 7.3.25).

**Table 18 — ParametricElementType**

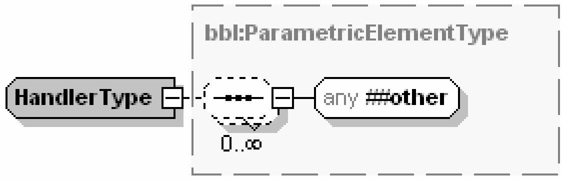
| | | | | |
|---|---|---|---|---|
| **Diagram** |  | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | id | xs:ID | optional | See InstanceType (7.3.10) |
| **Source** | `<xs:complexType name="ParametricElementType">`<br>`  <xs:complexContent>`<br>`    <xs:extension base="bbl:IdentifiableElementType">`<br>`      <xs:sequence minOccurs="0" maxOccurs="unbounded">`<br>`        <xs:any namespace="##other"/>`<br>`      </xs:sequence>`<br>`    </xs:extension>`<br>`  </xs:complexContent>`<br>`</xs:complexType>` | | | |

### 7.3.19 `Handler` element

The Handler element is of type HandlerType and registers a BBL Handler which represents a particular delivery channel in BBL.

A BBL Handler receives the byte content of each Packet output from the BBL process along with the Delivery Time and repeat parameters specified for the Packet. The Handler is then responsible for transmitting each Packet at the correct time(s) according to the rules of the associated delivery channel.

**Table 19 — HandlerType**

| | | | | |
|---|---|---|---|---|
| **Diagram** |  bbl:ParametricElementType<br>HandlerType<br>any ##other<br>0..∞ | | | |
| **Used by** | Register | | | |
| **Children** | The children of this element are provided as input parameters to the specified Handler. Permissible elements are specified by individual Handlers. | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | id | xs:ID | optional | The Handler ID. BBL elements make reference to this id to indicate that it is to be used to handle that element. |
| | handlerURI | xs:anyURI | required | The URI identifying the Handler to be used. The value of this attribute shall correspond to one of the Handlers specified in this document, or to a URI registered with the Registration Authority according to the procedure described in Annex D. |
| **Source** | `<xs:complexType name="HandlerType">`<br>`  <xs:complexContent>`<br>`    <xs:extension base="bbl:ParametricElementType">`<br>`      <xs:attribute name="handlerURI" type="xs:anyURI" use="required"/>`<br>`    </xs:extension>`<br>`  </xs:complexContent>`<br>`</xs:complexType>` | | | |

### 7.3.20 Encoder element

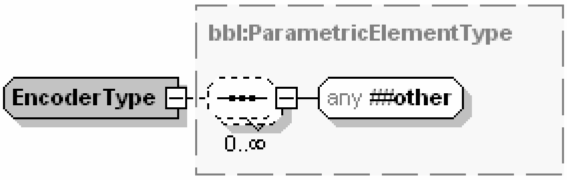
An Encoder element is of type EncoderType and registers a BBL Encoder which allows content to be encoded according to some desired scheme.

An Encoder receives an XML or binary fragment as input and returns a sequence of bytes representing the encoded version of the input content.

Permissible values for the encoderURI attribute are defined by the Registration Authority according to the procedures described in Annex D.

NOTE Examples of possible Encoders are TeM, BiM or other compression, an encryption algorithm, or a transcoder.

### Table 20 — EncoderType

| | | | | |
|---|---|---|---|---|
| **Diagram** |  bbl:ParametricElementType<br>EncoderType<br>any ##other<br>0..∞ | | | |
| **Used by** | `Register` | | | |
| **Children** | The children of this element are provided as input parameters to the specified Handler. Permissible elements are specified by individual Handlers. | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | `xmlSource` | xs:string | optional | See `SourceElementType` (7.3.8) |
| | `binarySource` | xs:string | optional | |
| | `bSrcNamespace` | xs:string | optional | |
| | `id` | xs:ID | optional | The Encoder ID. BBL elements make reference to this id to indicate that it is to be used to encode. content to be included in a Packet. |
| | `encoderURI` | xs:anyURI | required | The URI identifying the Encoder to be used. The value of this attribute shall correspond to one of the Encoders specified in this document, or to a URI registered with the Registration Authority according to the procedure described in Annex D. |
| **Source** | `<xs:complexType name="EncoderType">`<br>`<xs:complexContent>`<br>`<xs:extension base="bbl:ParametricElementType">`<br>`<xs:attribute name="encoderURI" type="xs:anyURI" use="required"/>`<br>`</xs:extension>`<br>`</xs:complexContent>`<br>`</xs:complexType>` | | | |

### 7.3.21 `Multiplexer` element

The `Multiplexer` element is of type `MultiplexerType` and registers a BBL `Multiplexer` that multiplexes `Packet`s into a BBL `Handler`.

A `Multiplexer` defines a logical layer between a BBL `Handler` and the `Packet`s that allows for prioritization of `Packet`s. A `Multiplexer` may contain zero or more `Channel`s or `Multiplexer`s. The `Multiplexer` enables the User to prioritize information in the bandwidth- domain instead of in the time-domain. Hence, it is an alternative to time-schemes. i.e., use of the `Multiplexer` will enable the User to expose related information to the same bandwidth. `Packet`s are assigned to sibling `Channel`s and/or `Multiplexer`s in the root `Multiplexer` as `Packet`s are generated.

EXAMPLE A structure with four channels is shown in the XML below and in the figure below. Channels c and d combined share as much bandwidth as channel b, which has half that of channel a.

```
<Multiplexer mode="bandwidth" bbl:handler="RTP">
  <Channel id="a" weight="20"/>
  <Channel id="b" weight="10"/>
  <Multiplexer mode="bandwidth" weight="10">
    <Channel id="c" weight="1"/>
    <Channel id="d" weight="1"/>
  </Multiplexer>
</Multiplexer>
```

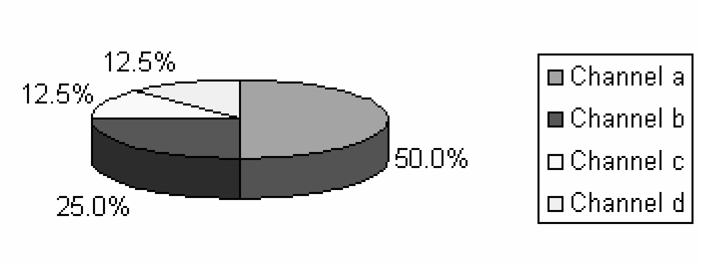
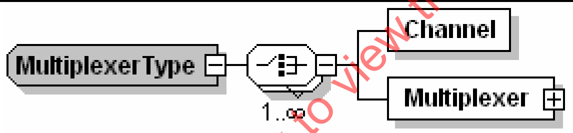


**Figure 5 — Example of Prioritizing of Channels in a Multiplexer**

**Validation Rule:**

- The ID referenced by the `bbl:handler` attribute shall be attached to a `Handler` element within the same `Instance` or `Binding` as this element or within a BBL Instance or Binding document to which the document containing this element is bound.
- If a `Multiplexer` element contains descendant `Multiplexer` elements, the value of the `bbl:handler` attribute on all descendant `Multiplexer` elements (where present) shall be equal to the value of the `bbl:handler` attribute on the ancestor element.
- If more than one `Multiplexer` element is instantiated as a child of a `Register` element, each such `Multiplexer` element shall have a unique value for the `bbl:handler` attribute (if present).
- At most one `Multiplexer` element that is a child of a `Register` element shall have the `bbl:handler` attribute not present.

**Table 21 — MultiplexerType**

| | | | | |
|---|---|---|---|---|
| **Diagram** |  | | | |
| **Used by** | `Register`, `Multiplexer` | | | |
| **Children** | `Channel`, `Multiplexer` | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | `bbl:handler` | xs:IDREF | optional | A reference to the Handler to which this Multiplexer is to be applied.<br>If not present and the `Multiplexer` element is a descendant of another `Multiplexer` element, then the `Handler` referenced by the nearest ancestor `Multiplexer` element shall be used.<br>If not present and the `Multiplexer` element is not a descendant of another `Multiplexer` element, then the first registered `Handler` shall be used |
| | `mode` | xs:sting | required | Defines whether the multiplexer measures its throughput by packet count or by bandwidth. The value of this attribute shall be one of:<br>— `packetCount`<br>— `bandwidth` |
| | `weight` | xs:int | optional | See `ChannelType` (7.3.22). |

| | |
|---|---|
| **Source** | `<xs:complexType name="MultiplexerType">`<br>`  <xs:choice maxOccurs="unbounded">`<br>`    <xs:element name="Channel" type="bbl:ChannelType"/>`<br>`    <xs:element name="Multiplexer" type="bbl:MultiplexerType"/>`<br>`  </xs:choice>`<br>`  <xs:attribute ref="bbl:handler" use="optional"/>`<br>`  <xs:attribute name="mode" use="required">`<br>`    <xs:simpleType>`<br>`      <xs:restriction base="xs:NCName">`<br>`        <xs:enumeration value="packetCount"/>`<br>`        <xs:enumeration value="bandwidth"/>`<br>`      </xs:restriction>`<br>`    </xs:simpleType>`<br>`  </xs:attribute>`<br>`  <xs:attribute name="weight" type="xs:int" use="optional" default="1"/>`<br>`</xs:complexType>` |

### 7.3.22 `Channel` Element

A `Channel` element is of type `ChannelType` type, and defines a logical transmission channel within a Multiplexer (see 7.3.21).

**Table 22 — ChannelType**

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Used by** | `Multiplexer` | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | `id` | xs:ID | required | A unique identifier which may be used by a Packet as a non-timed expression of the packet delivery. |
| | `weight` | xs:int | required | Defines the relative importance of this channel. The relative importance of a Channel is calculated by summarizing the weight of all **sibling** Multiplexer(s) and Channel(s) at the current level, and dividing this by the sum of the weight of the Channel(s) and Multiplexer(s) at the current level. |
| **Source** | `<xs:complexType name="ChannelType">`<br>`  <xs:attribute name="id" type="xs:ID" use="required"/>`<br>`  <xs:attribute name="weight" type="xs:int" use="optional" default="1"/>`<br>`</xs:complexType>` | | | |

### 7.3.23 `BSD` element

The `BSD` element is of type `BSDType` and is used to register that a namespace used in the BBL Document is a BSD Namespace. Elements with this namespace, whether part of the BBL Document or included by `Include` elements, will be converted to their binary representation by an ISO/IEC 21000-7 BSDToBin process, according to the BS Schema identified by `bsSchemaLocation`. Prefixes used in the BBL document (for example in W3C XPATH expressions) associated with BSD Namespaces shall be declared as per W3C XMLNAMES within the BBL document.

NOTE 1 This is the case for all prefix-namespace associations, as per W3C XMLNAMES.

NOTE 2 If the BBL document is to be encoded (for example by MPEG BiM) and in the decoding process namespace prefixes might be mangled, then this could result in the loss of prefix-namespace association for prefixes used in W3C XPATH expressions. To avoid this the W3C XPATH expressions can be authored without using namespace prefixes (for example using the namespace-uri() function).

Table 23 — BSDType

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | namespace | xs:anyURI | required | The namespace URI which is being declared as belonging to a BSD Namespace. |
| | bsSchemaLocation | xs:anyURI | required | A URI which corresponds to the location of the BS Schema associated with this BSD Namespace. |
| | fromBinarySource | xs:boolean | optional | If true or omitted, W3C XPATH expressions containing this namespace are to be resolved against the in-scope Binary Source Document. If false, W3C XPATH expressions are resolved against the in scope XML Source and subsequently converted to binary. |
| **Source** | `<xs:complexType name="BSDType">`<br>`  <xs:attribute name="namespace" type="xs:anyURI" use="required"/>`<br>`  <xs:attribute name="bsSchemaLocation" type="xs:anyURI" use="required"/>`<br>`  <xs:attribute name="fromBinarySource" type="xs:boolean" use="optional"`<br>`                default="true"/>`<br>`</xs:complexType>` | | | |

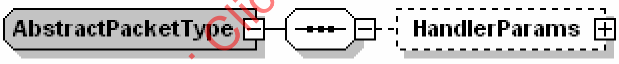
### 7.3.24 AbstractPacketType type

The AbstractPacketType type is a base type for elements concerned with Packets and Packet Streams such as the Packet element (see 7.3.26) and PacketStream element (see 7.3.27).

**Validation Rule:**

— The ID referenced by the handler attribute, if present, shall be attached to a Handler or a Channel element within the same Instance or Binding as this element or within a BBL Instance or Binding document to which the document containing this element is bound.

Table 24 — AbstractPacketType

| | | | | |
|---|---|---|---|---|
| **Diagram** |  | | | |
| **Children** | bbl:Content bbl:Variables | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | id | xs:ID | optional | See InstanceType (7.3.10). |
| | bbl:handler | xs:IDREF | optional | A reference to the Handler to be used to transmit the Packet. If this element is missing, the first Handler declared within register shall be used. |
| | deliveryCondition | xs:string | optional | A W3C XPATH expression which evaluates to a boolean. The associated Packet shall not be delivered until/unless this W3C XPATH expression evaluates to true. |

| | | | | |
|---|---|---|---|---|
| | deliveryTime | xs:string | optional | A W3C XPATH expression which is resolved against the in-scope Source document, and evaluates to a number indicating the first time that the Packet shall be delivered (in the in-scope time scheme, see 7.3.7) relative to the beginning of the BBL session.<br>If the attribute is not provided, then the Delivery Time is specified by non-normative means (for example, in a live streaming scenario, Packets may be output as soon as input data for them is available). |
| | repeat | xs:string | optional | A W3C XPATH expression which is resolved against the in-scope Source Document, and evaluates to the period of repetition (in the in-scope time scheme, see 7.3.7) for transmission of the Packet (Stream). A value of 0 indicates no repetition and is the default. |

**Source**

```
   <xs:complexType name="AbstractPacketType" abstract="true">
     <xs:complexContent>
       <xs:extension base="bbl:IdentifiableElementType">
         <xs:sequence>
           <xs:element name="HandlerParams" type="bbl:ParametricElementType"
minOccurs="0"/>
         </xs:sequence>
         <xs:attribute ref="bbl:handler" use="optional"/>
         <xs:attribute name="deliveryCondition" type="xs:string" use="optional"/>
         <xs:attribute name="deliveryTime" type="xs:string" use="optional"/>
         <xs:attribute name="repeat" type="xs:float" use="optional"
default="0.0"/>
       </xs:extension>
     </xs:complexContent>
   </xs:complexType>
```

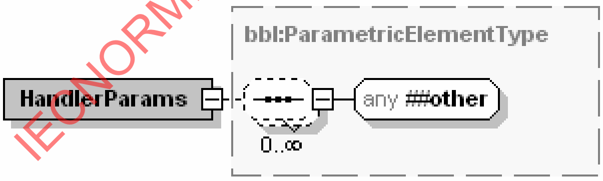
### 7.3.25 HandlerParams element

The HandlerParams element is of type ParametricElementType and contains parameters which are sent to the Handler associated with the parent Packet or PacketStream. The descendant content of HandlerParams is provided to the Handler at the same time as the parent Packet or PacketStream content.

The syntax and semantics of the descendant content of HandlerParams are determined by the Handler type of the handler being used by the parent Packet/PacketStream to map Packets to a delivery channel.

For an example of the use of this element, see C.5.

**Table 25 — HandlerParams element**

| | | | | |
|---|---|---|---|---|
| **Diagram** |  bbl:ParametricElementType<br>HandlerParams<br>any ##other<br>0..∞ | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | id | xs:ID | optional | See InstanceType (7.3.10) |
| **Source** | See ParametricElementType (7.3.18) | | | |

### 7.3.26 `Packet` element

A `Packet` element is of type PacketType and contains instructions for assembling a single Packet. It contains a `Content` element which describes how the Packet content is assembled, an optional `Variables` element, which allows variable values to be updated, and attributes to declare the Delivery Time, Delivery condition, Repeat Period, and Handler associated with the Packet.

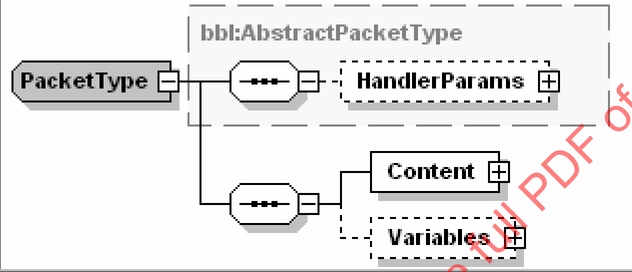
Along with the `PacketStream` element, `Packet` is the central structure used by BBL to declare streaming instructions.

For an example of the use of this element, see C.3.

**Validation Rule:**

— The ID referenced by the `handler` attribute, if present, shall be attached to a `Handler` or a `Channel` element within the same `Instance` or `Binding` as this element or within a BBL Instance or Binding document to which the document containing this element is bound.

**Table 26 — PacketType**

<table>
<tr><td><b>Diagram</b></td><td colspan="4">
</td></tr>
<tr><td><b>Children</b></td><td colspan="4">bbl:Content bbl:Variables</td></tr>
<tr><td rowspan="9"><b>Attributes</b></td><td><b>Name</b></td><td><b>Type</b></td><td><b>Use</b></td><td><b>Description</b></td></tr>
<tr><td>xmlSource</td><td>xs:string</td><td>optional</td><td rowspan="3">See SourceElementType (7.3.8)</td></tr>
<tr><td>binarySource</td><td>xs:string</td><td>optional</td></tr>
<tr><td>bSrcNamespace</td><td>xs:string</td><td>optional</td></tr>
<tr><td>id</td><td>xs:ID</td><td>optional</td><td>See InstanceType (7.3.10).</td></tr>
<tr><td>bbl:handler</td><td>xs:IDREF</td><td>optional</td><td rowspan="4">See AbstractPacketType (7.3.24)</td></tr>
<tr><td>deliveryCondition</td><td>xs:string</td><td>optional</td></tr>
<tr><td>deliveryTime</td><td>xs:string</td><td>optional</td></tr>
<tr><td>repeat</td><td>xs:string</td><td>optional</td></tr>
<tr><td><b>Source</b></td><td colspan="4"><pre>
<xs:complexType name="PacketType">
  <xs:complexContent>
    <xs:extension base="bbl:AbstractPacketType">
      <xs:sequence>
        <xs:element name="Content" type="bbl:AbstractContentType"/>
        <xs:element name="Variables" type="bbl:VariablesType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
</pre></td></tr>
</table>

### 7.3.27 `PacketStream` element

A `PacketStream` element is of type PacketStreamType and is used to declare a Stream of Packets. A `PacketStream` has the same attribute parameters and variables element as `Packet`. However, the content of a `PacketStream` is described using either a `ContentTemplate`, which provides instructions for assembling the content using static and dynamic elements, or a `bind` element, which instantiates a binding declared elsewhere using the in-scope Source Documents.

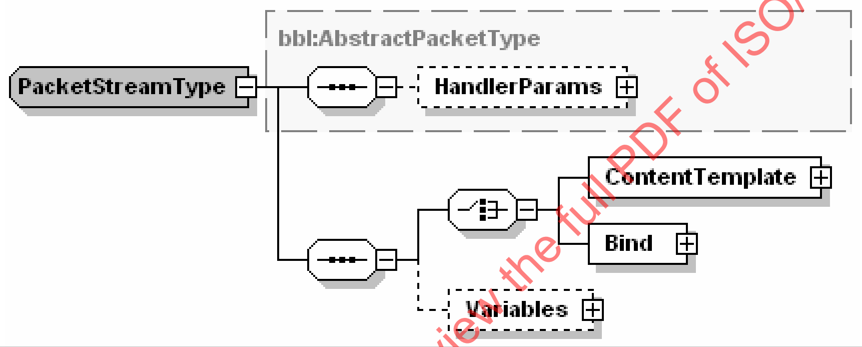
Along with the `Packet` element, `PacketStream` is the central structure used by BBL to declare streaming instructions.

For an example of the use of this element, see C.4.

**Validation Rule:**

— The ID referenced by the `handler` attribute, if present, shall be attached to a `Handler` or a `Channel` element within the same `Instance` or `Binding` as this element or within a BBL Instance or Binding document to which the document containing this element is bound.

**Table 27 — PacketStreamType**

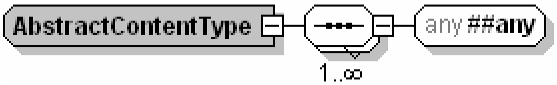
| | | | | |
|---|---|---|---|---|
| **Diagram** |   bbl:AbstractPacketType<br>PacketStreamType<br>HandlerParams<br>ContentTemplate<br>Bind<br>Variables | | | |
| **Children** | `bbl:ContentTemplate bbl:Bind bbl:Variables` | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | `xmlSource` | xs:string | optional | See `SourceElementType` (7.3.8) |
| | `binarySource` | xs:string | optional | |
| | `bSrcNamespace` | xs:string | optional | |
| | `id` | xs:ID | optional | See `InstanceType` (7.3.10) |
| | `bbl:handler` | xs:IDREF | optional | See `AbstractPacketType` (7.3.24) |
| | `deliveryCondition` | xs:string | optional | |
| | `repeat` | xs:string | optional | |
| | `deliveryTime` | xs:string | optional | See `AbstractPacketType` (7.3.24)<br>NOTE This value defines the Delivery Time of the first Packet in the stream, unless that Packet is matched by a `DeliveryTimes` element (see 7.3.34). The Delivery Time of subsequent Packets in the stream is defined by `DeliveryTimes` and/or `Durations` (see 7.3.34 and 7.3.35). |

<table>
<tr><td><b>Source</b></td><td><pre>
&lt;xs:complexType name="PacketStreamType"&gt;
  &lt;xs:complexContent&gt;
    &lt;xs:extension base="bbl:AbstractPacketType"&gt;
      &lt;xs:sequence&gt;
        &lt;xs:choice&gt;
          &lt;xs:element name="ContentTemplate" type="bbl:AbstractContentType"/&gt;
          &lt;xs:element name="Bind" type="bbl:BindType"/&gt;
        &lt;/xs:choice&gt;
        &lt;xs:element name="Variables" type="bbl:VariablesType" minOccurs="0"/&gt;
      &lt;/xs:sequence&gt;
    &lt;/xs:extension&gt;
  &lt;/xs:complexContent&gt;
&lt;/xs:complexType&gt;
</pre></td></tr>
</table>

### 7.3.28 `AbstractContentType` type

The `AbstractContentType` type is a base type for elements concerned with assembling content of Packets and Packet Streams such as the `Content` element (see 7.3.29) and the `ContentTemplate` element (see 7.3.30).

**Table 28 — AbstractContentType**

<table>
<tr><td><b>Diagram</b></td><td colspan="4">
</td></tr>
<tr><td rowspan="5"><b>Attributes</b></td><td><b>Name</b></td><td><b>Type</b></td><td><b>Use</b></td><td><b>Description</b></td></tr>
<tr><td>xmlSource</td><td>xs:string</td><td>optional</td><td rowspan="3">See SourceElementType (7.3.8)</td></tr>
<tr><td>binarySource</td><td>xs:string</td><td>optional</td></tr>
<tr><td>bSrcNamespace</td><td>xs:string</td><td>optional</td></tr>
<tr><td>id</td><td>xs:ID</td><td>optional</td><td>See InstanceType (7.3.10)</td></tr>
<tr><td><b>Source</b></td><td colspan="4"><pre>
&lt;xs:complexType name="AbstractContentType"&gt;
  &lt;xs:complexContent&gt;
    &lt;xs:extension base="bbl:IdentifiableElementType"&gt;
      &lt;xs:sequence maxOccurs="unbounded"&gt;
        &lt;xs:any namespace="##any" processContents="skip"/&gt;
      &lt;/xs:sequence&gt;
    &lt;/xs:extension&gt;
  &lt;/xs:complexContent&gt;
&lt;/xs:complexType&gt;
</pre></td></tr>
</table>

### 7.3.29 `Content` element

The `Content` element is of type `AbstractContentType` and contains instructions for assembling the content of a Packet. There are five ways that this may be conducted:

a) Elements from namespaces other than that of BBL may be directly instantiated as descendants of the Content element. This includes elements from namespaces which have been declared as BSD.

b) Elements which are instantiated in other XML Documents may be retrieved by an Include element and included in the content, either as a child of the Content or as a child of any of its descendants.

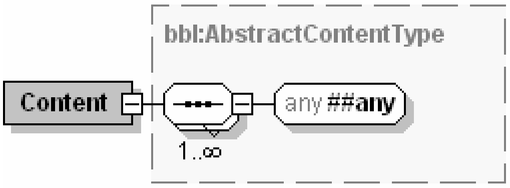
c) Structures from Binary Documents which are identified within the BS Schema identified with a BSD Namespace may be retrieved by an Include element and included in the content, either as a child of Content or as a child of any of its descendants.

d) Attributes may be attached to any element within the content by the use of an Attribute element with a Value-of child. Where attributes are to be attached to elements which are to be retrieved by any Include element, the match attribute may be used to indicate the elements to which the attribute is to be attached.

e) Element content may be added to any element within the content by the use of a Value-of element.

For an example of the use of this element, see C.3.

**Table 29 — Content element**

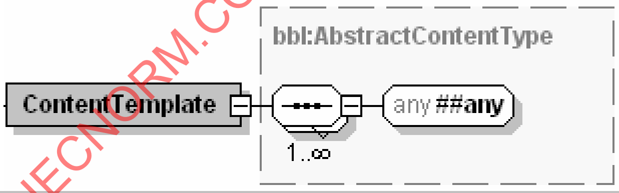
| | | | | |
|---|---|---|---|---|
| **Diagram** |  | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | id | xs:ID | optional | See InstanceType (7.3.10) |
| **Source** | See AbstractContentType (7.3.28) | | | |

### 7.3.30 ContentTemplate element

The ContentTemplate element is of type AbstractContentType and contains instructions for assembling the content of a Stream of Packets. This shall be conducted in the same manner as a Content element, with the exception that Include descendants may have additional semantics to fragment their returned content (see 7.3.31).

For an example of the use of this element, see C.4.

**Table 30 — ContentTemplate element**

| | | | | |
|---|---|---|---|---|
| **Diagram** |  bbl:AbstractContentType<br>ContentTemplate<br>any ##any<br>1..∞ | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | id | xs:ID | optional | See InstanceType (7.3.10) |
| **Source** | See AbstractContentType (7.3.28) | | | |

#### 7.3.31 `Include` element

The `Include` element is of type `IncludeType` and is used to retrieve fragments of external Documents. Its semantics are similar to the XInclude element, however the BBL `Include` has additional functionality tailored to BBL.

The `Include` element contains a `ref` attribute whose value is a W3C XPATH expression which resolves to a node-set that represents the root elements of the fragments to be included. The optional `depth` attribute subsequently selects a number of levels of descendants of the node-set to be included. The nodes within the node-set are guaranteed to be in the original document order.

`Include` elements may be nested – in which case the fragments returned by the inner `Include` shall be attached to those of the outer `Include` as specified by the `match` attribute, if present, or in their original relative location within the Source Document.

`Include` elements may be used to retrieve fragments of Binary Documents. This is done by referencing elements in a namespace defined as being to a BSD namespace (see 7.3.23). In such a case the context of the W3C XPATH expression is *the BS Description which would be created by applying the referenced BS Schema to the in-scope Binary Source Document*. The BS Description need not be actually instantiated, its contents may be inferred from the Binary Source Document directly, using the BS Schema.

As well as child `Include` elements, an `Include` element may have zero or more `Attribute` element children. These contain a `match` attribute to attach attribute(s) to one or more nodes returned by the `Include` element. If the `Include` element is within a `ContentTemplate`, it may also have `Timing` and `Fragmentation` children.

For a description of the fragmentation process, see 7.5.2.

For examples of the use of this element, see C.2, C.3, C.4 and 7.3.6.

**Validation Rules:**

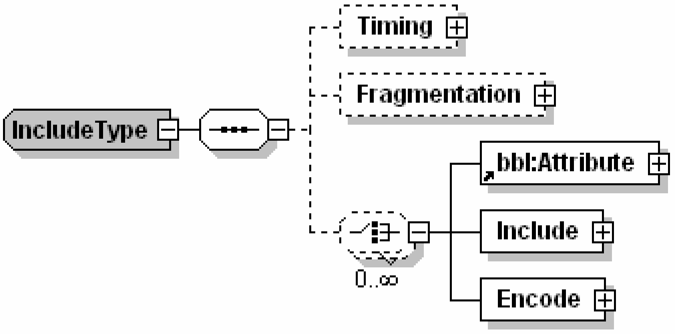
— Because `Include` elements are always descendants of a `Content` or `ContentTemplate` element (which have processContents set to "skip"), an `Include` element will never be validated directly by an XML Schema Validator. However, `Include` elements shall still be schema-valid.

— An `Include` shall declare a `match` attribute only if it is the child of another `Include` element.

— An `Include` element that is the child of another `Include` element shall not declare an `xmlSource` attribute unless it also declares a `match` attribute. (The insertion point for the nested `Include` element would become undefined).

— An `Include` element shall declare `Timing` and `Fragmentation` children only if it is the descendant of a `ContentTemplate` element.

Table 31 — IncludeType

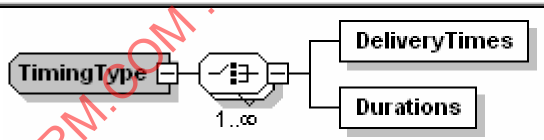
| | | | | |
|---|---|---|---|---|
| **Diagram** |  Timing<br>Fragmentation<br>IncludeType<br>bbl:Attribute<br>Include<br>Encode<br>0..∞ | | | |
| **Children** | bbl:Timing bbl:Fragmentation bbl:Attribute bbl:Include | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | ref | xs:string | required | A W3C XPATH expression which resolves to the node-set which represents the root elements of the fragment(s) to be included. |
| | depth | xs:integer | optional | The number of levels of descendants of the nodes returned by ref which are to be included. The default value is 0. A value of -1 indicates that all descendants are to be included. |
| | height | xs:integer | optional | The number of levels of direct ancestors of the nodes returned by ref which are to be included. The default value is 0. A value of -1 indicates that all direct ancestors are to be included. |
| | match | xs:string | optional | A W3C XPATH expression which is evaluated against the nodes returned by the parent Include element and resolves to the node(s) to which the included content is to be attached.<br>If this attribute is not provided, the included content is attached to the parent content as it was in the Source Document. |

| | |
|---|---|
| **Source** | `<xs:complexType name="IncludeType">`<br>`<xs:complexContent>`<br>`<xs:extension base="bbl:SourceElementType">`<br>`<xs:sequence>`<br>`<xs:element name="Timing" type="bbl:TimingType" minOccurs="0"/>`<br>`<xs:element name="Fragmentation" type="bbl:FragmentationType"`<br>`minOccurs="0"/>`<br>`<xs:choice minOccurs="0" maxOccurs="unbounded">`<br>`<xs:element ref="bbl:Attribute"/>`<br>`<xs:element name="Include" type="bbl:IncludeType"/>`<br>`<xs:element name="Encode" type="bbl:EncodeType"/>`<br>`</xs:choice>`<br>`</xs:sequence>`<br>`<xs:attribute name="ref" type="xs:string" use="required"/>`<br>`<xs:attribute name="depth" use="optional" default="0">`<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:integer">`<br>`<xs:minInclusive value="-1"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>`<br>`</xs:attribute>`<br>`<xs:attribute name="height" use="optional" default="0">`<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:integer">`<br>`<xs:minInclusive value="-1"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>`<br>`</xs:attribute>`<br>`<xs:attribute ref="bbl:match" use="optional"/>`<br>`</xs:extension>`<br>`</xs:complexContent>`<br>`</xs:complexType>` |

### 7.3.32 `Timing` element

A `Timing` element is of type TimingType and contains `DeliveryTimes` and `Durations` declarations which are used to attach Delivery Times and Durations (respectively) to fragments returned by the parent `Include` element as part of a `ContentTemplate`.

**Table 32 — TimingType**

| | |
|---|---|
| **Diagram** |  |
| **Children** | `bbl:DeliveryTimes bbl:Durations` |
| **Source** | `<xs:complexType name="TimingType">`<br>`<xs:choice maxOccurs="unbounded">`<br>`<xs:element name="DeliveryTimes" type="bbl:TimingElementType">`<br>`</xs:element>`<br>`<xs:element name="Durations" type="bbl:TimingElementType">`<br>`</xs:element>`<br>`</xs:choice>`<br>`</xs:complexType>` |

### 7.3.33 `TimingElementType` type

The `TimingElementType` type is a base type for elements concerned with timing information such as the `DeliveryTimes` element (see 7.3.34) and the `Durations` element (see 7.3.35).

**Table 33 — TimingElementType**

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | match | xs:string | optional | A W3C XPATH expression which returns the node-set of elements which are to be operated on by this element. The context of the expression is a Document containing the node-set returned by the parent Include element. If the attribute is not present, all elements returned by the parent Include element are matched. |
| | value | xs:string | optional | A W3C XPATH expression to be evaluated against each node returned by the match expression, which resolves to the timing value. The context of the expression is each node that it is evaluated against. |
| **Source** | <xs:complexType name="TimingElementType"><br>  <xs:attribute ref="bbl:match" use="optional"/><br>  <xs:attribute ref="bbl:value" use="required"/><br></xs:complexType><br><br><xs:attribute name="match" type="xs:string" default="."/><br><br><xs:attribute name="value" type="xs:string"/> | | | |

### 7.3.34 DeliveryTimes element

A DeliveryTimes element is of type TimingElementType and is used to attach a Delivery Time to Packets containing elements which match a certain W3C XPATH expression.

For an example of the use of this element, see C.4.

**Table 34 — DeliveryTimes element**

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | match | xs:string | optional | See TimingElementType (7.3.33) |
| | value | xs:string | optional | See TimingElementType (7.3.33)<br>The timing value to which the W3C XPATH expression resolves is the value (in the in-scope time scheme, see 7.3.7, since the beginning of the PacketStream) of the Delivery Time of that node. |
| **Source** | See TimingElementType (7.3.33) | | | |

### 7.3.35 Durations element

A Durations element is of type TimingElementType and is used to attach Durations to elements returned by an Include element which matches a certain W3C XPATH expression.

Table 35 — Durations element

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | match | xs:string | optional | See TimingElementType (7.3.33) |
| | value | xs:string | optional | See TimingElementType (7.3.33)<br>The timing value to which the W3C XPATH expression resolves is the value (in the in-scope time scheme, see 7.3.7) of the Duration of that node. |
| **Source** | See TimingElementType (7.3.33) | | | |

### 7.3.36 Fragmentation element

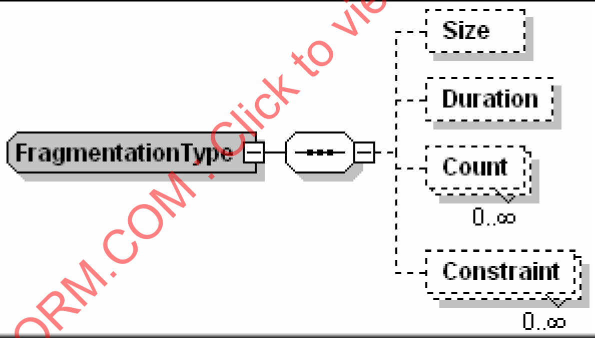
The Fragmentation element is of type FragmentationType and contains children which specify the rules for fragmenting the content returned by the parent Include element.

A Fragmentation element may contain zero children however such a case has no behaviour.

The algorithm for the fragmentation process is detailed in 7.5.2.

For an example of the use of this element, see C.4.

Table 36 — FragmentationType

| | |
|---|---|
| **Diagram** |  |
| **Children** | bbl:Size bbl:Duration bbl:Count bbl:Constraint |
| **Source** | `  <xs:complexType name="FragmentationType">`<br>`    <xs:sequence>`<br>`      <xs:element name="Size" type="bbl:SizeType" minOccurs="0"/>`<br>`      <xs:element name="Duration" type="bbl:DurationType" minOccurs="0"/>`<br>`      <xs:element name="Count" type="bbl:CountType" minOccurs="0"`<br>`maxOccurs="unbounded"/>`<br>`      <xs:element name="Constraint" type="bbl:ConstraintType" minOccurs="0"`<br>`                  maxOccurs="unbounded"/>`<br>`    </xs:sequence>`<br>`  </xs:complexType>` |

### 7.3.37 `Size` element

A `Size` element is of type `SizeType` and declares that content should be fragmented so that the size of the output Packet shall be no greater than the given value.

For an example of the use of this element, see C.4.

**Table 37 — SizeType**

| | | | | |
|---|---|---|---|---|
| **Diagram** | SizeType | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | `value` | xs:positiveInteger | required | The maximum size (in bytes) of the included content in any one Packet.. |
| **Source** | `<xs:complexType name="SizeType">`<br>`  <xs:attribute name="value" type="xs:positiveInteger" use="required"/>`<br>`</xs:complexType>` | | | |

### 7.3.38 `Duration` element

A `Duration` element is of type `DurationType` and declares that content should be fragmented so that the total Duration of elements in any one Packet is no greater than the given value.

Any element with no Duration attached to it (via the `Durations` element, 7.3.35) has a Duration of zero.

**Table 38 — DurationType**

| | | | | |
|---|---|---|---|---|
| **Diagram** | DurationType | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | `value` | xs:string | required | The maximum Duration (in the in-scope time scheme, see 7.3.7) of the included content in any one Packet. |
| **Source** | `<xs:complexType name="DurationType">`<br>`  <xs:attribute name="value" type="xs:string" use="required"/>`<br>`</xs:complexType>` | | | |

### 7.3.39 `Count` element

A `Count` element is of type `CountType` and declares that content should be fragmented so that the number of nodes matched by the `match` expression is no greater than the given value.

For an example of the use of this element, see C.4.

Table 39 — CountType

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | value | xs:positiveInteger | required | The maximum number of elements matching the match W3C XPATH expression in the included content in any one Packet. |
| | bbl:match | xs:string | required | See TimingElementType (7.3.33) |
| **Source** | `<xs:complexType name="CountType">`<br>`  <xs:attribute name="value" type="xs:positiveInteger" use="required"/>`<br>`  <xs:attribute ref="bbl:match" use="optional"/>`<br>`</xs:complexType>` | | | |

### 7.3.40 Constraint element

A Constraint element is of type ConstraintType and applies a Fragmentation Constraint to the Fragmentation of nodes which are matched by the match W3C XPATH expression. The possible constraints are given in Table 40.

Table 40 — Constraint type values

| Constraint type | Semantics |
|---|---|
| first | All matched nodes must be the first content included in any Packet. The Packet is consequently completed immediately before an element matched with this constraint. |
| last | All matched nodes must be the last content included in any Packet. The Packet is consequently completed immediately after an element matched with this constraint. |
| unbroken | All matched nodes must have their entire included subtree included within a single Packet. |

For an example of the use of this element, see C.4.

Table 41 — ConstraintType

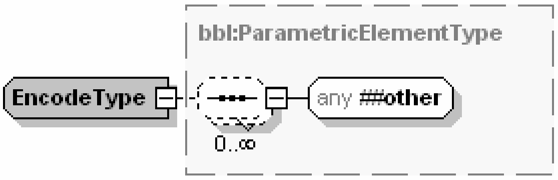
| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | type | enumeration | required | The type of constraint to be applied to elements matched by the bbl:match attribute. |
| | bbl:match | xs:string | required | See TimingElementType (7.3.33) |
| **Source** | `<xs:complexType name="ConstraintType">`<br>`  <xs:attribute name="type" use="required">`<br>`    <xs:simpleType>`<br>`      <xs:restriction base="xs:NCName">`<br>`        <xs:enumeration value="first"/>`<br>`        <xs:enumeration value="last"/>`<br>`        <xs:enumeration value="unbroken"/>`<br>`      </xs:restriction>`<br>`    </xs:simpleType>`<br>`  </xs:attribute>`<br>`  <xs:attribute ref="bbl:match" use="optional"/>`<br>`</xs:complexType>` | | | |

### 7.3.41 `Encode` element

The `Encode` element is of type `EncodeType` and is used to apply a registered Encoder to selected content returned by an `Include` element. The content is selected by use of the `match` attribute.

Any XML and/or Binary Source documents declared at this node shall apply only to any XPath expressions within child elements, and NOT to the XPath expression used by `match`.

**Table 42 — EncodeType**

<table>
<tr><td><b>Diagram</b></td><td colspan="4">
</td></tr>
<tr><td><b>Used by</b></td><td colspan="4"><code>Include</code></td></tr>
<tr><td><b>Children</b></td><td colspan="4">The children of this element are provided as input parameters to the specified Encoder. Permissible elements are specified by individual Encoders.</td></tr>
<tr><td rowspan="7"><b>Attributes</b></td><td><b>Name</b></td><td><b>Type</b></td><td><b>Use</b></td><td><b>Description</b></td></tr>
<tr><td><code>xmlSource</code></td><td>xs:string</td><td>optional</td><td rowspan="3">See <code>SourceElementType</code> (7.3.8)</td></tr>
<tr><td><code>binarySource</code></td><td>xs:string</td><td>optional</td></tr>
<tr><td><code>bSrcNamespace</code></td><td>xs:string</td><td>optional</td></tr>
<tr><td><code>id</code></td><td>xs:ID</td><td>optional</td><td>See <code>InstanceType</code> (7.3.10)</td></tr>
<tr><td><code>bbl:encoding</code></td><td>xs:IDREF</td><td>required</td><td>A reference to the Encoder to be used to transmit the Packet. If this element is missing the first Handler declared within register shall be used.</td></tr>
<tr><td><code>bbl:match</code></td><td>xs:string</td><td>optional</td><td>An XPath expression which resolves to the node(s) to which the encoded content is to be attached. The context node for the XPath expression is a Document Fragment containing the node-set returned by the parent <code>Include</code> element.</td></tr>
<tr><td><b>Source</b></td><td colspan="4"><pre>
&lt;xs:complexType name="EncodeType"&gt;
  &lt;xs:complexContent&gt;
    &lt;xs:extension base="bbl:ParametricElementType"&gt;
      &lt;xs:attribute ref="bbl:encoding" use="required"/&gt;
      &lt;xs:attribute ref="bbl:match" use="optional"/&gt;
    &lt;/xs:extension&gt;
  &lt;/xs:complexContent&gt;
&lt;/xs:complexType&gt;
&lt;xs:attribute name="encoding" type="xs:IDREF"/&gt;
</pre></td></tr>
</table>

### 7.3.42 `Bind` element

A `Bind` element is of type `BindType` and is used to instantiate a `Binding`.

The `Binding` to be used may either be specified by the `binding` attribute, attached directly as a child element, or attached via an XInclude statement.

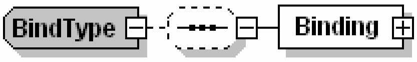
The context of all timing and W3C XPATH expressions within the associated `Binding` is relative to that in-scope at the `Bind` element.

EXAMPLE If the `PacketStream` in which the `Bind` element is contained has `deliveryTime` equals "10", and the referenced Binding contains a `Packet` with `deliveryTime` equals "5", then that Packet will be delivered at time equals 15 (in the in-scope time scheme). See 7.3.6 for an example regarding the context of W3C XPATH expressions within a `Binding`.

**Validation Rule:**

— If the `binding` attribute is present, the `Binding` child element shall be absent, and vice versa.

**Table 43 — BindType**

| | | | | |
|---|---|---|---|---|
| **Diagram** |  | | | |
| **Children** | bbl:Binding | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | binding | xs:string | optional | EITHER:<br>A URI which locates the Binding to be attached at this point,<br>OR<br>A W3C XPATH expression, delimited by "?" (see also 7.3.4), which resolves to a URI locating the Binding to be attached at this point. |
| **Source** | `<xs:complexType name="BindType">`<br>`  <xs:complexContent>`<br>`    <xs:extension base="bbl:SourceElementType">`<br>`      <xs:sequence minOccurs="0">`<br>`        <xs:element name="Binding" type="bbl:BindingType"/>`<br>`      </xs:sequence>`<br>`      <xs:attribute name="binding" type="xs:string" use="optional"/>`<br>`    </xs:extension>`<br>`  </xs:complexContent>`<br>`</xs:complexType>` | | | |

### 7.3.43 `value-of` element

The `value-of` element has the same semantics as the XSLT [2] element of the same name. It is used to populate the content of elements and attributes declared as part of a `Content` or `ContentTemplate` element, with values retrieved from an external source. While `Include` provides a powerful facility for assembling XML subtrees, `value-of` is intended to return values of simple type.

The text content of the containing element or value of the containing attribute is set to the resolved value of the `select` W3C XPATH expression.

**Table 44 — value-of element**

| | | | | |
|---|---|---|---|---|
| **Diagram** | | | | |
| **Used by** | Attribute, elements of non-BBL namespace within Content or ContentTemplate | | | |
| **Attributes** | **Name** | **Type** | **Use** | **Description** |
| | xmlSource | xs:string | optional | See SourceElementType (7.3.8) |
| | binarySource | xs:string | optional | |
| | bSrcNamespace | xs:string | optional | |
| | select | xs:string | required | A W3C XPATH expression which evaluates to a simple type. The context of the expression is always the in-scope XML Source Document.. |

| | |
|---|---|
| **Source** | <xs:element name="value-of"><br>  <xs:complexType><br>    <xs:complexContent><br>      <xs:extension base="bbl:SourceElementType"><br>        <xs:attribute name="select" type="xs:string" use="required"/><br>      </xs:extension><br>    </xs:complexContent><br>  </xs:complexType><br></xs:element> |

### 7.3.44 `Attribute` element

An `Attribute` element is used to attach an attribute to an element which is either declared as a descendant of a `Content` or `ContentTemplate` element, or inferred by an `Include` element.

In the former case, the attribute element is declared as a child of the element to which the attribute is to be attached.

EXAMPLE 1

```
<Content>
  <did:DIDL>
    <Attribute bbl:name="xmlns:bbl">
      <value-of select="bbl:BBL/namespace-uri()" xmlSource="#"/>
    </Attribute>
    <!-- ... -->
  </did:DIDL>
</Content>
```

In the latter case (where the attribute is to be attached to an element inferred by an `Include`), the attribute element is declared as a child of the `Include` element, and provided with a `match` attribute which returns the node(s) to which the attribute is to be attached.

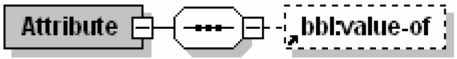
EXAMPLE 2

```
<Content>
  <Include ref="/did:DIDL" depth="2">
    <Attribute bbl:name="xmlns:bbl" match=".">
      <value-of select="bbl:BBL/namespace-uri()" xmlSource="#"/>
    </Attribute>
    <!-- ... -->
  </Include>
</Content>
```

**Validation Rule:**

— An `Attribute` element shall have a `match` attribute only if the `Attribute` element is the child of an `Include` element.

**Table 45 — Attribute element**

<table>
<tr><td><b>Diagram</b></td><td colspan="4">
</td></tr>
<tr><td><b>Children</b></td><td colspan="4">bbl:value-of</td></tr>
<tr><td><b>Used by</b></td><td colspan="4">Include</td></tr>
<tr><td rowspan="7"><b>Attributes</b></td><td><b>Name</b></td><td><b>Type</b></td><td><b>Use</b></td><td><b>Description</b></td></tr>
<tr><td>xmlSource</td><td>xs:string</td><td>optional</td><td rowspan="3">See SourceElementType (7.3.8)</td></tr>
<tr><td>binarySource</td><td>xs:string</td><td>optional</td></tr>
<tr><td>bSrcNamespace</td><td>xs:string</td><td>optional</td></tr>
<tr><td>id</td><td>xs:ID</td><td>optional</td><td>See Instance (7.3.10)</td></tr>
<tr><td>bbl:name</td><td>xs:QName</td><td>required</td><td>The name of the attribute.</td></tr>
<tr><td>bbl:match</td><td>xs:string</td><td>optional</td><td>See Include (7.3.31)</td></tr>
<tr><td><b>Source</b></td><td colspan="4"><pre>
&lt;xs:element name="Attribute"&gt;
  &lt;xs:complexType&gt;
    &lt;xs:complexContent&gt;
      &lt;xs:extension base="bbl:IdentifiableElementType"&gt;
        &lt;xs:sequence&gt;
          &lt;xs:element ref="bbl:value-of" minOccurs="0"/&gt;
        &lt;/xs:sequence&gt;
        &lt;xs:attribute ref="bbl:name" use="required"/&gt;
        &lt;xs:attribute ref="bbl:match" use="optional"/&gt;
      &lt;/xs:extension&gt;
    &lt;/xs:complexContent&gt;
  &lt;/xs:complexType&gt;
&lt;/xs:element&gt;
</pre></td></tr>
</table>

## 7.4 BBL provided attributes

**Table 46 — BBL provided attributes**

| Attribute name | Semantics |
|---|---|
| bbl:context | The value of this attribute shall be a W3C XPATH expression which shall resolve to the location at which the parent element (and its subtree) should be attached to the re-constructed DI. |
| bbl:rap | This attribute shall have a boolean value indicating whether the containing Packet may be considered to be a Random Access Point. This is to be used when the underlying delivery channel does not have a facility for signalling such a condition. If a bbl:rap attribute is not attached to a Packet, and the underlying delivery channel does not indicate it, the Packet is assumed to be not a Random Access Point. |
| bbl:seqNumber | The value of this attribute shall be an integer indicating the sequence number of the containing Packet. It is to be used when the underlying delivery channel does not provide such a value, and is necessary for computing the Processability of Packets in a stream. |
| bbl:processable | This attribute shall have a boolean value indicating whether the containing Packet is Processable. If the value is false, then the Packet shall not be processed until a subsequent Packet (as indicated by bbl:seqNumber or the underlying delivery channel) is received with a bbl:processable attribute set to true. A Packet without a bbl:processable attribute is assumed to have such an attribute with a value of true. |

The attributes listed in Table 46 — BBL provided attributes are normatively specified by BBL, so that an MPEG-21 receiving Peer may reassemble a DID and infer other properties of the Packet.

## 7.5 BBL processing model

### 7.5.1 Processing order

Figure 6 shows the order in which `Packet` or `PacketStream` elements shall be processed. For each `Packet` or `PacketStream`

a) `Include` elements are resolved (and possibly fragmented, see 7.3.31),

b) variables in `Define` and `Assign` elements are processed (see 7.3.14 and 7.3.15),

c) `value-of` and `Timing` elements are resolved (see 7.3.43 and 7.3.32 to 7.3.35),

d) BSD content is parsed,

e) content may optionally be encoded,

f) the resulting Packet content is passed to the Handler which outputs the content to the delivery channel (see 7.3.19).

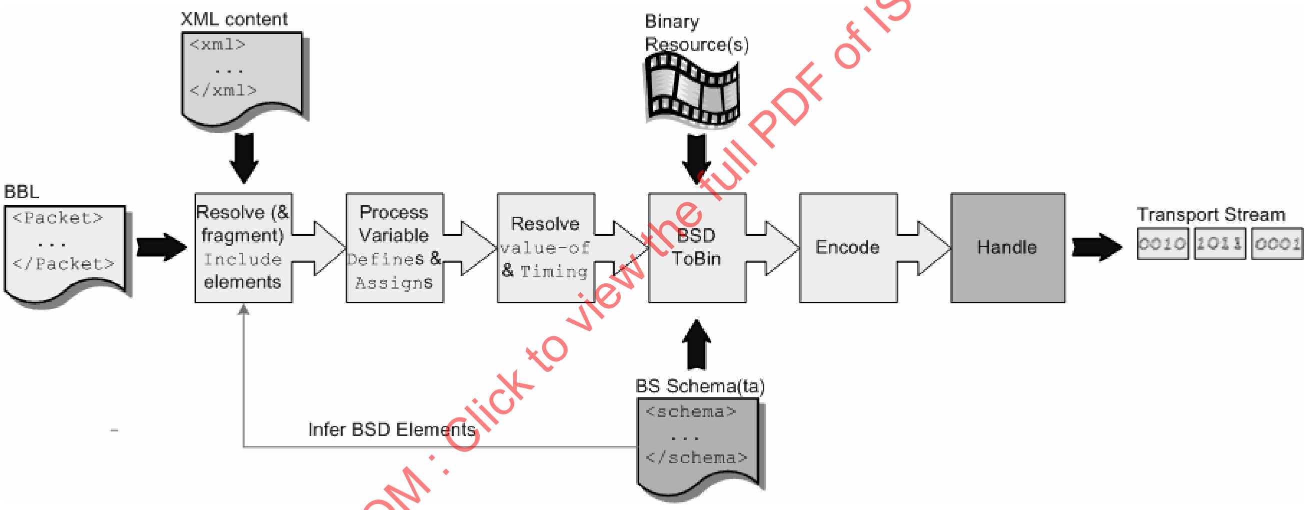


**Figure 6 — BBL Packet processing order**

NOTE It is important to note that `value-of` and `Timing` are resolved after the variables have been processed, so that these elements can utilise variable values from the current Packet.

### 7.5.2 Fragmentation process

In processing `Include` (7.3.31) elements within a `ContentTemplate` (7.3.30), the following algorithm is applied:

a) For each element returned by the Include (taking into account both the W3C XPATH expression and depth and/or height attributes, if present):

   1) Determine the Duration and/or Delivery Time of the element from `Timing` (7.3.32) parameters.

   2) If the element matches a "`first`" constraint (7.3.40), begin a new Packet before adding the element.

   3) If the element matches a "`last`" constraint (7.3.40), begin a new Packet after the element and its descendants have been processed.

4) If the element matches an "unbroken" constraint (7.3.40), apply the Fragmentation Rules (specified by a Fragmentation element, 7.3.36) to the subtree of the element, to the height and depth indicated. If the rules are met, insert the subtree into the current Packet, else insert it into the following Packet.

5) Else, calculate whether the content from this Include element within the current Packet will meet the Fragmentation Rules if the current element is added. If so, add the element, otherwise begin a new Packet and add the element to it.

NOTE The algorithm allows for cases where all Fragmentation Rules might not be able to be satisfied.

EXAMPLE If an element is constrained to be unbroken, but the size of the element is greater than the maximum Packet size, then the element is included by itself in a separate Packet.

# 8 Definition of Handlers for the Bitstream Binding Language

## 8.1 Introduction

This clause defines the syntax and semantics of the Handlers defined by this part of ISO/IEC 21000 which are provided for use by the Bitstream Binding Language.

A Handler is identified by its type URI which should be registered with the Registration Authority according to the procedure specified in Annex D. Further Handlers may be defined by other parties and their identifiers registered with the Registration Authority.

In addition to its type URI a Handler definition includes specification of the parameters available for the Handler. Parameters can be global for a given Handler (as child elements of the Handler element, see 7.3.19) or on a per Packet/Packet Stream basis (as child elements of a HandlerParams element contained in a Packet/PacketStream element, see 7.3.25, 7.3.26, and 7.3.27). The parameters will be specific to the underlying delivery layer of the Handler. Such parameters should include parameters related to robustness if these are available in the underlying delivery layer.

## 8.2 Metadata over MPEG-2 Transport Stream Handler

### 8.2.1 Introduction

The Metadata over MPEG-2 Transport Stream Handler shall conform to the carriage of metadata extensions as specified in ISO/IEC 13818-1.

The output of the Handler is a Transport Stream which can be multiplexed with other Transport Streams by an external system.

### 8.2.2 Handler type declaration

The MPEG-2 Transport Stream Handler shall be instantiated by registering a Handler with a type value of

```
urn:mpeg:mpeg21:2007:01-BBL-NS:handler:MPEG2TS
```

See 7.3.19 for more information.

### 8.2.3 Handler parameter syntax

The following W3C XMLSCHEMA fragment specifies the syntax for parameters which may be instantiated as child elements of a Handler with type equal to that specified in 8.2.2.

As specified in 7.3.4 all parameter values may be delimited by W3C XPATH expressions. Each W3C XPATH expression shall resolve to a value conforming to the element type.

This clause defines parameters for MPEG-2 syntactical elements only when they can be specified as part of a Handler declaration in a BBL document. All other MPEG-2 fields shall be populated by the MPEG-2 Transport Stream Handler according to ISO/IEC 13818-1.

```
<xs:simpleType name="hexByte">
  <xs:restriction base="xs:hexBinary">
    <xs:length value="1"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="Mode">
  <xs:simpleType>
    <xs:restriction base="Hmpg2:hexByte">
      <xs:enumeration value="15"/>
      <xs:enumeration value="16"/>
      <xs:enumeration value="17"/>
      <xs:enumeration value="18"/>
      <xs:enumeration value="19"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="MetadataServiceID" type="Hmpg2:hexByte"/>

<xs:element name="MetadataFormat" type="Hmpg2:hexByte"/>

<xs:element name="MetadataApplicationFormat">
  <xs:simpleType>
    <xs:restriction base="xs:hexBinary">
      <xs:length value="2"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:simpleType name="uint22">
  <xs:restriction base="xs:unsignedInt">
    <xs:maxExclusive value="4194304"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="MetadataInputLeakRate" type="Hmpg2:uint22"/>

<xs:element name="MetadataBufferSize" type="Hmpg2:uint22"/>

<xs:element name="MetadataOutputLeakRate" type="Hmpg2:uint22"/>

<xs:element name="ContentReferenceID" type="xs:base64Binary"/>

<xs:element name="DecoderConfig" type="xs:base64Binary"/>

<xs:element name="ContentTimeBaseIndicator">
  <xs:simpleType>
    <xs:restriction base="xs:unsignedByte">
      <xs:maxExclusive value="16"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="PrivateData" type="xs:base64Binary"/>
```

### 8.2.4 Handler parameter semantics

Table 47 specifies the semantics for the parameters of the MPEG-2 Transport Stream Handler.

**Table 47 — MPEG-2 Transport Stream Handler parameter element semantics**

| Element name | Use | Semantics |
|---|---|---|
| `Hmpg2:Mode` | required | Specifies the metadata delivery mechanism used according to ISO/IEC 13818-1:— 1), Table 2.29. Permissible values are 0x15 to 0x19. |
| `Hmpg2:MetadataServiceID` | required | Specifies the metadata service id as defined in ISO/IEC 13818-1:— 1), subclause 2.6.59. |
| `Hmpg2:MetadataFormat` | required | Indicates the format and coding of the metadata as specified by ISO/IEC 13818-1:— 1), Table 2-67. |
| `Hmpg2:MetadataApplicationFormat` | optional | Specifies the application responsible for defining usage, syntax and semantics of the Metadata Pointer and Content Labelling descriptors, as per ISO/IEC 13818-1:— 1), subclause 2.6.57. |
| `Hmpg2:MetadataInputLeakRate` | required | Specifies the input leak rate for the associated metadata as defined in ISO/IEC 13818-1:— 1), subclause 2.6.63. |
| `Hmpg2:MetadataBufferSize` | required | Specifies the size of buffer in the STD model for the associated metadata stream as defined in ISO/IEC 13818-1:— 1), subclause 2.6.63. |
| `Hmpg2:MetadataOutputLeakRate` | optional | Specifies the output leak rate for the associated metadata as defined in ISO/IEC 13818-1:— 1), subclause 2.6.63.<br>If `mode` is set to 0x15 or 0x19, this field shall not be present, otherwise it is required. |
| `Hmpg2:ContentReferenceID` | optional | Specifies the content_reference_id_record as defined in ISO/IEC 13818-1:— 1), subclause 2.6.56.<br>If this element is present, content_reference_id_record_flag shall be set to 1, content_reference_id_record_length shall be set to a value equal to the number of bytes contained in the element value, and the element value shall be stored in the required number of content_reference_id_bytes.<br>If this element is absent, content_reference_id_record_flag shall be set to 0. |
| `Hmpg2:DecoderConfig` | optional | Specifies the decoder configuration information as defined in ISO/IEC 13818-1:— 1), subclause 2.6.61.<br>The Handler shall decide the appropriate delivery mechanism for the value of this element, and shall set the decoder_config_bytes, the dec_config_identification_record_bytes and/or the decoder_config_metadata_service_id fields accordingly. |
| `Hmpg2:ContentTimeBaseIndicator` | required | Specifies the value of content_time_base_indicator as defined in ISO/IEC 13818-1:— 1), subclause 2.6.57. |
| `Hmpg2:PrivateData` | optional | Specifies private information associated with a metadata service as defined in ISO/IEC 13818-1:— 1), subclause 2.6.56.<br>If present, the value of this element shall be inserted as private_data_bytes. |

1) To be published.

# Annex A
(informative)

# XML Schema for Bitstream Binding Language

```
<xs:schema
  xmlns:bbl="urn:mpeg:mpeg21:2007:01-BBL-NS"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:mpeg:mpeg21:2007:01-BBL-NS"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="1.0.0">
  <!-- **********************************************************************-->
  <!--                          Abstract Type Declarations                ** -->
  <!-- ********************************************************************* -->
  <xs:attribute name="name" type="xs:QName"/>
  <!-- *********************************************************************** -->
  <xs:attribute name="value" type="xs:string"/>
  <!-- *********************************************************************** -->
  <xs:attribute name="match" type="xs:string" default="."/>
  <!-- *********************************************************************** -->
  <xs:attribute name="handler" type="xs:IDREF"/>
  <!-- *********************************************************************** -->
  <xs:attribute name="encoding" type="xs:IDREF"/>
  <!-- *********************************************************************** -->
  <xs:attributeGroup name="timeScheme">
    <xs:attribute name="timeScheme">
      <xs:simpleType>
        <xs:union>
          <xs:simpleType>
            <xs:restriction base="xs:NCName">
              <xs:enumeration value="npt"/>
              <xs:enumeration value="smpte-24"/>
              <xs:enumeration value="smpte-24-drop"/>
              <xs:enumeration value="smpte-25"/>
              <xs:enumeration value="smpte-30"/>
              <xs:enumeration value="smpte-30-drop"/>
              <xs:enumeration value="smpte-50"/>
              <xs:enumeration value="smpte-60"/>
              <xs:enumeration value="smpte-60-drop"/>
              <xs:enumeration value="mp7t"/>
              <xs:enumeration value="clock"/>
            </xs:restriction>
          </xs:simpleType>
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:pattern value="frac\(\d+\)"/>
            </xs:restriction>
          </xs:simpleType>
          <xs:simpleType>
            <xs:restriction base="xs:string"/>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
  </xs:attributeGroup>
  <!-- *********************************************************************** -->
  <xs:complexType name="SourceElementType" abstract="true">
    <xs:attribute name="xmlSource" type="xs:string" use="optional"/>
    <xs:attribute name="binarySource" type="xs:string" use="optional"/>
    <xs:attribute name="bSrcNamespace" type="xs:string" use="optional"/>
    <xs:attributeGroup ref="bbl:timeScheme"/>
  </xs:complexType>
  <!-- ********************************************************************** -->
  <xs:complexType name="IdentifiableElementType">
    <xs:complexContent>
      <xs:extension base="bbl:SourceElementType">
        <xs:attribute name="id" type="xs:ID" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ********************************************************************** -->
  <xs:complexType name="BBLDocType">
    <xs:complexContent>
      <xs:extension base="bbl:IdentifiableElementType">
        <xs:sequence>
          <xs:element name="Declarations" minOccurs="0"
                      type="bbl:DeclarationsType"/>
          <xs:element name="Variables" minOccurs="0" type="bbl:VariablesType"/>
          <xs:element name="Register" type="bbl:RegisterType"/>
          <xs:choice maxOccurs="unbounded">
            <xs:element name="Packet" type="bbl:PacketType"/>
            <xs:element name="PacketStream" type="bbl:PacketStreamType"/>
          </xs:choice>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ********************************************************************** -->
  <xs:complexType name="AbstractVarType">
    <xs:complexContent>
      <xs:extension base="bbl:SourceElementType">
        <xs:attribute ref="bbl:name" use="required"/>
        <xs:attribute ref="bbl:value" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ********************************************************************** -->
  <xs:complexType name="ParametricElementType">
    <xs:complexContent>
      <xs:extension base="bbl:IdentifiableElementType">
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
          <xs:any namespace="##other"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ********************************************************************** -->
  <xs:complexType name="AbstractPacketType" abstract="true">
    <xs:complexContent>
      <xs:extension base="bbl:IdentifiableElementType">
        <xs:sequence>
          <xs:element name="HandlerParams" type="bbl:ParametricElementType"
minOccurs="0"/>
        </xs:sequence>
        <xs:attribute ref="bbl:handler" use="optional"/>
        <xs:attribute name="deliveryCondition" type="xs:string" use="optional"/>
        <xs:attribute name="deliveryTime" type="xs:string" use="optional"/>
        <xs:attribute name="repeat" type="xs:float" use="optional" default="0.0"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ********************************************************************** -->
  <xs:complexType name="AbstractContentType">
    <xs:complexContent>
      <xs:extension base="bbl:IdentifiableElementType">
        <xs:sequence maxOccurs="unbounded">
          <xs:any namespace="##any" processContents="skip"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="TimingElementType">
  <xs:attribute ref="bbl:match" use="optional"/>
  <xs:attribute ref="bbl:value" use="required"/>
</xs:complexType>
<!-- *********************************************************************** -->
<!-- **                    BBL concrete types                             ** -->
<!-- *********************************************************************** -->
<xs:element name="BBL">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:element name="Instance" type="bbl:InstanceType"/>
      <xs:element name="Binding" type="bbl:BindingType"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
<!-- *********************************************************************** -->
<xs:complexType name="InstanceType">
  <xs:complexContent>
    <xs:extension base="bbl:BBLDocType"/>
  </xs:complexContent>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="BindingType">
  <xs:complexContent>
    <xs:extension base="bbl:BBLDocType"/>
  </xs:complexContent>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="DeclarationsType">
  <xs:sequence maxOccurs="unbounded">
    <xs:any namespace="##any" processContents="skip"/>
  </xs:sequence>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="VariablesType">
  <xs:complexContent>
    <xs:extension base="bbl:SourceElementType">
      <xs:choice maxOccurs="unbounded">
        <xs:element name="Define" type="bbl:DefineType"/>
        <xs:element name="Assign" type="bbl:AbstractVarType"/>
      </xs:choice>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="RegisterType">
  <xs:complexContent>
    <xs:extension base="bbl:SourceElementType">
      <xs:sequence>
        <xs:element name="Handler" type="bbl:HandlerType"
                    minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="Encoder" type="bbl:EncoderType"
                     minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="Multiplexer" type="bbl:MultiplexerType"
                     minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="BSD" type="bbl:BSDType"
                     minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *********************************************************************** -->
```

```
<xs:complexType name="DefineType">
  <xs:complexContent>
    <xs:extension base="bbl:AbstractVarType">
      <xs:attribute name="type" type="bbl:VarType" use="optional" default="xs:int"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:simpleType name="VarType">
  <xs:restriction base="xs:QName">
    <xs:enumeration value="xs:integer"/>
    <xs:enumeration value="xs:int"/>
    <xs:enumeration value="xs:long"/>
    <xs:enumeration value="xs:short"/>
    <xs:enumeration value="xs:decimal"/>
    <xs:enumeration value="xs:float"/>
    <xs:enumeration value="xs:double"/>
    <xs:enumeration value="xs:boolean"/>
    <xs:enumeration value="xs:byte"/>
    <xs:enumeration value="xs:QName"/>
    <xs:enumeration value="xs:dateTime"/>
    <xs:enumeration value="xs:base64Binary"/>
    <xs:enumeration value="xs:hexBinary"/>
    <xs:enumeration value="xs:unsignedInt"/>
    <xs:enumeration value="xs:unsignedShort"/>
    <xs:enumeration value="xs:unsignedByte"/>
    <xs:enumeration value="xs:time"/>
    <xs:enumeration value="xs:date"/>
    <xs:enumeration value="xs:anySimpleType"/>
  </xs:restriction>
</xs:simpleType>
<!-- *********************************************************************** -->
<xs:complexType name="HandlerType">
  <xs:complexContent>
    <xs:extension base="bbl:ParametricElementType">
      <xs:attribute name="handlerURI" type="xs:anyURI" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="EncoderType">
 <xs:complexContent>
   <xs:extension base="bbl:ParametricElementType">
     <xs:attribute name="encoderURI" type="xs:anyURI"
                   use="required"/>
   </xs:extension>
 </xs:complexContent>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="MultiplexerType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="Channel" type="bbl:ChannelType"/>
    <xs:element name="Multiplexer" type="bbl:MultiplexerType"/>
  </xs:choice>
  <xs:attribute ref="bbl:handler" use="optional"/>
  <xs:attribute name="mode" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:NCName">
        <xs:enumeration value="packetCount"/>
        <xs:enumeration value="bandwidth"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="weight" type="xs:int" use="optional" default="1"/>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="ChannelType">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="weight" type="xs:int" use="optional" default="1"/>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="BSDType">
  <xs:attribute name="namespace" type="xs:anyURI" use="required"/>
  <xs:attribute name="bsSchemaLocation" type="xs:anyURI" use="required"/>
  <xs:attribute name="fromBinarySource" type="xs:boolean" use="optional"
                default="true"/>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="PacketType">
  <xs:complexContent>
    <xs:extension base="bbl:AbstractPacketType">
      <xs:sequence>
        <xs:element name="Content" type="bbl:AbstractContentType"/>
        <xs:element name="Variables" type="bbl:VariablesType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="PacketStreamType">
  <xs:complexContent>
    <xs:extension base="bbl:AbstractPacketType">
      <xs:sequence>
        <xs:choice>
          <xs:element name="ContentTemplate" type="bbl:AbstractContentType"/>
          <xs:element name="Bind" type="bbl:BindType"/>
        </xs:choice>
        <xs:element name="Variables" type="bbl:VariablesType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *********************************************************************** -->
<xs:complexType name="IncludeType">
  <xs:complexContent>
    <xs:extension base="bbl:SourceElementType">
      <xs:sequence>
        <xs:element name="Timing" type="bbl:TimingType" minOccurs="0"/>
        <xs:element name="Fragmentation" type="bbl:FragmentationType"
                    minOccurs="0"/>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element ref="bbl:Attribute"/>
          <xs:element name="Include" type="bbl:IncludeType"/>
          <xs:element name="Encode" type="bbl:EncodeType"/>
        </xs:choice>
      </xs:sequence>
      <xs:attribute name="ref" type="xs:string" use="required"/>
      <xs:attribute name="depth" use="optional" default="0">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="-1"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="height" use="optional" default="0">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="-1"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute ref="bbl:match" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<!-- ************************************************************************ -->
<xs:complexType name="TimingType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="DeliveryTimes" type="bbl:TimingElementType">
    </xs:element>
    <xs:element name="Durations" type="bbl:TimingElementType">
    </xs:element>
  </xs:choice>
</xs:complexType>
<!-- ************************************************************************ -->
<xs:complexType name="FragmentationType">
  <xs:sequence>
    <xs:element name="Size" type="bbl:SizeType" minOccurs="0"/>
    <xs:element name="Duration" type="bbl:DurationType" minOccurs="0"/>
    <xs:element name="Count" type="bbl:CountType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Constraint" type="bbl:ConstraintType" minOccurs="0"
                maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- ************************************************************************ -->
<xs:complexType name="SizeType">
  <xs:attribute name="value" type="xs:positiveInteger" use="required"/>
</xs:complexType>
<!-- ************************************************************************ -->
<xs:complexType name="DurationType">
  <xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
<!-- ************************************************************************ -->
<xs:complexType name="CountType">
  <xs:attribute name="value" type="xs:positiveInteger" use="required"/>
  <xs:attribute ref="bbl:match" use="optional"/>
</xs:complexType>
<!-- ************************************************************************ -->
<xs:complexType name="ConstraintType">
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:NCName">
        <xs:enumeration value="first"/>
        <xs:enumeration value="last"/>
        <xs:enumeration value="unbroken"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute ref="bbl:match" use="optional"/>
</xs:complexType>
<!-- ************************************************************************ -->
<xs:complexType name="EncodeType">
  <xs:complexContent>
    <xs:extension base="bbl:ParametricElementType">
      <xs:attribute ref="bbl:encoding" use="required"/>
      <xs:attribute ref="bbl:match" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ************************************************************************ -->
<xs:complexType name="BindType">
  <xs:complexContent>
    <xs:extension base="bbl:SourceElementType">
      <xs:sequence minOccurs="0">
        <xs:element name="Binding" type="bbl:BindingType"/>
      </xs:sequence>
      <xs:attribute name="binding" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- ************************************************************************ -->
<!-- ** Elts & Attrs used in BBL documents attached to non-bbl elements   ** -->
```

```
  <!-- *********************************************************************** -->
  <xs:element name="value-of">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="bbl:SourceElementType">
          <xs:attribute name="select" type="xs:string" use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- *********************************************************************** -->
  <xs:element name="Attribute">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="bbl:IdentifiableElementType">
          <xs:sequence>
            <xs:element ref="bbl:value-of" minOccurs="0"/>
          </xs:sequence>
          <xs:attribute ref="bbl:name" use="required"/>
          <xs:attribute ref="bbl:match" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- *********************************************************************** -->
  <!-- **  BBL attributes which are transmitted as part of xml fragments    ** -->
  <!-- *********************************************************************** -->
  <!-- *********************************************************************** -->
  <xs:attribute name="context" type="xs:string" fixed="$addr"/>
  <!-- *********************************************************************** -->
  <xs:attribute name="rap" type="xs:boolean" fixed="true"/>
  <!-- *********************************************************************** -->
  <xs:attribute name="seqNo" type="xs:integer"/>
  <!-- *********************************************************************** -->
  <xs:attribute name="processable" type="xs:boolean" default="true"/>
</xs:schema>
```

# Annex B
(informative)

# Real-time Transport Protocol Handler

## B.1 Introduction

This informative Annex defines a Real-time Transport Protocol (RTP) Handler. Implementations of this part of ISO/IEC 21000 are not required to support the RTP Handler as defined in this Annex (with type equal to that specified in B.2.2). However, implementations that do support the RTP Handler as defined in this Annex shall comply with the requirements as specified in this Annex for such an RTP Handler.

## B.2 RTP Handler

### B.2.1 Introduction

The RTP Handler shall output Packets as specified by IETF RFC3550.

Packet data passed to the Handler shall be inserted into the RTP payload. RTP header fields shall be populated generally according to the principles laid down in IETF RFC3550, and specifically as follows:

- Padding, Extension, and CSRC count are set to zero
- Marker and Payload Type are set according to the parameters specified below
- Sequence number is initialised to a random value at the commencement of the session and is incremented by one each time a Packet is output
- Timestamp is set to an offset (which is initialised with a random value) plus the Delivery Time of the Packet multiplied by the value of the timebase parameter (below)
- SSRC is assigned to a random value at the commencement of each session

One RTP Handler may be instantiated for each individual stream to be transmitted.

### B.2.2 Handler type declaration

The RTP Handler shall be instantiated by registering a `Handler` with type of value

```
    urn:mpeg:mpeg21:2007:01-BBL-NS:handler:RTP
```

See 7.3.19 for more information.

### B.2.3 Handler parameter syntax

The following W3C XMLSCHEMA fragment specifies the syntax for parameter elements which may be instantiated as child elements of a handler with type equal to that specified in B.2.2.

```
<xs:element name="timeBase" type="xs:unsignedLong"/>
<xs:element name="payloadType" type="xs:unsignedByte"/>
<xs:element name="host" type="xs:normalizedString"/>
<xs:element name="port" type="xs:unsignedShort"/>
<xs:element name="sdp" type="xs:anyURI"/>
<xs:element name="sdpMediaNo" type="xs:unsignedByte"/>
```

The following W3C XMLSCHEMA fragment specifies the syntax for parameter elements which may be instantiated as child elements of a `HandlerParams` element for a Handler with type equal to that specified in B.2.2.

```
<xs:element name="marker">
  <xs:simpleType>
    <xs:restriction base="xs:unsignedByte">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

As specified in 7.3.4 all parameter values may be delimited W3C XPATH expressions. Each W3C XPATH expression shall resolve to a value conforming to the element type.

### B.2.4 Handler parameter semantics

Table B.1 — RTP Handler parameter elements specifies semantics for the parameters of the RTP Handler.

**Table B.1 — RTP Handler parameter elements**

| Element name | Use | Semantics |
|---|---|---|
| `Hrtp:timeBase` | required | The amount which the timestamp header field is to increment each second. |
| `Hrtp:payloadType` | optional | The value which shall be used for the payload type header field. If the element is absent, the Handler shall assign a value to the payload type field based on the SDP data. |
| `Hrtp:host` | optional | The IPv4 address or host string to which the RTP session shall be directed. If this is absent the Handler shall address packets according to the SDP (if provided), or to the host which has requested the BBL. |
| `Hrtp:port` | optional | The port to which the RTP session shall be directed. If the element is absent, the Handler shall assign a port based on the SDP data. |
| `Hrtp:sdp` | optional | A URI resolving to an SDP file which shall be transmitted as the commencement of the session. If `payloadType` or `port` are absent, `sdp` shall be present. |
| `Hrtp:sdpMediaNo` | optional | A value between 0 and n-1 (where n is the number of media declarations within the SDP file) indicating the media declaration from which the payload type and/or port are to be drawn. If `payloadType` or `port` are absent, `sdpMediaNo` shall be present. |
| `Hrtp:marker` | optional | This element may optionally be provided as the child of a `HandlerParams` element to indicate the value of the marker header field for a Packet or Packet Stream. Its value may be 0 or 1. If the element is not present, the marker header field shall be zero. |

# Annex C
(informative)

# Working with BBL

## C.1 Introduction

This Annex provides additional information and examples that will assist in understanding how to work with BBL.

## C.2 Inside a BBL document

Figure C.1 — Example BBL document shows an example BBL document – in this case an Instance – which illustrates the elements found in both Instances and Bindings. This is presented as an overview, detailed information of the behaviour of each element is provided in Clause 7.

```
<Instance timeScheme="npt">

  <Declarations>
    <!-- ... -->
  </Declarations>

  <Variables>
    <Define bbl:name="SeqNo" type="xs:int" bbl:value="0"/>
    <Define bbl:name="SSRC" type="xs:int" bbl:value="0"/>
  </Variables>

  <Register>
    <Handler id="TCP" handlerURI="urn:foo:bbl:handler:tcp">
      <footcp:param1>...</footcp:param1>
      <!-- ... -->
    </Handler>
    <Handler id="RTP" handlerURI="urn:mpeg:mpeg21:2007:01-BBL-NS:handler:RTP">
      <Hrtp:timeBase>1</Hrtp:timeBase>
      <!-- ... -->
    </Handler>
    <BSD namespace="urn:bar:bsd:schema:rtp" bsSchemaLocation="schemas/RTP.xsd"/>
  </Register>

  <Packet bbl:handler="TCP" deliveryTime="0.0" repeat="99">
    <!-- ... -->
  </Packet>

  <PacketStream bbl:handler="RTP" deliveryTime="5.0">
    <!-- ... -->
  </PacketStream>

</Instance>
```

**Figure C.1 — Example BBL document**

The `Declarations` element functions equivalently to a DIDL Declaration, which is used to allow elements to be declared, without being instantiated, so that they can be referred to in other parts of the document using W3C XINCLUDE.

The Variables element defines variables which can subsequently be used in W3C XPATH expressions. Further Variables elements are found as children of Packet and PacketStream elements which can update the value of previously declared variables and define others.

The Register element is used to register the Handlers and BSD namespaces subsequently referred to in the document.

Finally, Packet and PacketStream elements specify the output of a BBL process, by declaring the content of packets which are to be provided to a Handler for output. Packet elements declare individual packets, which are generally useful for specifying content that is unique in makeup within the streamed DI (such as a Packet containing configuration information, or a Packet of metadata that is not part of a sequence). PacketStream elements, on the other hand, provide a means to declare a sequence of Packets according to a template. This allows, for example a stream of video access units, or a set of sequential metadata elements (such as subtitles or gBSD) to be specified using a single instruction. This is a fundamentally important feature, since a model that requires each individual Packet to be specified does not scale to real-world content. Further details about the operation of Packet and PacketStream are provided below.

## C.3 Adding content to a Packet

The examples below highlight the ways in which content is assembled for packetisation. In Figure 7a, a number of elements from the DIDL namespace are specified as contents of the Packet, forming a valid DID. The <did:Item> element has a context attribute attached, which is populated with the value of the $bbl:addr variable (a built-in bbl variable which evaluates to the original context of the attached item in its Source Document – see 7.3.5).

```
<Packet bbl:handler="TCP" deliveryTime="0.0"
        repeat="0.2">
  <Content>
    <did:DIDL>
      <did:Item>
        <Attribute name="bbl:context">
          <value-of select="$bbl:addr"/>
        </Attribute>
        <did:Component>
          <Include depth="2"
            ref="//did:Descriptor
            [@id='gBSD']">
            <Include depth="-1"
              ref="//did:Descriptor
              [@id='gBSD']
              //dia:DescriptionMetadata"/>
          </Include>
          <did:Resource mimeType="audio/mpeg"
            ref="theSong.mp3"/>
        </did:Component>
      </did:Item>
    </did:DIDL>
  </Content>
</Packet>
```

**Figure C.2a — Example Packet – XML**

```
<Packet bbl:handler="RTP" deliveryTime="0.0">
  <Content>
    <rtp:Header id="RTPhdr">
      <rtp:version/>
      <rtp:padding/>
      <rtp:extension/>
      <rtp:CSRC_count/>
      <rtp:marker/>
      <rtp:payload_type>
        14
      </rtp:payload_type>
      <rtp:sequence_number>
        <value-of select="$SeqNo"/>
      </rtp:sequence_number>
      <rtp:timestamp>
        <value-of select="$time"/>
      </rtp:timestamp>
      <rtp:SSRC_identifier>
        <value-of select="$SSRC"/>
      </rtp:SSRC_identifier>
    </rtp:Header>
    <rtp:Payload>
      <bbl:Include
        ref="/mp4:bitstream/mp4:VOP[1]"/>
    </rtp:Payload>
  </Content>
  <Variables>
    <Assign bbl:name="SeqNo"
            bbl:value="bbl:rnd()"/>
    <Assign bbl:name="time"
            bbl:value="$initTime + 0"/>
    <Assign bbl:name="SSRC"
            bbl:value="bbl:rnd()"/>
  </Variables>
</Packet>
```

**Figure C.2b — Example Packet – RTP**

`<did:Component>` is populated with the content returned by the `Include` elements, which are nested – so that the nodes returned by the inner element are inserted into those of the outer element *as they were in the Source Document*.

The example in Figure 7b is syntactically identical to that of the previous example. However, in this case, the namespace prefixes `rtp` and `mp4` have been declared (as per W3C XMLNAMES) for namespaces that are registered as being BSD namespaces. Consequently, when this Packet is processed, elements from these namespaces are converted to the binary equivalent specified by the associated BS Schemata. Note also that the empty elements (`<rtp:version>`, `<rtp:padding>`, and so on) are populated with their default value according to W3C XMLSCHEMA.

Further information about the way in which content is resolved is provided below (see also 7.5.2).

## C.4 Creating a stream of Packets

The content of a stream of Packets is specified using a `ContentTemplate`, which is syntactically similar to `Content` above, with the addition of `Fragmentation` Rules which define how to break the data returned by an `Include` element into fragments which are then inserted into separate Packets. Figure C.3 — Example PacketStream provides an example.

NOTE Again, the prefixes `rtp` and `mp4` are declared as namespace prefixes for namespaces registered as being BSD namespaces.

```
<PacketStream>
  <ContentTemplate>
    <xi:include xmlns:xi="http://www.w3.org/2001/XInclude" xpointer="RTPhdr"/>
    <rtp:Payload>
      <Include xpath="/mp4:Bitstream//mp4:VOP" height="1">
        <Timing>
          <DeliveryTimes match="." value="$time + ./mp4:vop_time_increment / $Modulo"/>
        </Timing>
        <Fragmentation>
          <Size value="1500"/>
          <Count value="1" match="."/>
          <Constraint type="unbroken" match="./mp4:VP"/>
          <Constraint type="first" match="."/>
        </Fragmentation>
      </Include>
    </rtp:Payload>
  </ContentTemplate>
  <Variables>
    <Assign bbl:name="SeqNo" bbl:value="$SeqNo + 1"/>
    <Assign bbl:name="time"
        bbl:value="if (rtp:Payload/mp4:VOP/mp4:modulo_time_base > 0)
                   then ($time + 1)
                   else ($time)"/>
  </Variables>
</PacketStream>
```

**Figure C.3 — Example PacketStream**

In this example, the RTP header has been previously defined and is included in the `ContentTemplate` via a W3C XINCLUDE statement. Further, the `Include` element mandates that every VOP within the MPEG-4 Video Elementary stream is to be included, but the `Fragmentation` rules specify exactly which content is to be included in each individual Packet. In this case, these rules are that the content returned by the `Include`

— is a maximum of 1500 bytes in size,

— contains at most one mp4:VOP element (which matches the "." W3C XPATH expression whose context node is that returned by the include reference),