

TesLA Alliance Wiki: TBML Specification

Test Bed Definition Specification - v0.9.1

DRAFT

TesLA Alliance

23920 Freedom Circle, Santa Clara, CA 94054, USA

www.teslaalliance.org

Gale Technologies

2350 Mission College Boulevard, Suite 825, Santa Clara, CA 95054, USA

www.galetechnologies.com

History

Version	Date	Committer
0.1	01/06/2009	Patrick Deloulay
0.1b	01/09/2009	Andrew Gillis
0.2	01/21/2009	Patrick Deloulay
0.3	02/04/2009	Patrick Deloulay
0.4	04/09/2009	Patrick Deloulay
0.5	05/06/2009	Patrick Deloulay
0.6	05/28/2009	Patrick Deloulay
0.7	06/02/2009	Patrick Deloulay
0.8	06/03/2009	Patrick Deloulay
0.9	06/14-17/2009	Patrick Deloulay
0.9b	06/19/2009	Patrick Deloulay
0.9.1	07/14/2009	TechPubs

Table of Content

- Contributors
- Status of this Memo
- Abstract
- Introduction
 - Purpose
 - Guidelines
 - Terminology
- Test Bed Definition
 - Overview
 - Requirements
- TBML Schema
 - Definition
 - XML Schema
 - Structure
 - Schema Location
 - Schema Validation
- TBML Documents
 - File Extension
 - Entity Escaping
 - Compliance with Specification
- Tutorial
- Downloads and Samples
- Links
- Version Control History

Contributors

- Patrick Deloulay - Gale Technologies - Director of Engineering - patrick@galetechnologies.com
- Andrew Gillis - Gale Technologies - Software Architect - agillis@galetechnologies.com
- Kingston Duffie - Fanfare Software Group - CTO - kduffie@fanfaresoftware.com
- Cliff Hannel - IXIA - Technology Consultant - channel@ixiacom.com

Status of this Memo

This document specifies a test bed standard definition for the TesLA alliance as part of the Lab Management Technical Committee, chaired by Gale Technologies. Distribution of this memo is limited to TesLA members. This document is currently in DRAFT mode and is being reviewed by the TesLA alliance at large.

Abstract

This specification defines the Test Bed Markup Language (TBML), which is an eXtensible Markup Language (XML) application for describing test bed definitions and capturing both its structure and content. The goal of TBML is to enable test bed definitions to be served, received, and processed on the World Wide Web, just as HTML has enabled this functionality for text.

This TBML specification is intended primarily for an audience consisting of those who develop or implement renderers or editors using it, or software that communicates using TBML as a protocol for input or output. It is not a User's Guide, but rather a reference document.

TBML can be used to encode both test bed requirements and test bed environments. While TBML is human-readable, it is anticipated that, in all but the simplest cases, authors will use the Topology Resource Editor (TRE)-based test bed editors, conversion programs, and other specialized software tools to generate TBML.

Introduction

Purpose

This specification document describes the proposed need for standardization in each area, proposes the basic concept of what the specification will define, and describes how compliance with the specification can be achieved.

Guidelines

In many standard tracked documents, several words are used to signify the requirements in the specification. These words are often capitalized. Authors who follow these guidelines should incorporate this phrase near the beginning of their document:

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", this document are to be interpreted as described below.

Note: The pow of these words is modified by the requirement level of the document in which they are used. An implementation is not compliant if it fails to satisfy one or more of the MUST requirements for the protocols it implements. An implementation that satisfies all the MUST and all the SHOULD requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST requirements but not all the SHOULD requirements for its protocols is said to be "conditionally compliant."

MUST

This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

MUST NOT

This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.

SHOULD

This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different

course.

SHOULD NOT

This phrase, or the phrase "NOT RECOMMENDED", mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

MAY

This word, or the adjective "optional", means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it, or because it enhances the product, for example; another vendor may omit the same item.

Terminology

This specification uses a number of terms to refer to the roles played by participants in, and objects of, the Lab Management solution.

property

Uses a name/value format. Also a W3 type.property that can be found at resource and link definitions.

illustration

Section of the specification that allows for a graphical rendering of the TBML resources and connectivity.

HTTPS

HyperText Transfer Protocol Secure. A URL scheme that is syntactically identical to the http: scheme normally used for accessing resources using HTTP. Using an https: URL indicates that HTTP is to be used, but with a different default port (443) and an additional encryption/authentication layer between HTTP and the Transmission Control Protocol (TCP). See also URL.

link

The specification of a logical entity that connects two device interfaces/ports. Links are usually logical entities.

resource

Physical, virtual, or logical element representing an element under lab or test management.

switching infrastructure

A variety of optical and/or electrical switches and power controllers used in conjunction to form a software-controllable lab architecture managed by the Lab Manager application.

test bed

A set of hardware and software requirements required to accomplish a specific set of tasks under test, automated or manual, or mixed environments.

topology

Test lab resources under test, the logical connections between them, and attributes associated with items in the topology. A set of the lab resources under test, the logical connections between them, and attributes associated with items in the topology—one or more devices, their interfaces, and the connections between those interfaces.

TBML

Test Bed Markup Language. Based on the TBML XML Schema definition, end-users create TBML XML documents that comply to the specification.

TRE

Topology Resource Editor. A client application that allows you to view and manage resource and topology definitions.

URI

Uniform Resource Identifier. A compact string of characters used to identify or name a resource. The main purpose of this identification is to enable interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols. URIs are defined in schemes with a specific syntax and associated protocols. Also known as URL.

URL

Universal Resource Locator. A standardized addressing scheme for accessing hypertext documents and other services using a browser.

Test Bed Definition

Overview

Note: The terms, *Test Bed* and *Topology*, can be used interchangeably and have the same meaning in the context of this specification document.

The industry has been in search for higher interoperability between vendors in the same space and the market has reached a level of maturity where standardization is no longer a nice-to-have, but a must-have. Such specifications, along with a strong validation and certification program, would facilitate a more rapid adoption and the convergence of a newer set of tools to comply to the TesLA alliance.

With the current trend of lab consolidations and hard economic circumstances, Lab Managers are exploring better resource management tools to facilitate the allocation and workflow of owned resources across multiple groups, usually in a distributed environment. They are under heavy scrutiny to maximize equipment utilization and minimize overbooking of resources.

The TesLA Test Bed specifications allows lab users and consumers to define the proper set and level of requirements, highlighted resources, and connectivity between those resources—from exclusive or shared allocation—from fine-grained to coarse-grain. Such complex definitions would empower Lab Managers with the control of their environment. Thirdly, the specification would allow tool vendors to offer graphical tools to build and render the requirements.

Other benefits we foresee with the specification would make use of the same TBML language in

many areas of the lab and test automation life cycle, including:

- Test bed authoring
- Test bed reservation (scheduling and assignment)
- Test bed provisioning (equipment configuration)
- Test bed connectivity realization with switching infrastructures
- Execution of test automation logic against test beds

Requirements

Language-Independent Data Representation

The underlying representation of a test bed is performed using an XML document, formulated based on a set of XML Schemas. Defining Test Bed requirements as an XML document aids interoperability and allows flexibility to grow the data representation. Schema definitions can be found below as part of the Implementation section.

Representation of Test Bed Requirements

Should provide a mean to describe a set of devices and resources, usually taken from the overall inventory pool, that is required to perform a specific set of tasks. A topology must be uniquely identifiable when persisted and retrieved. A topology should be able to transport a great deal of metadata associated with the test bed definition (author, version, name, description, dates ...).

Representation of Concrete/Physical Device Under Test

A topology must also be able to precisely identify an actual piece of equipment (or sub-elements of the device) within the default pool of inventory.

Representation of Abstract/Logical Elements

The topology must be able to define specific (concrete) equipment necessary for testing. A topology must also be able to describe criteria for selecting items of equipment (abstract) instead of specifying an actual piece of equipment. This abstract representation is necessary when the specific equipment is unknown at the time of topology creation.

Support for Partial Equipment (Shared Devices)

An item under test may be one of many components in a larger piece of equipment. It is usually necessary to represent this relationship in the topology. It allows a user to make sense of where and what the item under test is. It also list users or applications and identifies which items are affected by actions that affect the overall piece of equipment. Consider when some ports, out of many in a device, are used in one test. Another test requires only one port, so it could use the same device and choose a different port. However, if that other test requires a device that can be safely power cycled, then it must select a port on a different device.

Support for Topology-Level Properties and Resource Properties

This provides a general mechanism for extending the description of test equipment and for attaching test data to equipment under test. These attributes could be consumed by users, test scripts, or other applications working with the topology. A common set of properties coming from TesLA specifications must be available. Third-party extension will be able to extend.

Represent Logical Connectivity Requirements between Devices Under Test

Connections define which devices will interact directly, and what their interface is to each other. A test bed definition should be able to define zero or more connectivity between elements defined. Element endpoints should be also defined when a connectivity definition is present.

Specify Software and Provisioning Requirements for Devices Under Test

Specification of a required device configuration is necessary for an automated or manual device configuration, validation of configuration, and association of specific configuration with result data.

Support for Graphical Representation

Support for diagramming is necessary to prevent graphical user interface (GUI) applications from having to implement automated layout and view de-cluttering. Having graphical support also provides a consistent visual representation when displayed on different GUIs. This consistency is critical for different engineers working together on the same topology viewed on different GUI applications. A graphical representation should allow a user-agent to render all level of devices, ports, and connectivity with some layout/coordinate capability.

Support for Topology Annotations

Annotations provide a mean to users to add or manage visual cues—post-it notes and labels—on the rendered topology. Annotations do not affect operation and structure on the topology and are for documentation purposes only. Annotations (a visual indicator for display to a user) are different from attributes (a data item that is not part of the graphical depiction).

Support for an Audit Trail

Topologies evolve overtime and an audit trail of the changes must be recorded. This requirement is not associated with topology versioning, but would offer a simple mechanism for an end-user to add history (notes) to the definition.

Allow Hierarchical Composition

Topologies must be able to be composed of other topologies. This allows topologies to be constructed from templates. It also allows a topology to be compartmentalized into sub-topologies. This facilitates the manipulation of, association of data with, or action upon entire subsets of a test bed. This can also be used to manage static vs. dynamic test beds or represent test beds as computing cloud resources.

Allows for Definition Extensions

The core test bed definitions should drive the minimum set of specifications and extension points. We would like to ensure that the language can be extended by a third party and then later

re-integrated back as part of the core, under TesLA supervision, so that TesLA benefits from the entire community.

TBML Schema

Definition

A single XML schema defines the TBML specification. A TBML author can use the following schema to define TBML documents. A graphical representation of the TBML schema can be found [here](#). The following content describes the overall structure of the XML schema and element definitions.

XML Schema

Schema Name	Target Namespace Example	Description
tbml-core.xsd	http://www.teslaalliance.org/trs/tbml/1.0	Root tbml Element definitio

TRS TBML schema target namespace will follow the proposed technical committee naming scheme

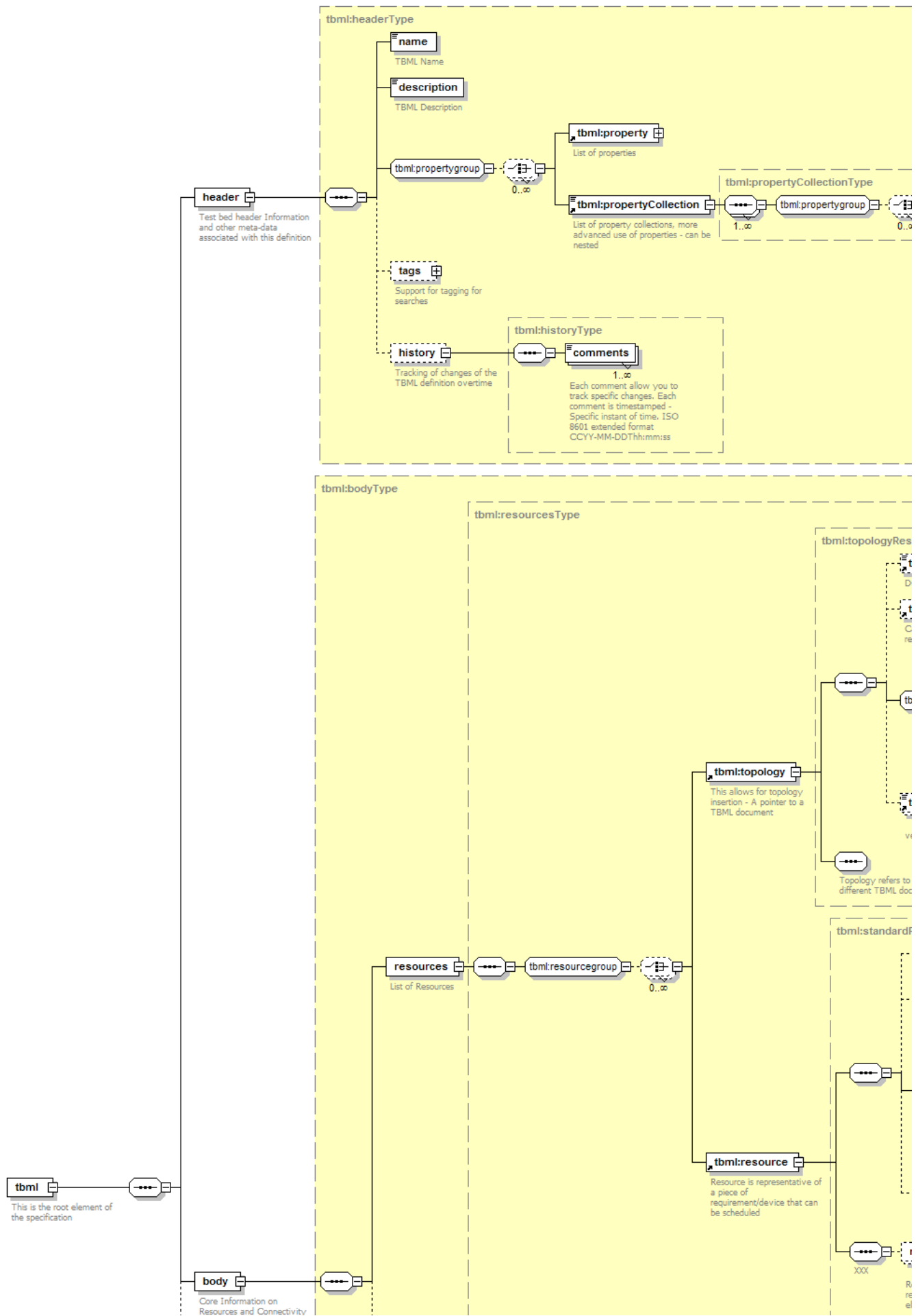
```
[TesLA web site url / committee name / schema name / current
standard version]
```

In our case, the committee name is now TRS, the TBML specifications is what we define, and 1.0 is our first version of the schema. Adding `version` into the target namespace is common practice and ensures that TBML authored-documents are validated against the proper TBML schema version. Using lowercase for the target namespace is required.

```
http://www.teslaalliance.org/trs/tbml/1.0
```

Structure

A graphical representation of the main TBML schema can be found under [tbml-core.png \(/f/tbml-core.png\)](#)

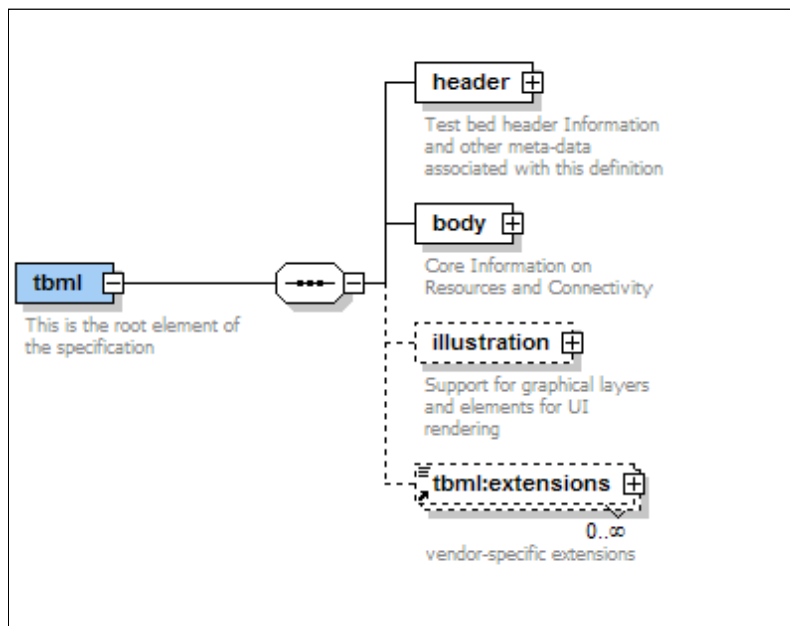


[tbml] Root Element

There are few global elements defined in the TBML core schema. Only one should be considered the root element: `<tbml>`. This root element type should be used when building TBML documents.

There are 4 sub-elements under `tbml`:

- (Required.) The `header` element contains metadata information associated with the test bed. The `name` and `Description` are formal elements of it.
- (Required.) The `body` element represents the core content of the test bed definition as a set of resources and links between those resources.
- (Optional.) The `illustration` element allows for graphical representation of user interface elements, notes and labels; [TRE \(ArchitectureCommittee\)](#)-based software vendors could take advantage of such facilities.
- The last `extensions` element allows for third party or vendor-specific content into the TBML. The `extensions` element has also been graphed to other elements definition throughout the schema.

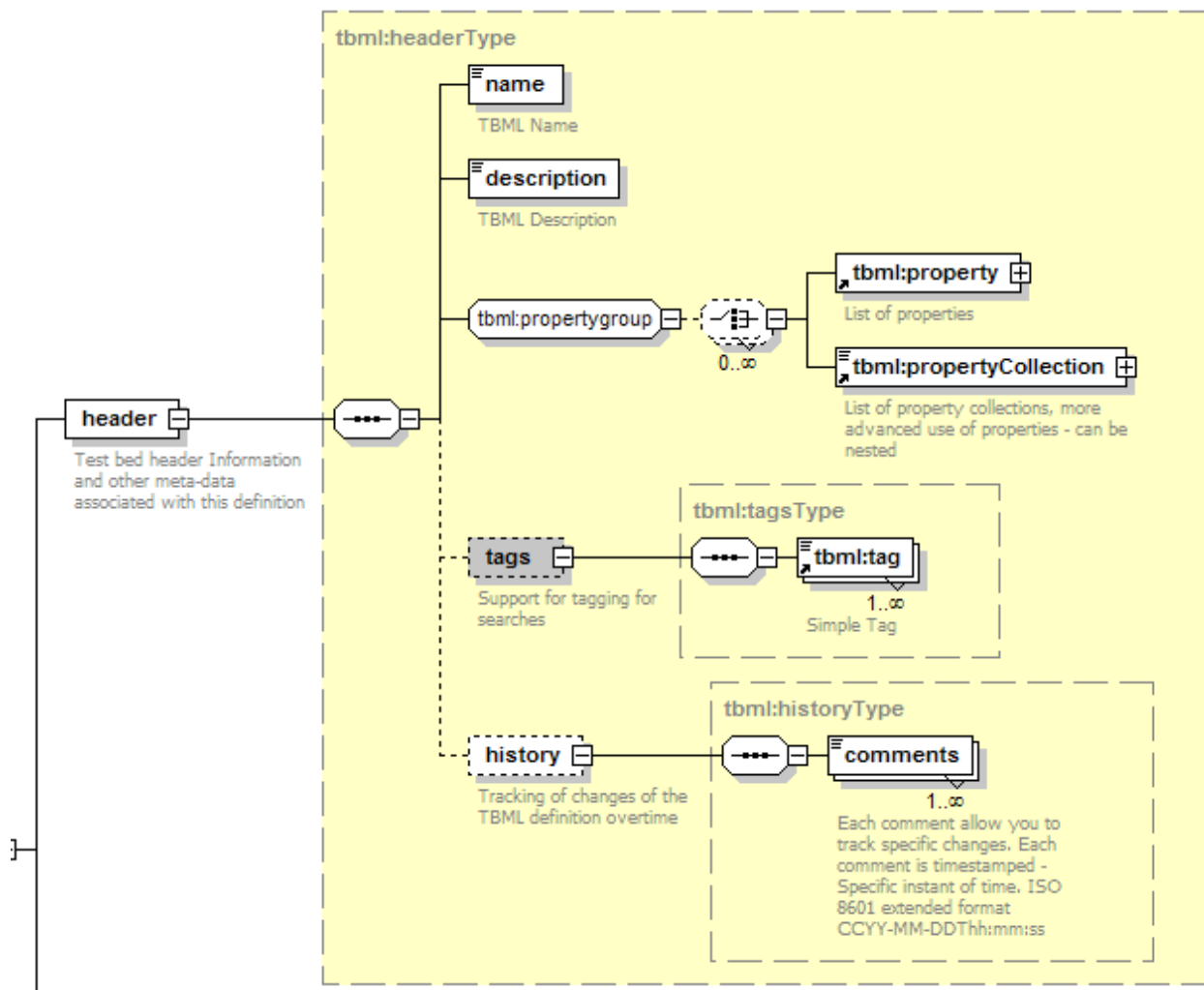


[header] Element

The header element is a required element of the `tbml` element. It supports the use of the `property` and `propertyCollection` elements metadata associated with the test bed definition. The header element has a few critical elements: 1) Required `name` element, 2) Required `description` element, and 3) Optional `tags` and `history` elements that allow end-users to transport history and provide searching capability on a large set of topologies.

Name	Type	Cardinality	Description
name	Element	1	A test bed/topology name

description	Element	1	A test bed/topology description
property	Element	0..*	Set of test bed-level properties; see Property definition
propertyCollection	Element	0..*	Set of grouped or parent-child properties; see Property Definition
tags	Element	0..1	Set of tags that can be used for indexing, searching, archiving refer Each tag within the tags list definition should contain one word or
history	Element	0..1	Container for a set of comments to track updates to the test bed /



[property] and [propertyCollection] Elements

A property element allows a user to define metadata associated with a TBML document, resource, or link element. This is a simple way to add characteristics with those element definitions. You can define basic properties using the `property` element or define more complex structures using the `propertyCollection` element. They key attributes for properties include the following:

Name	Type	Use	Default	Fixed	Annotation
------	------	-----	---------	-------	------------

name	tbml:propertyNameType	required		Name of the property. The name va dictionary and should be the prefer
vendor	string	optional	org.teslaalliance.tr	Default vendor name is set for org.t vendor-specific value should be bas e.g. com.acme.product.
type	anySimpleType	optional		The property can be typed to enfor level. Current simple types as speci Visit http://www.w3.org/TR/xmlschemer/TR/xmlschema-2/ for a complete li:

Here are few examples of property definitions

1. A derived property picked from the TBML definition in a TBML document: `<property name="Name">This is a simple name</property>`
2. A derived vendor-specific property a TBML document: `<property name="DataCenterId" vendor="com.galetechnologies.autolab" type="integer">101</property>`
3. A property can appear with same name to provide array like functionality

```

<property name="DataCenterId" vendor="com.galetechnologies.autolab"
type="integer">101</property>
<property name="DataCenterId" vendor="com.galetechnologies.autolab"
type="integer">102</property>
<property name="DataCenterId" vendor="com.galetechnologies.autolab"
type="integer">103</property>

```
4. A property collection allows for containment/hierarchical definition

```

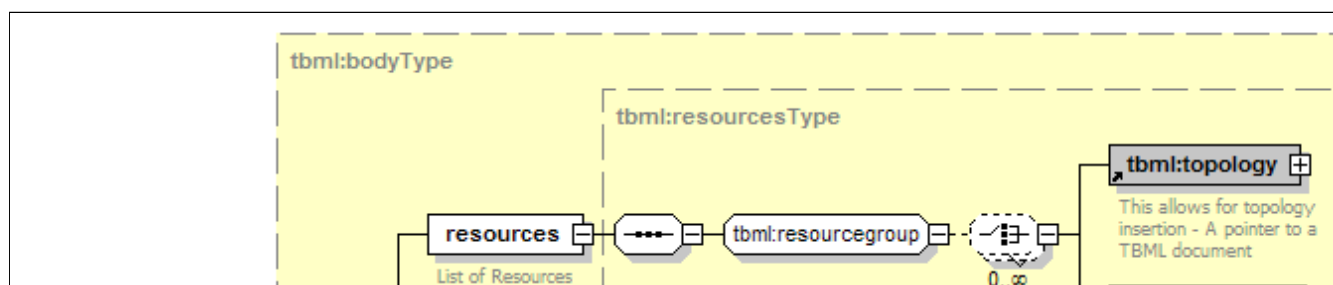
<propertyCollection name="book">
  <propertyCollection name="chapters">
    <property name="c1">Introduction</property>
    <property name="c2">History</property>
  </propertyCollection>
  <property name="author">Patrick Deloulay</property>
  <property name="title">TBML Specification</property>
</propertyCollection>

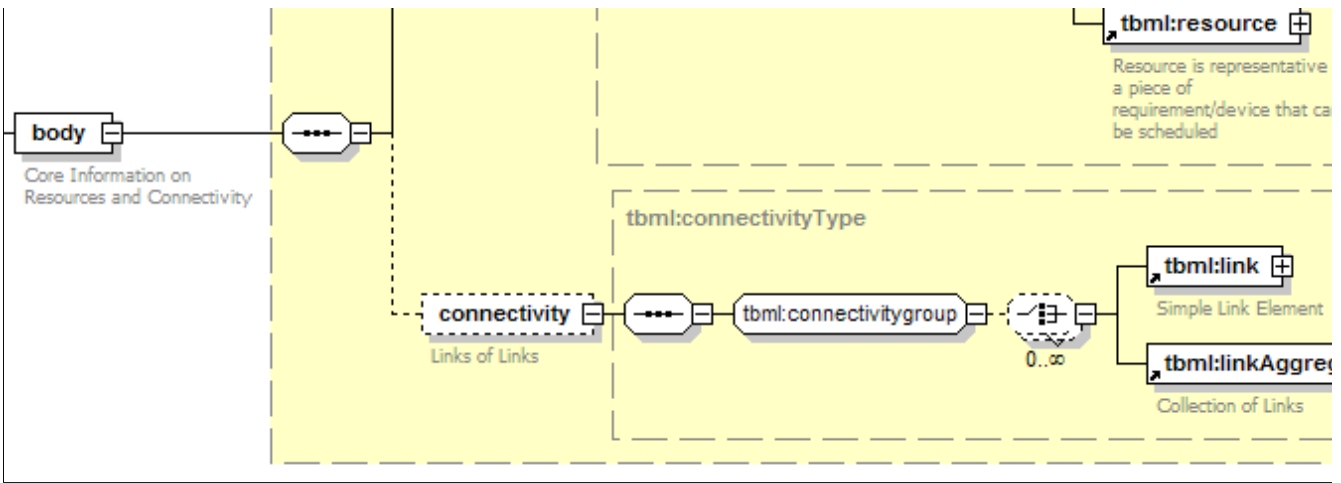
```

[body] Element

The body element is a required element of the tbml element. It contains metadata associated with the test bed you define. The two areas of interest under body are resource and connectivity definitions. At least one resource is required. Connectivity is optional.

Name	Type	Cardinality	Description
resources	Element	1..1	Required. A container for resources in the test bed/topology.
connectivity	Element	0..1	Optional. A container for all connectivity definitions.





[resources] Element

The `resources` element defines the core of the test bed definition and has information regarding all the resources and metadata associated with those resources. Two types of elements are supported to meet the requirement we established earlier: `topology` and `resource`. With no specific order, TBML documents can contain a combination of `topology` and `resource` definitions.

Each of those two base resource types support metadata associated with them with the use of the property element type. A `topology` refers to an external TBML document, which allows TBML users to refer to other test bed definitions within their own TBML without explicitly inline TBML content from other references. A `resource` definition can be used to define a simple or more complex structure. The `resource` element can also contain resources that define parent-child relationships between resources. TBML users can then describe complex devices that require a hierarchy between elements, like a card, chassis, or port.

A `resource` can contain metadata using the property elements. This is highly recommended to define your resource characteristics. Refer to the `property` element definition for more information. the `boundary` element can be used to locate the resource in a graphical canvas. The `boundary` attributes (`x`, `y`) are assumed to be relative positioning based on the parent resource if available.

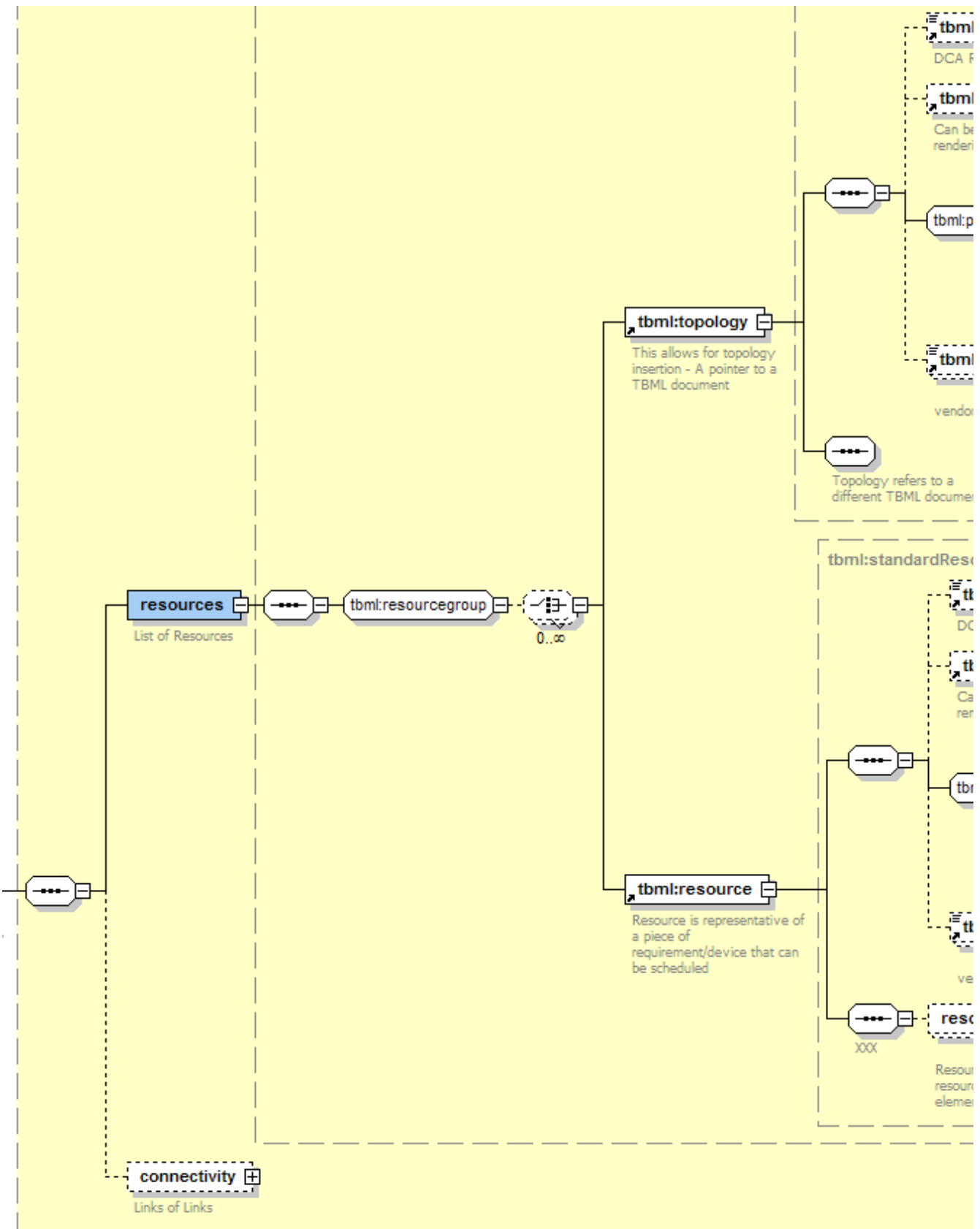
resources			
Name	Type	Cardinality	Description
topology	Element	0..*	A reference to URI describing a new TBML document. The topology can with it.
resource	Element	0..*	A resource can be used as is or can be part of a resource container. Th more complex resource definition. This can also be used for a logical ele

Topology			
Name	Type	Cardinality	Description

boundary	Element	0..1	Graphical references, if resource should be displayed.
properties	Element	0..*	Set of properties associated with this topology.

Resource

Name	Type	Cardinality	Description
dca	Element	0..1	TesLA device control adapter (DCA) information available for
boundary	Element	0..1	Graphical references if resource should be displayed
property	Element	0..*	Set of properties associated with this topology
propertyCollection	Element	0..*	Set of grouped properties
resource	Element	0..*	Set of sub-resources associated with the resource. See Po



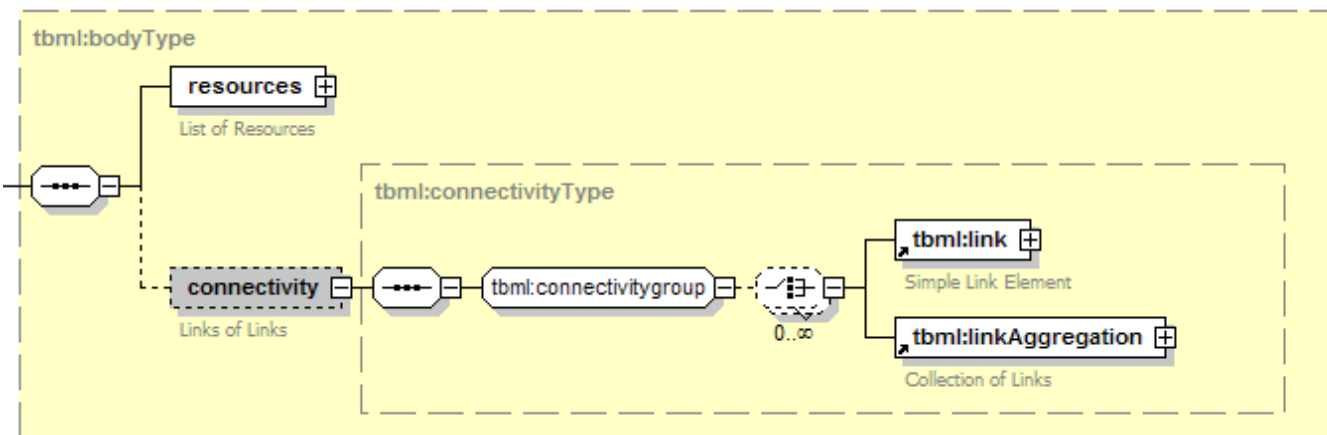
Each resource definition carries a set of attributes, as defined in the following table.

Name	Type	Use	Default	Fixed	Annotation
------	------	-----	---------	-------	------------

id	ID	required		This is a XML identifier-based type to ensure the XML document can be uniquely identified. The connectivity definition to identify both ends of the link.
guid	string	optional		A global unique identifier. This unique identifier is used by sub-systems to uniquely identify a resource. It could be a serial number, a bar code, etc. It supports possible scheduling capabilities.
shared	boolean	optional	false	This attribute is available to support scheduling. It enforces resource sharing across multiple resources. If set to true, resources can be effectively shared among multiple users. If set to false, a resource cannot be shared.
type	tbml:resourceType	required		The TRS TBML specification comes with a set of resource types (such as chassis, card, port) that allow for the definition of resource types. The type can be extended. The TBML dictionary is also extended.

[connectivity] Element

The connectivity element type describes the connectivity between the resources defined above. The support for [link] and [linkAggregation] is available so that links can be bundled into a logical link.



A link element supports the property and propertyCollection definitions. A link is comprised of two endpoint elements that must point to specific resources defined in the topology. The endpoint element can also transport some directional attribute value (direction), but this is optional. Having one endpoint be a topology-based resource should not be recommended at this time. It also supports the tbml:extensionelement so that specific-vendor information can also be tied to the link definition. A linkAggregation element allows for grouping of simple link elements. It is a simple collection of links with a group name attribute. **Note:** The linkAggregation element also supports properties to be attached.

[illustration] Element

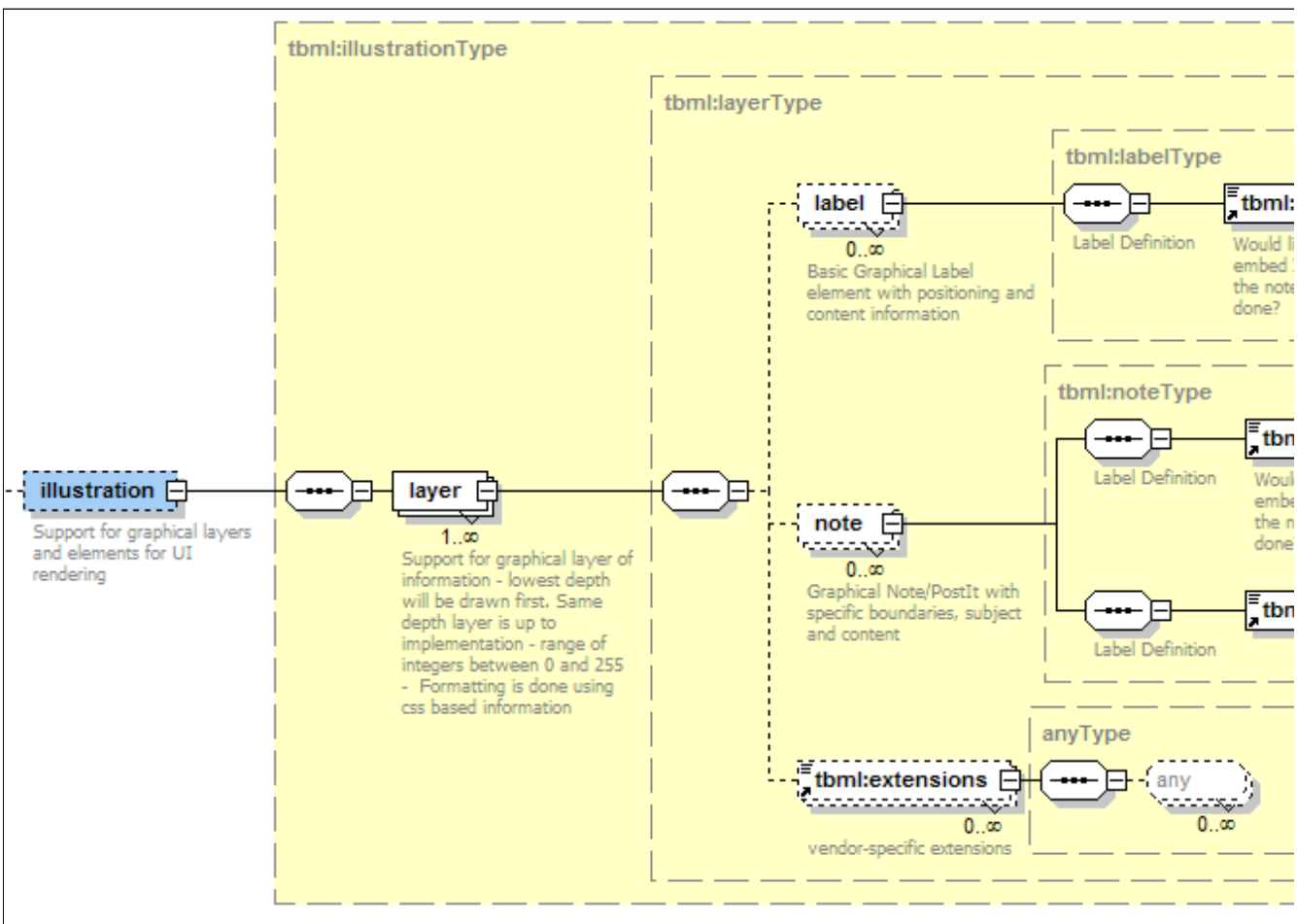
The illustration element is an optional element of the tbml element. It carries elements that would allow a TRE-based software vendor to graphically represent notes and labels

associated with each test bed definition.

At least one `layer` element should be defined. Each layer can contain a set of labels (`label` element) and notes (`note` element); a note being richer in terms of a UI element, because it can also transport a heading (`subject`). Each element can be displayed or hidden based on its `visibility` attribute. The entire layer can also be displayed or not. A layer also supports third party or vendor-specific information to be plugged-in using the `tbml:extensions` element.

Name	Type	Cardinality	Description
layer	Element	1..*	At least one layer to be defined. Each layer can contain label and notes. Those using a cascading style sheet (CSS).

Name	Type	Cardinality	Description
label	Element	0..*	Basic graphical label that can be placed into the test red rendering.
note	Element	0..*	Richer element that contains a subject and content.
extensions	Element	0..*	Support for third party or vendor-specific information that must be carried



Schema Location

TesLA should host all schemas in a single container on their web site so that TBML and other

documents can download specific XML schema versions.

The proposal is to use the target namespace name as a valid URL on the TesLA web site.

Schema Validation

- The [tbml-core.xsd](#) schema has passed the W3 schema validation online tool.
- Passed W3 Schema validation tool (XSV version: XSV 3.1-1 of 2007/12/11 16:20:05)
- <http://www.w3.org/2001/03/webdata/xsv> (<http://www.w3.org/2001/03/webdata/xsv>)

TBML Document

File Extension

The TBML document will have a `.tbml` extension (e.g. `my-favorite-test-bed.tbml`)

Entity Escaping

Your TBML file must be UTF-8 encoded (you can generally do this when you save the file). As with all XML files, any data values (including URLs) must use entity escape codes for the characters listed in the table below.

Character		Escape Code
Ampersand	&	<code>&amp;</code>
Single Quote	'	<code>&apos;</code>
Double Quote	"	<code>&quot;</code>
Greater Than	>	<code>&gt;</code>
Less Than	<	<code>&lt;</code>

In addition, all URLs must be URL-escaped and encoded for readability. However, if you are using any sort of script, tool, or log file to generate your URLs (anything except typing them in by hand), this is usually already done for you. Please verify that your URLs follow the RFC-3986 standard for URIs, the RFC-3987 standard for IRIs, and the XML standard.

Below is an example of a URL that uses a non-ASCII character (ü), as well as a character that requires entity escaping (&):

`http://www.example.com/ümlat.php&q=name`

Below is that same URL, ISO-8859-1 encoded (for hosting on a server that uses that encoding) and URL escaped:

`http://www.example.com/%FCmlat.php&q=name`

Below is that same URL, UTF-8 encoded (for hosting on a server that uses that encoding) and

URL escaped:

`http://www.example.com/%C3%BCmlat.php&q=name`

Below is that same URL, but also entity escaped:

`http://www.example.com/%C3%BCmlat.php&q=name`

Compliance with Specification

An application can achieve compliance with the topology specification by reading and/or writing structurally valid topology XML. Structurally valid refers to meeting the rules of a topology structure—not just providing XML validation. For example, having the same concrete device represented multiple times in a topology would be structurally invalid, even though the XML alone may be valid.

A set of XML-based tools support built-in validation and well-formness of any XML document based on their schema definition.

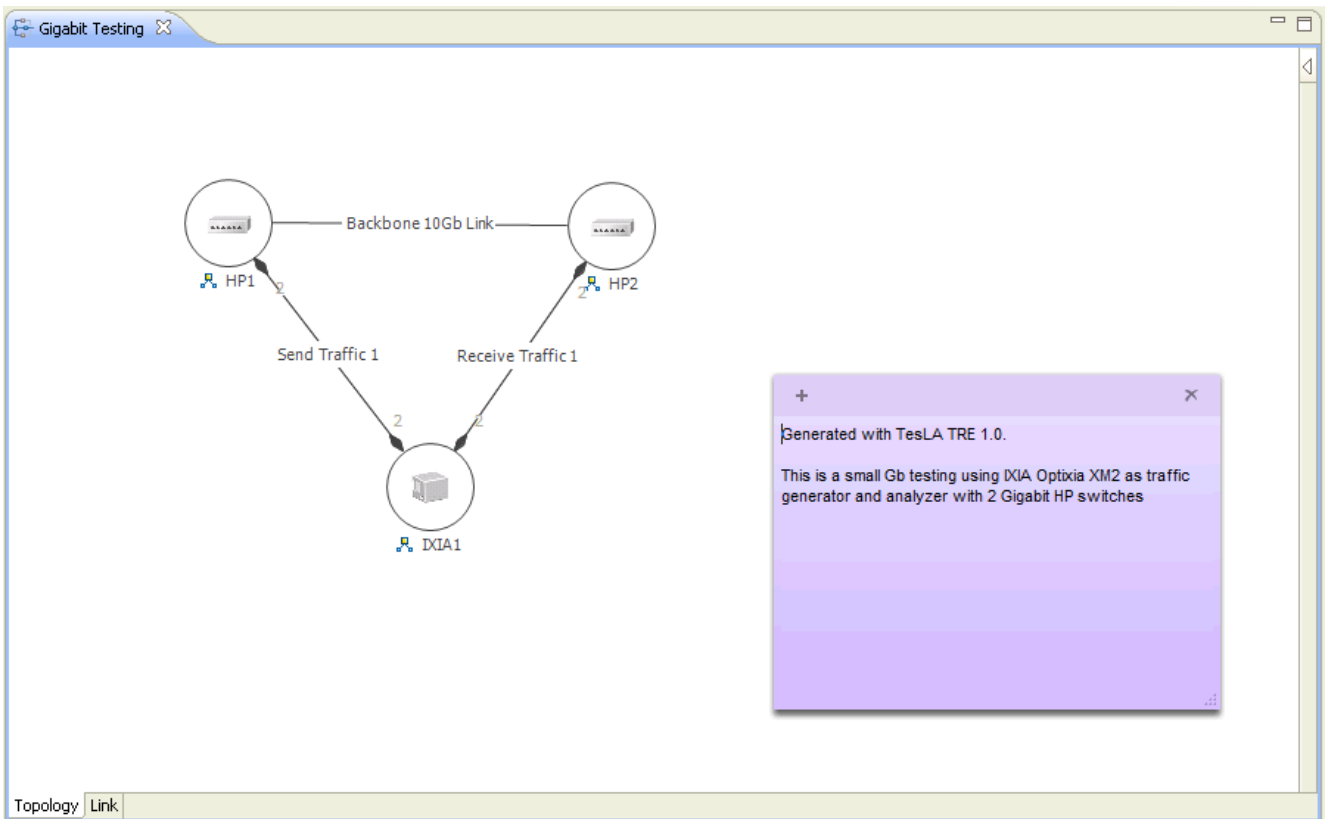
Tutorial—Building a Simple TBML Document

(Coming Soon)

Downloads and Samples

The following downloads and samples will be packaged with the release of the specifications and available as well during the review certification period:

- TBML Schemas ([tbml-core.xsd](#))
- Auto-generated TBML Schema description ([tbml-core.mht \(/f/tbml-core.mht\)](#))
- Example 1: [minimal.tbml \(/f/minimal.tbml\)](#)—Minimal test bed—limited header, one resource, no annotations
- Example 2: [example.tbml \(/f/example.tbml\)](#)—More complex test bed definition—complex resources, full annotations
- Example 3: Gigabit Testing—Two HP switches with one IXIA box
 - TRE-based rendering



- TBML generated by TRE ([gigabittesting \(/f/example.tbml\) .tbml \(/f/example.tbml\)](#)) as specified by the end-user. All three generic boxes; it also carries vendor-specific information for provisioning the two HP switches.
- TBML published by scheduler ([gigabittesting-post.tbml \(/f/example.tbml\)](#)) ([/f/gigabittesting-post.tbml \(/f/example.tbml\)](#)). The scheduler has overloaded the TBML with concrete property/value pairs associated with the resources.
- Example 4: OnPath Simple Discovery-based TBML output ([onpath.tbml \(/f/example.tbml\)](#)) ([/f/onpath.tbml \(/f/example.tbml\)](#)).

Links

Description Notes

Gale Technologies: www.galetechnologies.com

TesLA Alliance: www.teslaalliance.org

Version Control History

1.0.0.68

- Fixed Issues reported on XML Schema Validation and Sample files
- Upgraded XML Schema Development Tool to catch errors
 - Updated propertyGroup cardinality
 - Validated against XMLSpy Enterprise 2009
 - Validated against W3 Validation Tool - XSV version: XSV 3.1-1 of 2007/12/11 16:20:05 - <http://www.w3.org/2001/03/webdata/xsv> (<http://www.w3.org/2001/03/webdata/xsv>)

1.0.0.62

- Move basicLinkType as abstract element
- Moved out end-point into standardLinkType instead

1.0.0.60

- Added <extensions> element at the <tbml>, <resource>, <link> and <layer> to allow for vendor specific integrations
- New Comments in Schema with TesLA
- Removed <resource> from <topology>
- Updated <boundary> element for resources

1.0.0.52

Property element attribute 'type' accepts any basic XSD driven datatypes

- - <http://www.w3.org/TR/xmlschema-2/#dt-anySimpleType> (<http://www.w3.org/TR/xmlschema-2/#dt-anySimpleType>)
- Resource element 'shared' attribute default value is now set to false
- introducing an optional <dca> element with min, max and preferred revision scheme.
 - Enforcing x.y.z.t (at least one .) using the dcaRevisionType
 - ```
<simpleType name="dcaRevisionType">
 <annotation>
 <documentation>any x.y.z.t pattern is considered valid starting with x.y</documentation>
 </annotation>
 <restriction base="string">
 <pattern value="[0-9]+(\.[0-9]+){1,3}"/>
 </restriction>
</simpleType>
```

### 1.0.0.51

- [schemas] Introduced propertyCollection element type to allow for more complex property management (list, hash)
- [schemas] Allow for 'grouping of Property and PropertyCollection in Headers, Resources and Links
- [schemas] Updated UI, Label and Note elements with newer attributes
- [schemas] Updated Link Aggrgation Element Definition

### 1.0.0.48

- [schemas] Moved guid from property into Resource attribute - Scheduler requirement
- [schemas] Removed Link Type attribute
- [schemas] Updated Link Element
- [schemas] Trying to get XHTML content into Illustration elements (In Progress)
- [samples] updated to match the new spec - cleaned up casing

### 1.0.0.47

- [schemas] Removed .tbml-extension schema and moved it into core.xsd
- [schemas] Updated Resource definition to support sub-resources. Remvoed container element to avoid confusion.
- [schemas] Introduce Resource Type to distinguish devices from port based resources.
- [schemas] Updated Topology resource definition with URI
- [schemas] Reset cardinality for property element to 0 as minimum
- [schemas] Moved Dictionary value into lowercasing

- [samples] Updated to match the new specification

#### **1.0.0.40**

- [schemas] Resource, topology, and container are now moved into a group choice instead of sequence, allowing any order positioning when building XML
- [samples] Updated to match the new specification

#### **1.0.0.38**

- [schemas] Removed property modifier attribute
- [schemas] Renamed namespace into vendor property attribute name
- [samples] Updated to match the new specification

-----  
EOD

# Analytics

## Visitors

- [Patrick Deloulay \(http://teslaalliance.pbworks.com/user/29e456aaec004d320ce39963ecbaff84d09e0d5f\)](http://teslaalliance.pbworks.com/user/29e456aaec004d320ce39963ecbaff84d09e0d5f) (you)  
now

## Counter

17