# Blink Message XML Format Specification

beta4 - 2013-06-05

*This document specifies a method for encoding messages structured by a Blink schema into an XML format.*



## Contents



## Appendices



## 1 Overview

The core Blink specification [BLINK] defines a schema language for specifying the structure of data messages. It also defines a binary format for messages defined by such schema.

This specification provides a complementary format where messages are encoded in XML. The encoding of data values is based on the corresponding encoding specified in the Blink Tag Format Specification [TAG] as noted in the sections below.

In the XML format:

- Each message or dynamic group is mapped to an element with the same name as the source.
- Within a group, each field is mapped to an element with the same name as the field.
- Each primitive field value is represented as text, much in the same way as in the Blink Tag format [TAG].
- Sequences are represented by a sequence of XML elements, one for each item.

The following shows what the first example in the core Blink specification would look like in the XML format:

Assuming the schema:

```
Hello -> string Greeting
```

a **Hello** message carrying the greeting "Hello World" would be encoded as

```
<Hello><Greeting>Hello World</Greeting></Hello>
```

## 2 Basic Mapping

The following sections specifies how to represent Blink messages in an XML format.

### 2.1 Wrapper Element

Since an XML document can only have a single root element, the XML representation of a stream of Blink messages will always be enclosed in a single wrapper element. The name of the wrapper element is not significant. However, this specification will always call the wrapper element **root**:

```
<root>
  <Hello><Greeting>Hello</Greeting></Hello>
  <Hello><Greeting>Hello Again</Greeting></Hello>
</root>
```

### 2.2 Namespaces

Blink namespaces are mapped to XML namespaces. The URI of the XML namespace is the name of the Blink namespace. The prefix of an XML namespace is not significant in general, but this specification uses the Blink name here too for clarity. This is also the recommended method.

Assuming two Blink namespace names **Foo** and **Bar**, then the corresponding XML namespace declarations would typically be placed on the wrapper element like this:

```
<root xmlns:Foo="Foo" xmlns:Bar="Bar"> ... </root>
```

## 2.3 Message

Top-level messages and internal dynamic groups are represented as XML elements with the same name as the corresponding group. If the group is defined in a namespace, then the corresponding XML namespace will be used for the name.

Assuming a schema:

```
namespace Draw
Rect -> ...
```

then a **Draw:Rect** message would be represented as

```
<root xmlns:Draw="Draw">
  <Draw:Rect> ... </Draw:Rect>
</root>
```

**NOTE**: Throughout the rest of the document, the wrapper element and any XML namespace declarations will be left out for brevity.

## 2.4 Field

Each field of a group is represented by an XML element with the same name as the field. The element name of a field is always in the null XML namespace. The order of the field elements is not significant.

An optional field that has no value is simply left out.

Assuming a schema:

```
namespace Draw
Rect -> u32 Width, u32 Height, string Text?
```

then an instance could look like

```
<Draw:Rect>
  <Width>10</Width>
  <Height>20</Height>
</Draw:Rect>
```

or, including the optional **Text** field:

```
<Draw:Rect>
  <Text>Square</Text>
  <Width>17</Width>
  <Height>17</Height>
</Draw:Rect>
```

## 2.5 Primitive Data Types

Any data type that is not a string, binary, fixed, sequence or group is represented as character data in the XML document. The data is formatted in the same way as defined in the Blink Tag Format Specification [TAG].

## 2.6 String

A string is represented as character data. The special XML characters **<** and **&** must be escaped. If a string type has a max size property, then the string must not be longer that the specified value when represented as a [UTF-8] byte sequence.

The following example illustrates the different escaping rules between the Tag and XML formats. A message in Tag format:

```
@Exec|Command=grep Blink < /dev/random \| wc
```

would be represented like this in XML:

```
<Exec>
  <Command>grep Blink &lt; /dev/random | wc</Command>
</Exec>
```

## 2.7 Binary and Fixed

If the byte sequence of a binary or fixed value is a valid UTF-8 sequence, then it can be encoded as a string. Otherwise, a hexadecimal list must be used. A hex list is represented by a sequence of hex digits possibly separated by whitespace. The corresponding byte sequence value is obtained by removing any whitespace and translating each pair of hex digits into a byte. It is an error if the number of hex digits is not a multiple of two.

If an element contains a hex list, it must also have an attribute **binary = "yes"** to differentiate it from a normal string.

If a binary type has a max size property, then the byte sequence that results from a hex list must not be longer that the specified value. For a fixed type, the byte sequence must be exactly as long as specified in the size property.

Assuming a schema:

```
inetAddr = fixed (4)
Packet -> inetAddr Host, ...
```

then a host address can be represented like this:

```
<Packet><Host binary="yes">3e 6d 3c ea</Host> ...
```

## 2.8 Sequence

A sequence is represented as a sequence of XML elements, one for each item. Unless it is a sequence of dynamic groups, the name of the item element is not significant. The item elements are placed as children of the field element.

Assuming a schema:

```
Sample -> u32 Values []
```

then a message could look like this:

```
<Sample>
  <Values> <e>1</e> <e>2</e> <e>3</e> </Values>
</Sample>
```

## 2.9 Static Group

A static group value is represented by its fields. The field elements are placed as child elements of the field or sequence item element.

Assuming a schema:

```
Point -> u32 X, u32 Y
Line  -> Point From, Point To
Path  -> Point Points []
```

then a message could look like this:

```
<Line>
  <From> <X>0</X> <Y>0</Y> </From>
  <To> <X>10</X> <Y>10</Y> </To>
</Line>
```

In sequences of static groups, the fields are put in each item element:

```
<Path>
  <Points>
    <Point> <X>0</X> <Y>0</Y> </Point>
    <Point> <X>10</X> <Y>10</Y> </Point>
    <Point> <X>17</X> <Y>10</Y> </Point>
  </Points>
</Path>
```

The item element name in the example above was selected to be the same as the group type, **Point**. Even if this is considered good practice, any name could have been used for the item elements.

## 2.10 Dynamic Group

A dynamic group is represented in the same way as a top level message: by an XML element with the same name and namespace as the group.

If the dynamic group appears as the value of a field, then the element is placed as a child element of the field element.

Assuming a schema:

```
Shape
Rect   : Shape -> u32 Width, u32 Height
Circle : Shape -> u32 Radius
Frame  : Shape -> Shape* Content
Canvas -> Shape* [] Shapes
```

then a **Frame** is represented like this:

```
<Frame>
  <Content>
    <Rect><Width>10</Width><Height>20</Height></Rect>
  </Content>
</Frame>
```

When a dynamic group appears as the value of a sequence item, then the group element itself is used as the item element:

```
<Canvas>
  <Shapes>
    <Rect><Width>10</Width><Height>20</Height></Rect>
    <Circle><Radius>10</Radius></Circle>
  </Shapes>
</Canvas>
```

In sequences of dynamic groups, the name of the item element is significant since it identifies the dynamic type of the value of the item.

# 3 Message Extensions

Extension content of messages or internal dynamic groups is represented in the XML format by putting it in a special extension element. The name of the element is **extension** and it should be in an XML namespace with the URI **http://blinkprotocol.org/ns/blink**.

Assuming a schema:

```
Mail  -> string Subject, string Body
Trace -> string Hop
```

and an extended message:

```
@Mail|Subject=Hello|Body=How are you?|
[@Trace|Hop=local.eg.org;@Trace|Hop=mail.eg.org]
```

then it would be represented like this in XML:

```
<Mail>
  <Subject>Hello</Subject>
  <Body>How are you?</Body>
  <blink:extension
      xmlns:blink="http://blinkprotocol.org/ns/blink">
    <Trace><Hop>local.eg.org</Hop></Trace>
    <Trace><Hop>mail.eg.org</Hop></Trace>
  </blink:extension>
</Mail>
```

# 4 Whitespace

Whitespace is ignored if it appears before or after elements representing groups, fields, or sequence items.

Whitespace is significant, i.e, is part of the value, if it appears in a field or sequence item element containing a primitive value as defined here: Section 2.5 (page 2) .

# 5 Annotation Attributes

Attributes may appear freely on any element. Since any significant value or structure is represented by elements or character data, any attribute is treated as an annotation and is ignored.

## A References

| | |
|---|---|
| **BLINK** | http://blinkprotocol.org/spec/BlinkSpec-beta4.pdf |
| **TAG** | http://blinkprotocol.org/spec/BlinkTagSpec-beta4.pdf |
| **UTF-8** | http://tools.ietf.org/html/rfc3629 |