Synchronized Multimedia Integration Language Document Object Model

W3C Working Draft *15 November, 1999*

This version:
http://www.w3.org/TR/1999/WD-smil-boston-dom-19991115
(Plain text file, single HTML file, PDF file, PostScript file, ZIP file)
Latest version:
http://www.w3.org/TR/smil-boston-dom
Previous versions:
http://www.w3.org/1999/08/WD-smil-boston-19990820

Editors:
Philippe Le Hégaret, *W3C*
Patrick Schmitz, *Microsoft*

Status of this document

This is a W3C Working Draft for review by W3C members and other interested parties. It is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by, or the consensus of, either W3C or members of the SYMM working group.

This document has been produced as part of the W3C Multimedia Activity. The authors of this document are the SYMM WG members.

This document is for public review. Comments on this document should be sent to the public mailing list www-smil@w3.org.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Abstract

This specification defines the Document Object Model (DOM) specification for synchronized multimedia functionality. It is part of work in the Synchronized Multimedia Working Group (SYMM) towards a next version of the SMIL language and SMIL modules. Related documents describe the specific application of this SMIL DOM for SMIL documents and for HTML and XML documents that integrate SMIL functionality. The SMIL DOM builds upon the DOM Core functionality, adding support for timing and synchronization, media integration and other extensions to support synchronized multimedia documents.

Table of contents

Expanded Table of Contents

Copyright Notice

1. Introduction

(***ED:*** In all the interfaces defined in the draft, the details need discussion and review. Do not assume that the defined types, return values or exceptions described are final. The IDL interfaces will be moved to specific module documents once they are ready.)

The first W3C Working Group on Synchronized Multimedia (SYMM) developed SMIL - Synchronized Multimedia Integration Language. This XML-based language is used to express synchronization relationships among media elements. SMIL 1.0 documents describe multimedia presentations that can be played in SMIL-conformant viewers.

SMIL 1.0 did not define a Document Object Model. Because SMIL is XML based, the basic functionality defined by the Core DOM is available. However, just as HTML and CSS have defined DOM interfaces to make it easier to manipulate these document types, there is a need to define a specific DOM interface for SMIL functionality. The current SYMM charter includes a deliverable for a SMIL-specific DOM to address this need, and this document specifies the SMIL DOM interfaces.

Broadly defined, the SMIL DOM is an Application Programming Interface (API) for SMIL documents and XML/HTML documents that integrate SMIL functionality. It defines the logical structure of documents and the way a document is accessed and manipulated. This is described more completely in *"What is the Document Object Model"*.

The SMIL DOM will be based upon the DOM Core functionality (see DOM Core: The SMIL DOM Foundation [p.11]). This describes a set of objects and interfaces for accessing and manipulating document objects. The SMIL DOM will also require the additional event interfaces described in the *DOM Level 2 Events module*. The SMIL DOM extends these interfaces to describe elements, attributes, methods and events specific to SMIL functionality.

Note that the SMIL DOM does not require support for DOM Level 2 Views, Stylesheets, CSS, Traversal, and Model Range modules.

The SYMM Working Group is also working towards a *modularization of SMIL functionality*, to better support integration with HTML and XML applications. Accordingly, the SMIL DOM is defined in terms of the SMIL modules.

2. Requirements

The design and specification of the SMIL DOM must meet the following set of requirements.

General requirements:

- Inherit DOM Level 2 core functionality - the SMIL DOM will be based upon the generic Core DOM foundation.
- Support constraints on DOM core functionality (e.g. mutation), especially at runtime.
- Support both SMIL documents as well as hybrid documents that integrate XML/HTML and SMIL functionality.
- Support and reflect the modularization of SMIL functionality. It must be possible to design hybrid documents combining XML/HTML and SMIL functionality, that can in turn support a hybrid or combined DOM.

SMIL specific requirements:

- Support SMIL specific elements. It must be possible to access and manipulate all SMIL elements within a SMIL document or a hybrid document that integrates XML and SMIL modules.
- Support SMIL specific attributes and methods. It must be possible to access and manipulate the SMIL attributes and methods on SMIL elements as well as XML/HTML elements in a hybrid documents.
- Support SMIL specific events, including:
  - Specification of statically defined SMIL events
  - Support for dynamically defined events
  - The ability to create and raise all the above events
- Support SMIL semantics. This includes various constraints on document structure, attribute values and method invocation.
- Define basic control interface for media player/renderers. Do not define a plug-in API, but rather just the timing and sync control interface.

It is not yet clear what all the requirements ([DOM Requirements]) on the SMIL DOM will be related to the modularization of SMIL functionality. While the HTML Working Group is also working on modularization of XHTML, a modularized HTML DOM is yet to be defined. In addition, there is no general mechanism yet defined for combining DOM modules for a particular profile.

3. DOM Core: The SMIL DOM Foundation

The SMIL DOM has as its foundation the Core DOM. The SMIL DOM includes the support defined in the DOM Level 2 Core, and the DOM Level 2 Events.

3.1. DOM Level 2 Core

The *DOM Level 2 Core* describes the general functionality needed to manipulate hierarchical document structures, elements and attributes. The SMIL DOM describes functionality that is associated with or depends upon SMIL elements and attributes. Where practical, we would like to simply inherit functionality that is already defined in the DOM Level 2 Core. Nevertheless, we want to present an API that is easy to use, and familiar to script authors that work with the HTML DOM definitions.

Following the *pattern of the HTML DOM*, the SMIL DOM defines a naming convention for properties, methods, events, collections and data types. All names are defined as one or more English words concatenated together to form a single string. The property or method name starts with the initial keyword in lowercase, and each subsequent word starts with a capital letter. For example, a method that converts a time on an element local timeline to global document time might be called "localToGlobalTime". The attribute "xml:link" will be called "xmlLink" in DOM.

3.1.1. Properties and methods

In the ECMAScript binding, properties are exposed as properties of a given object. In Java, properties are exposed with get and set methods.

Most of the properties are directly associated with attributes defined in the SMIL syntax. By the same token, most
(***ED:*** (or all?))
of the attributes defined in the SMIL syntax are reflected as properties in the SMIL DOM. There are also additional properties in the DOM that present aspects of SMIL semantics (such as the current position on a timeline).

The SMIL DOM methods support functionality that is directly associated with SMIL functionality (such as control of an element timeline).

3.1.2. Constraints on Core interfaces

In some instances, the SMIL DOM defines constraints on the Level 2 Core interfaces. These are introduced to simplify the SMIL associated runtime engines. The constraints include:

- Read-only properties, precluding arbitrary manipulation of the SMIL element properties at runtime.
- Disallowed structural changes, precluding certain changes to the structure of the document (and the associated time graph) at runtime.

(***ED:*** This section will need to be reworked once we have a better handle on the approach we take (w.r.t. modality, etc.) and the details of the interfaces.)
(***ED:*** We probably also want to include notes on the recent discussion of a presentation or runtime object model as distinct from the DOM.)

3.2. DOM Level 2 Event Model

One of the goals of the DOM Level 2 Event Model is the design of a generic event system which allows registration of event handlers, describes event flow through a tree structure, and provides basic contextual information for each event. The SMIL event model includes the definition of a standard set of events for synchronization control and presentation change notifications, a means of defining new events dynamically, and the defined contextual information for these events.

3.2.1. SMIL and DOM Level 2 events

The DOM Level 2 Events specification currently defines a base Event interface and three broad event classifications:

- UI events
- UI Logical events
- Mutation events

In HTML documents, elements generally behave in a passive (or sometimes reactive) manner, with most events being user-driven (mouse and keyboard events). In SMIL, all timed elements behave in a more active manner, with many events being content-driven. Events are generated for key points or state on the element timeline (at the beginning, at the end and when the element repeats). Media elements generate additional events associated with the synchronization management of the media itself.

The SMIL DOM makes use of the general UI and mutation events, and also defines new event types, including:

- Object Temporal Events
- Logical Temporal Events
- Synchronization Events
- Media-delivery Events

Some runtime platforms will also define new UI events, e.g. associated with a control unit for web-enhanced television (e.g. channel change and simple focus navigation events). In addition, media players within a runtime may also define specific events related to the media player (e.g. low memory).

The SMIL events are grouped into four classifications:

Static SMIL events
This is a group of events that are required for SMIL functionality. Some of the events have more general utility, while others are specific to SMIL modules and associated documents (SMIL documents as well as HTML and XML documents that integrate SMIL modules).

Platform and environment specific events
: These events are not defined in the specification, but may be created and raised by the runtime environment, and may be referenced by the SMIL syntax.

Author-defined events
: This is a very important class of events that are not specifically defined in the DOM, but that must be supported for some common use-case scenarios. A common example is that of broadcast or streaming media with embedded triggers. Currently, a media player exposes these triggers by calling script on the page. To support purely declarative content, and to support a cleaner model for script integration, we allow elements to raise events associated with these stream triggers. The events are identified by names defined by the author (e.g. "onBillWaves" or "onScene2"). Declarative syntax can bind to these events, so that some content can begin (or simply appear) when the event is raised. This is very important for things like Enhanced Television profiles, Enhanced DVD profiles, etc. This functionality is built upon the DOM Level 2 Events specification.

Property mutation events
: These are mutation events as defined in the DOM Level 2 Events specification. These events are raised when a particular property is changed (either externally via the API, or via internal mechanisms).
: Note that SMIL Animation does not "change properties" in the manner referenced above, and so does not generate property mutation events. For details, see the *SMIL Animation* specification.

3.3. Event propagation support

In addition to defining the basic event types, the DOM Level 2 Events specification describes event flow and mechanisms to manipulate the event flow, including:

- Event Capturing
- Event Bubbling
- Event Cancellation

The SMIL DOM defines the behavior of Event capture, bubbling and cancellation in the context of SMIL and SMIL-integrated Documents.

In the HTML DOM, events originate from within the DOM implementation, in response to user interaction (e.g. mouse actions), to document changes or to some runtime state (e.g. document parsing). The DOM provides methods to register interest in an event, and to control event capture and bubbling. In particular, events can be handled locally at the target node or centrally at a particular node. This support is included in the SMIL DOM. Thus, for example, synchronization or media events can be handled locally on an element, or re-routed (via the bubbling mechanisms) to a parent element or even the document root. Event registrants can handle events locally or centrally.

Note: It is currently not resolved precisely how event flow (dispatch, bubbling, etc.) will be defined for SMIL timing events. Especially when the timing containment graph is orthogonal to the content structure (e.g. in XML/SMIL integrated documents), it may make more sense to define timing event flow relative to the timing containment graph, rather than the content containment graph. This may also cause problems, as different event types will behave in very different ways within the same document.

Note: In Documents using SMIL Layout, it is currently not resolved precisely how certain user interface events (e.g. onmouseover, onmouseout) will be defined and will behave. It may make more sense to define these events relative to the regions and layout model, rather than the timing graph.

4. Constraints imposed upon DOM

We have found that the DOM has utility in a number of scenarios, and that these scenarios have differing requirements and constraints. In particular, we find that editing application scenarios require specific support that the browser or runtime environment typically does not. We have identified the following requirements that are directly associated with support for editing application scenarios as distinct from runtime or playback scenarios:

4.1. Document modality

Due to the time-varying behavior of SMIL and SMIL-integrated document types, we need to be able to impose different constraints upon the model depending upon whether the environment is editing or browsing/playing back. As such, we need to introduce the notion of modality to the DOM (and perhaps more generally to XML documents). We need a means of defining modes, of associating a mode with a document, and of querying the current document mode.

We are still considering the details, but it has been proposed to specify an active mode that is most commonly associated with browsers, and a non-active or editing mode that would be associated with an editing tool when the author is manipulating the document structure.

4.2. Node locking

Associated with the requirement for modality is a need to represent a lock or read-only qualification on various elements and attributes, dependent upon the current document mode.

For an example that illustrates this need within the SMIL DOM: To simplify runtime engines, we want to disallow certain changes to the timing structure in an active document mode (e.g. to preclude certain structural changes or to make some properties read-only). However when editing the document, we do not want to impose these restrictions. It is a natural requirement of editing that the document structure and properties be mutable. We would like to represent this explicitly in the DOM specification.

There is currently some precedent for this in HTML browsers. E.g. within Microsoft Internet Explorer, some element structures (such as tables) cannot be manipulated while they are being parsed. Also, many script authors implicitly define a "loading" modality by associating script with the document.onLoad event. While this mechanism serves authors well, it nevertheless underscores the need for a generalized model for document modality.
(***ED:*** The node locking could be currently supported with the `DOMException` `NO_MODIFICATION_ALLOWED_ERR`.)

4.3. Grouped, atomic changes

A related requirement to modality support is the need for a simplified transaction model for the DOM. This would allow us to make a set of logically grouped manipulations to the DOM, deferring all mutation events and related notification until the atomic group is completed. We specifically do not foresee the need for a DBMS-style transaction model that includes rollback and advanced transaction functionality.

We are prepared to specify a simplified model for the atomic changes. For example, if any error occurs at a step in an atomic change group, the atomicity can be broken at that point.

As an example of our related requirements, we will require support to optimize the propagation of changes to the time-graph modeled by the DOM. A typical operation when editing a timeline shortens one element of a timeline by trimming material from the beginning of the element. The associated changes to the DOM require two steps:

- Change the begin time of the element to be later in time
- Change the duration of the element to preserve the end time

Typically, a timing engine will maintain a cache of the global begin and end times for the elements in the timeline. These caches are updated when a time that they depend on changes. In the above scenario, if the timeline represents a long sequence of elements, the first change will propagate to the whole chain of time-dependents and recalculate the cache times for all these elements. The second change will then propagate, recalculating the cache times again, and restoring them to the previous value. If the two operations could be grouped as an atomic change, deferring the change notice, the cache mechanism will see no effective change to the end time of the original element, and so no cache update will be required. This can have a significant impact on the performance of an application.

When manipulating the DOM for a timed multimedia presentation, the efficiency and robustness of the model will be greatly enhanced if there is a means of grouping related changes and the resulting event propagation into an atomic change.

5. SMIL Document Object Model

A DOM application can use the `hasFeature` method of the `DOMImplementation` interface to determine whether the SMIL Object Model interfaces are supported or not. The feature string for the fundamental interfaces listed in this section is "SMIL". The version number of this feature for SMIL Boston is "2.0".

The purpose of the SMIL Boston DOM is to provide an easy access to the attributes and functionalities of the SMIL Boston specification ([SMIL Boston]). Not all SMIL Boston attributes are accessible with the following API but the user can still use the DOM Core ([DOM Level 2]) to access them (see *setAttributeNS* and *getAttributeNS*).

5.1. SMIL Core extensions

(***ED:*** A separate document should describe the integrated DOM associated with SMIL documents, and documents for other document profiles (like HTML and SMIL integrations).)

Interface *SMILDocument*

A SMIL document is the root of the SMIL Hierarchy and holds the entire content. Beside providing access to the hierarchy, it also provides some convenience methods for accessing certain sets of information from the document.
(***ED:*** Cover document timing, document locking?, linking modality and any other document level issues. Are there issues with nested SMIL files?
Is it worth talking about different document scenarios, corresponding to differing profiles? E.g. Standalone SMIL, HTML integration, etc.)
IDL Definition

```
interface SMILDocument : Document {
};
```

Interface *SMILElement*

The `SMILElement` interface is the base for all SMIL element types. It follows the model of the `HTMLElement` in the HTML DOM, extending the base `Element` class to denote SMIL-specific elements.

Note that the `SMILElement` interface overlaps with the `HTMLElement` interface. In practice, an integrated document profile that include HTML and SMIL modules will effectively implement both interfaces (see also the DOM documentation discussion of *Inheritance vs Flattened Views of the API*).
(***ED:*** // etc. This needs attention)
IDL Definition

```
interface SMILElement : Element {
          attribute DOMString        id;
                                       // raises(DOMException) on setting

};
```

Attributes

`id` of type `DOMString`
The unique id.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

5.2. Structure extensions

(***ED:*** This module will include the smil, head and body elements.)

5.3. Meta informations extensions

(***ED:*** The following interfaces are based on SMIL 1.0 and will change in future versions.)

Interface *SMILMetaElement*

@@TODO.
IDL Definition

```
interface SMILMetaElement : SMILElement {
          attribute DOMString        content;
                                       // raises(DOMException) on setting

          attribute DOMString        name;
                                       // raises(DOMException) on setting

          attribute DOMString        skipContent;
                                       // raises(DOMException) on setting

};
```

Attributes

`content` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`name` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`skipContent` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

5.4. Layout extensions

(***ED:*** The following interfaces are based on SMIL 1.0 and will change in future versions.)

This module includes the layout, root_layout and region elements, and associated attributes.

Interface *SMILLayoutElement*

Declares layout type for the document. See the *LAYOUT element definition* in SMIL 1.0.
IDL Definition

```
interface SMILLayoutElement : SMILElement {
           attribute DOMString        type;
                                        // raises(DOMException) on setting

};
```

Attributes
`type` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

Interface *SMILRootLayoutElement*

Declares layout properties for the root element. See the *root-layout element definition* in SMIL 1.0.
IDL Definition

```
interface SMILRootLayoutElement : SMILElement {
           attribute DOMString        title;
                                        // raises(DOMException) on setting

           attribute DOMString        skipContent;
                                        // raises(DOMException) on setting
```

```
            attribute DOMString        backgroundColor;
                                         // raises(DOMException) on setting

            attribute long             height;
                                         // raises(DOMException) on setting

            attribute long             width;
                                         // raises(DOMException) on setting

    };
```

Attributes

`title` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`skipContent` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`backgroundColor` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`height` of type `long`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`width` of type `long`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

Interface *SMILRegionElement*

Controls the position, size and scaling of media object elements. See the *region element definition* in SMIL 1.0.

IDL Definition

```
interface SMILRegionElement : SMILElement {
           attribute DOMString       title;
                                       // raises(DOMException) on setting

           attribute DOMString       skipContent;
                                       // raises(DOMException) on setting

           attribute DOMString       fit;
                                       // raises(DOMException) on setting

           attribute DOMString       backgroundColor;
                                       // raises(DOMException) on setting

           attribute long            height;
                                       // raises(DOMException) on setting

           attribute long            width;
                                       // raises(DOMException) on setting

           attribute DOMString       top;
                                       // raises(DOMException) on setting

           attribute long            zIndex;
                                       // raises(DOMException) on setting

};
```

Attributes

`title` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`skipContent` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`fit` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`backgroundColor` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`height` of type `long`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`width` of type `long`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`top` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`zIndex` of type `long`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

The layout module also includes the region attribute, used in SMIL layout to associate layout with content elements. This is represented as an individual interface, that is supported by content elements in SMIL documents (i.e. in profiles that use SMIL layout).

Interface *SMILRegionInterface*

Declares rendering surface for an element. See the *region attribute definition* in SMIL 1.0.
IDL Definition

```
interface SMILRegionInterface {
           attribute SMILRegionElement  region;
};
```

Attributes

`region` of type `SMILRegionElement` [p.20]

5.5. Timing and synchronization extensions

This module includes extensions for timing and synchronization.
(***ED:*** This will be fleshed out as we work on the timing module. For now, we will define a time leaf interface as a placeholder for simple media elements (i.e. those that are not also time containers). This is just an indication of one possibility - this is subject to discussion and review.)

Interface *Time*

The `Time` interface is a datatype that represents times within the timegraph. A `Time` has a type, key values to describe the time, and a boolean to indicate whether the values are currently unresolved. Note that if the boolean value "resolved" is `false`, then the time must be considered to be "indefinite" (although the other values are still set to describe the `Time`).

IDL Definition

```
interface Time {
  readonly attribute boolean          resolved;
  readonly attribute double           resolvedOffset;
  // TimeTypes
  const unsigned short      SMIL_TIME_INDEFINITE           = 0;
  const unsigned short      SMIL_TIME_OFFSET               = 1;
  const unsigned short      SMIL_TIME_SYNC_BASED           = 2;
  const unsigned short      SMIL_TIME_EVENT_BASED          = 3;
  const unsigned short      SMIL_TIME_WALLCLOCK            = 4;
  const unsigned short      SMIL_TIME_MEDIA_MARKER         = 5;

  readonly attribute unsigned short   timeType;
           attribute double           offset;
                                        // raises(DOMException) on setting

           attribute Element          baseElement;
                                        // raises(DOMException) on setting

           attribute boolean          baseBegin;
                                        // raises(DOMException) on setting

           attribute DOMString        event;
                                        // raises(DOMException) on setting

           attribute DOMString        marker;
                                        // raises(DOMException) on setting

};
```

Attributes

`resolved` of type `boolean`, readonly

A boolean indicating whether the current `Time` has been fully resolved to the document schedule. Note that for this to be true, the current `Time` must be defined (not indefinite), the syncbase and all `Time`'s that the syncbase depends on must be defined (not indefinite),

and the begin `Time` of all ascendent time containers of this element and all `Time` elements that this depends upon must be defined (not indefinite).
If this `Time` is based upon an event, this `Time` will only be resolved once the specified event has happened, subject to the constraints of the time container.
Note that this may change from true to false when the parent time container ends its simple duration (including when it repeats or restarts).

`resolvedOffset` of type `double`, readonly
The clock value in seconds relative to the parent time container begin. This indicates the resolved time relationship to the parent time container. This is only valid if resolved is true.

Definition group *TimeTypes*

An integer indicating the type of this time value.
Defined Constants

SMIL_TIME_INDEFINITE	The `Time` is specified to be "indefinite". The `Time` may have a resolved value, if a method call is made to activate this time (beginElement for a begin time, or endElement for an active end time). If the `Time` is resolved, the resolvedOffset will reflect the time at which the method call was made. The `Time` attributes offset, baseBegin, event and marker are undefined for this type.
SMIL_TIME_OFFSET	The value is a simple offset from the default syncbase. The baseElement will be the default syncbase element defined by the time model, the baseBegin indicates whether the default syncbase is relative to the begin or active end of the baseElement, and the offset will be the specified offset. The `Time` attributes baseElement, baseBegin, event and marker are undefined for this type.

SMIL_TIME_SYNC_BASED	The value is relative to the begin of the specified baseElement. The baseElement will be the specified syncbase, the baseBegin indicates whether the default syncbase is relative to the begin or active end of the baseElement, and the offset will be the specified offset. Note that this includes times specified with the logical syncbases "prev.begin" or "prev.end". The `Time` attributes event and marker are undefined for this type.
SMIL_TIME_EVENT_BASED	The value is relative to the specified event raised on the baseElement. The baseElement will be the specified syncbase, and the offset will be the specified offset. If the `Time` is resolved, the resolvedOffset will reflect the time at which the event was raised. The `Time` attributes baseBegin and marker are undefined for this type.
SMIL_TIME_WALLCLOCK	The value is specified as a wallclock value. If the `Time` is resolved, the resolvedOffset will reflect the wallclock time, converted to document time. The `Time` attributes baseElement, baseBegin, event and marker are undefined for this type.
SMIL_TIME_MEDIA_MARKER	The value is specified as a marker value associated with a given media element. The baseElement will be the specified media element, and the marker will be the name of the media marker value. If the `Time` is resolved, the resolvedOffset will reflect the time associated with the specified marker value. The `Time` attributes offset, baseElement and event are undefined for this type.

`timeType` of type `unsigned short`, readonly
A code representing the type of the underlying object, as defined above.

`offset` of type `double`
The clock value in seconds relative to the syncbase or eventbase. Default value is `0`.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised on attempts to modify this readonly attribute.

`baseElement` of type `Element`
The base element for a sync-based or event-based time.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised on attempts to modify this readonly attribute.

`baseBegin` of type `boolean`
If `true`, indicates that a sync-based time is relative to the begin of the baseElement. If `false`, indicates that a sync-based time is relative to the active end of the baseElement.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised on attempts to modify this readonly attribute.

`event` of type `DOMString`
The name of the event for an event-based time. Default value is `null`.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised on attempts to modify this readonly attribute.

`marker` of type `DOMString`
The name of the marker from the media element, for media marker times. Default value is `null`.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised on attempts to modify this readonly attribute.

Interface *TimeList*

The `TimeList` interface provides the abstraction of an ordered collection of times, without defining or constraining how this collection is implemented.

The items in the `TimeList` are accessible via an integral index, starting from 0.
IDL Definition

```
interface TimeList {
  Time               item(in unsigned long index);
  readonly attribute unsigned long    length;
};
```

Attributes

`length` of type `unsigned long`, readonly

The number of times in the list. The range of valid child time indices is 0 to `length-1` inclusive.

Methods

`item`

Returns the `index`th item in the collection. If `index` is greater than or equal to the number of times in the list, this returns `null`.

Parameters

`unsigned long`	`index`	Index into the collection.

Return Value

`Time` [p.23]	The time at the `index`th position in the `TimeList`, or `null` if that is not a valid index.

No Exceptions

Interface *ElementTime*

This interface defines the set of timing attributes that are common to all timed elements.

IDL Definition

```
interface ElementTime {
           attribute TimeList         begin;
                                        // raises(DOMException) on setting

           attribute TimeList         end;
                                        // raises(DOMException) on setting

           attribute float            dur;
                                        // raises(DOMException) on setting

           attribute float            repeatCount;
                                        // raises(DOMException) on setting

           attribute long             repeatDur;
                                        // raises(DOMException) on setting

  boolean            beginElement();
  boolean            endElement();
  void               pauseElement();
  void               resumeElement();
  void               seekElement(inout DOMString seekTo);
};
```

Attributes

`begin` of type `TimeList` [p.26]

The desired value (as a list of times) of the *begin* instant of this node.

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`end` of type `TimeList` [p.26]

The desired value (as a list of times) of the *end* instant of this node.

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`dur` of type `float`

The desired simple duration value of this node in seconds. Negative value means "indefinite".

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`repeatCount` of type `float`

Causes the element to play repeatedly (loop) for the specified number of times. A negative value repeat the element indefinitely. Default value is 0 (undefined).

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`repeatDur` of type `long`

Causes the element to play repeatedly (loop) for the specified duration in milliseconds. Negative means "indefinite".

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

Methods

`beginElement`

Causes this element to begin the local timeline (subject to sync constraints).

Return Value

`boolean`	`true` if the method call was successful and the element was begun. `false` if the method call failed. Possible reasons for failure include: • The element doesn't support the `beginElement` method. (the `beginEvent` attribute is not set to `"none"`) • The element is already active and can't be restart when it is active. (the `restart` attribute is set to `"whenNotActive"`) • The element is active or has been active and can't be restart. (the `restart` attribute is set to `"never"`).

No Parameters
No Exceptions

`endElement`

Causes this element to end the local timeline (subject to sync constraints).
Return Value

`boolean`	`true` if the method call was successful and the element was endeed. `false` if method call failed. Possible reasons for failure include: • The element doesn't support the `endElement` method. (the `endEvent` attribute is not set to `"none"`) • The element is not active.

No Parameters
No Exceptions

`pauseElement`

Causes this element to pause the local timeline (subject to sync constraints).
No Parameters
No Return Value
No Exceptions

`resumeElement`

Causes this element to resume a paused local timeline.
No Parameters
No Return Value
No Exceptions

`seekElement`

Seeks this element to the specified point on the local timeline (subject to sync constraints). If this is a timeline, this must seek the entire timeline (i.e. propagate to all timeChildren).
Parameters

`DOMString`	`seekTo`	The desired position on the local timeline.

No Return Value
No Exceptions

Events:

begin
: This event is raised when the element local timeline begins to play. It will be raised each time the element begins the active duration (i.e. not on repeats - see the repeat Event). It may be raised both in the course of normal (i.e. scheduled or interactive) timeline play, as well as in the case that the element was begun with the beginElement() method. Note that if an element is restarted while it is currently playing, the element will raise an end event and another begin event, as the element restarts. Note that if an element is not yet ready to play (e.g. if media is not ready), the begin event should not be raised until the element timeline actually begins to play and local time begins to advance.
As a composite timeline begins to play, each element will raise an begin event as it in turn begins to play. A parent element will raise an begin event before any child elements do.

end
: This event is raised at the active end of the element. Note that this event is not raised at the simple end of each repeat - see the repeat event. This event may be raised both in the course of normal (i.e. scheduled or interactive) timeline play, as well as in the case that the element was ended with the endElement() method. As a time container reaches its simple and/or active end, child elements will raise an end event if the time container "cuts short" its active duration. Note that if an element is restarted while it is currently playing, the element will raise an end event and another begin event, as the element restarts.

repeat
: This event is raised when the element local timeline repeats. It will be raised each time the element repeats, after the first iteration.
(***ED:*** This event should support an integer attribute to indicate the current repeat iteration.)

pause
: This event is raised when the element local timeline is paused. This is only raised when the element pauseElement() method is invoked.
(***ED:*** When pausing a timeline, I do not think that all descendents should also raise pause events. However, it may be useful to have media descendents raise a pause event. This needs attention.
I think we should consider supporting a "reason" attribute on this event. This would allow authors to disambiguate a pause due to a method call, and a pause forced by the timing engine as part of handling an out-of-sync problem.)

resume
: This event is raised when the element local timeline resumes after being paused. This is only raised when the element `resumeElement` method is invoked, and only if the element was actually paused.
(***ED:*** When resuming a timeline, I do not think that all descendents should also raise resume events. However, it may be useful to have media descendents raise a resume event. This needs attention.)

outOfSync
: This event is raised when an element timeline falls out of sync (either for internal or external reasons). The default action of the timing model is to attempt to reestablish the synchronization, however the means may be implementation dependent. Depending upon the synchronization rules, this event may propagate up the time graph for each timeline that is affected.

syncRestored

This event is raised when an element that has fallen out of sync has been restored to the proper sync relationship with the parent timeline. This will only be raise after an outOfSync event has been raised. Depending upon the synchronization rules, this event may propagate down the time graph, effectively "unwinding" the original outOfSync stack.

Interface *ElementTimeManipulation*

This interface support use-cases commonly associated with animation.

IDL Definition

```
interface ElementTimeManipulation {
           attribute float           speed;
                                       // raises(DOMException) on setting

           attribute float           accelerate;
                                       // raises(DOMException) on setting

           attribute float           decelerate;
                                       // raises(DOMException) on setting

           attribute boolean         autoReverse;
                                       // raises(DOMException) on setting

};
```

Attributes

`speed` of type `float`

Defines the playback speed of element time. The value is specified as a multiple of normal (parent time container) play speed. Legal values are signed floating point values. Zero values are not allowed. The default is `1.0` (no modification of speed).

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`accelerate` of type `float`

The percentage value of the simple acceleration of time for the element. Allowed values are from `0` to `100`. Default value is `0` (no acceleration).
The sum of the values for accelerate and decelerate must not exceed 100. If it does, the deceleration value will be reduced to make the sum legal.

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`decelerate` of type `float`

The percentage value of the simple decelerate of time for the element. Allowed values are from `0` to `100`. Default value is `0` (no deceleration).

The sum of the values for accelerate and decelerate must not exceed 100. If it does, the deceleration value will be reduced to make the sum legal.

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`autoReverse` of type `boolean`

The "play forwards then backwards" functionality. Default value is `false`.

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

Interface *ElementTimeSynchronization*

The synchronization behavior extension.

IDL Definition

```
interface ElementTimeSynchronization {
  readonly attribute DOMString        syncBehavior;
  readonly attribute float            syncTolerance;
  readonly attribute DOMString        defaultSyncBehavior;
  readonly attribute float            defaultSyncTolerance;
  readonly attribute boolean          syncMaster;
};
```

Attributes

`syncBehavior` of type `DOMString`, readonly

The runtime synchronization behavior for an element.

`syncTolerance` of type `float`, readonly

The sync tolerance for the associated element. It has an effect only if the element has `syncBehavior="locked"`.

`defaultSyncBehavior` of type `DOMString`, readonly

Defines the default value for the runtime synchronization behavior for an element, and all descendents.

`defaultSyncTolerance` of type `float`, readonly

Defines the default value for the sync tolerance for an element, and all descendents.

`syncMaster` of type `boolean`, readonly

If set to true, forces the time container playback to sync to this element.

Interface *ElementTimeContainer*

This is a placeholder - subject to change. This represents generic timelines.

IDL Definition

```
interface ElementTimeContainer : ElementTime {
  readonly attribute NodeList         timeChildrens;
  NodeList           getActiveChildrenAt(inout DOMString instant);
};
```

Attributes

`timeChildrens` of type `NodeList`, readonly

A NodeList that contains all timed childrens of this node. If there are no timed children, the `Nodelist` is empty.

Note: An iterator is more appropriate here than a node list but it requires Traversal module support.

Methods

`getActiveChildrenAt`

Returns a list of child elements active at the time of invocation.

Parameters

`DOMString`	`instant`	The desired position on the local timeline.

Return Value

`NodeList`	List of timed child-elements active at instant.

No Exceptions

Interface *ElementParallelTimeContainer*

A `parallel` container defines a simple parallel time grouping in which multiple elements can play back at the same time.

IDL Definition

```
interface ElementParallelTimeContainer : ElementTimeContainer {
           attribute DOMString        endSync;
                                        // raises(DOMException) on setting

};
```

Attributes

`endSync` of type `DOMString`

Controls the end of the container.

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

Interface *ElementSequentialTimeContainer*

A `seq` container defines a sequence of elements in which elements play one after the other.
IDL Definition

```
interface ElementSequentialTimeContainer : ElementTimeContainer {
};
```

Interface *ElementExclusiveTimeContainer*

This interface defines a time container with semantics based upon par, but with the additional constraint that only one child element may play at a time.
IDL Definition

```
interface ElementExclusiveTimeContainer : ElementTimeContainer {
           attribute DOMString        endSync;
                                        // raises(DOMException) on setting

};
```

Attributes
`endSync` of type `DOMString`
Controls the end of the container.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

5.6. Media Object extensions

This module includes the media elements, and associated attributes. They are all currently represented by a single interface, as there are no specific attributes for individual media elements.
(***ED:*** This extensions are based on SMIL 1.0 and will change in future versions.)

Interface *SMILMediaElement*

Declares media content.
IDL Definition

```
interface SMILMediaElement : ElementTime, SMILElement {
           attribute DOMString        abstractAttr;
                                        // raises(DOMException) on setting

           attribute DOMString        alt;
                                        // raises(DOMException) on setting
```

```
            attribute DOMString        author;
                                         // raises(DOMException) on setting

            attribute DOMString        clipBegin;
                                         // raises(DOMException) on setting

            attribute DOMString        clipEnd;
                                         // raises(DOMException) on setting

            attribute DOMString        copyright;
                                         // raises(DOMException) on setting

            attribute DOMString        longdesc;
                                         // raises(DOMException) on setting

            attribute DOMString        src;
                                         // raises(DOMException) on setting

            attribute DOMString        title;
                                         // raises(DOMException) on setting

            attribute DOMString        type;
                                         // raises(DOMException) on setting

};
```

Attributes

`abstractAttr` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`alt` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`author` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`clipBegin` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`clipEnd` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`copyright` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`longdesc` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`src` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`title` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`type` of type `DOMString`
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

Interface *SMILRefElement*
(***ED:*** // audio, video, ...)

IDL Definition

```
interface SMILRefElement : SMILMediaElement {
};
```

5.7. Animations extensions

This module will include interfaces associated with animations extensions.

5.7.1. SMIL Animation

The following interface comes from the *SMIL Animation* draft.

Interface *ElementTimeControl*

IDL Definition

```
interface ElementTimeControl {
  boolean            beginElement()
                                       raises(DOMException);
  boolean            endElement()
                                       raises(DOMException);
};
```

Methods

`beginElement`

Causes this element to begin the local timeline (subject to sync constraints).

Return Value

`boolean`	`true` if the method call was successful and the element was begun. `false` if the method call failed. Possible reasons for failure include: ● The element doesn't support the `beginElement` method. (the `beginEvent` attribute is not set to `"none"`) ● The element is already active and can't be restart when it is active. (the `restart` attribute is set to `"whenNotActive"`) ● The element is active or has been active and can't be restart. (the `restart` attribute is set to `"never"`).

Exceptions

`DOMException`	SYNTAX_ERR: The element was not defined with the appropriate syntax to allow `beginElement` calls.

No Parameters

`endElement`

Causes this element to end the local timeline (subject to sync constraints).

Return Value

`boolean`	`true` if the method call was successful and the element was ended. `false` if method call failed. Possible reasons for failure include: • The element doesn't support the `endElement` method. (the `endEvent` attribute is not set to `"none"`) • The element is not active.

Exceptions

`DOMException`	SYNTAX_ERR: The element was not defined with the appropriate syntax to allow `endElement` calls.

No Parameters

5.7.2. SMIL Boston Animation

Interface *ElementAnimation*

This interface define the set of animation extensions.
(***ED:*** The [XLink] attributes will go in a XLink interface.)

IDL Definition

```
interface ElementAnimation : ElementTime, ElementTimeControl {
           attribute Element          targetElement;
                                        // raises(DOMException) on setting

           attribute DOMString        href;
                                        // raises(DOMException) on setting

};
```

Attributes

`targetElement` of type `Element`

This attribute specifies the target element to be animated. If no `href` and `targetElement` are specified in the animation document, the default value of this attribute is the first element ancestor. If a `href` is not `null`, setting this attribute has no effect.

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`href` of type `DOMString`
This attribute specifies an [XLink] reference to the target element to be animated.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

Interface *SMILAnimateElement*

This interface represents the SMIL `animate` element, the SMIL `animateColor` element and the SMIL `animateMotion` element.
IDL Definition

```
interface SMILAnimateElement : ElementAnimation, SMILElement {
           attribute TimeList         keyTimes;
                                        // raises(DOMException) on setting

           attribute TimeList         keySplines;
                                        // raises(DOMException) on setting

};
```

Attributes
`keyTimes` of type `TimeList` [p.26]
A semicolon-separated list of time values used to control the pacing of the animation.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`keySplines` of type `TimeList` [p.26]
A set of Bezier control points associated with the keyTimes list.
Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

Interface *SMILSetElement*

This interface represents the `set` element.
IDL Definition

```
interface SMILSetElement : ElementAnimation, SMILElement {
};
```

Interface *SMILAnimateMotionElement*

This interface present the `animationMotion` element in SMIL.
IDL Definition

```
interface SMILAnimateMotionElement : SMILAnimateElement {
           attribute DOMString        path;
                                        // raises(DOMException) on setting

           attribute DOMString        origin;
                                        // raises(DOMException) on setting

};
```

Attributes

`path` of type `DOMString`

Specifies the curve that describes the attribute value as a function of time.

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`origin` of type `DOMString`

Specifies the origin of motion for the animation.

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

5.8. Transition extensions

This module will include interfaces associated with transition markup. This is yet to be defined.

5.9. Linking extensions

This module includes interfaces for hyperlinking elements.

5.10. Content Control extensions

This module includes interfaces for content control markup.
(***ED:*** This extensions are based on SMIL 1.0 and will change in future versions.)

Interface *SMILSwitchElement*

Defines a block of content control. See the *switch element definition* in SMIL 1.0.
IDL Definition

```
interface SMILSwitchElement : SMILElement {
           attribute DOMString        title;
                                        // raises(DOMException) on setting

};
```

Attributes

`title` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

Interface *ElementTest*

Defines the test attributes interface. See the *Test attributes definition* in SMIL 1.0.

IDL Definition

```
interface ElementTest {
           attribute DOMString        systemBitrate;
                                        // raises(DOMException) on setting

           attribute DOMString        systemCaptions;
                                        // raises(DOMException) on setting

           attribute DOMString        systemLanguage;
                                        // raises(DOMException) on setting

           attribute DOMString        systemOverdubOrCaption;
                                        // raises(DOMException) on setting

           attribute DOMString        systemRequired;
                                        // raises(DOMException) on setting

           attribute DOMString        systemScreenSize;
                                        // raises(DOMException) on setting

           attribute DOMString        systemScreenDepth;
                                        // raises(DOMException) on setting

};
```

Attributes

`systemBitrate` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`systemCaptions` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`systemLanguage` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`systemOverdubOrCaption` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`systemRequired` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`systemScreenSize` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

`systemScreenDepth` of type `DOMString`

Exceptions on setting

`DOMException`	NO_MODIFICATION_ALLOWED_ERR: Raised if this attribute is readonly.

5.11. Media Player extensions

(***ED:*** This is NOT a plug-in interface, but rather a simple interface that describes some guaranteed methods that any application plug-in interface must support. This provides a means of standardizing extensions to the timing model, independent of the specific application.)

5.11.1. Media Player Level 1 Interface

5.11.2. Media Player Level 2 Interface

5.11.3. Media Player Level 3 Interface

5.11.3. Media Player Level 3 Interface

Appendix A: IDL Definitions

This appendix contains the complete OMG IDL for the SYMM Object Model definitions. The definitions are divided into SYMM [p.45] .

The IDL files are also available as: http://www.w3.org/TR/1999/WD-smil-boston-dom-19991115/idl.zip

A.1: SYMM Document Object Model

smil.idl:

```
// File: smil.idl
#ifndef _SMIL_IDL_
#define _SMIL_IDL_

#include "dom.idl"

#pragma prefix "dom.w3c.org"
module smil
{
  typedef dom::Element Element;
  typedef dom::DOMString DOMString;
  typedef dom::NodeList NodeList;
  typedef dom::Document Document;

  interface SMILRegionElement;

  interface SMILRegionInterface {
            attribute SMILRegionElement  region;
  };

  interface Time {
    readonly attribute boolean          resolved;
    readonly attribute double           resolvedOffset;
    // TimeTypes
    const unsigned short      SMIL_TIME_INDEFINITE          = 0;
    const unsigned short      SMIL_TIME_OFFSET              = 1;
    const unsigned short      SMIL_TIME_SYNC_BASED          = 2;
    const unsigned short      SMIL_TIME_EVENT_BASED         = 3;
    const unsigned short      SMIL_TIME_WALLCLOCK           = 4;
    const unsigned short      SMIL_TIME_MEDIA_MARKER        = 5;

    readonly attribute unsigned short   timeType;
             attribute double           offset;
                                        // raises(dom::DOMException) on setting

             attribute Element          baseElement;
                                        // raises(dom::DOMException) on setting

             attribute boolean          baseBegin;
                                        // raises(dom::DOMException) on setting

             attribute DOMString        event;
```

```
                                       // raises(dom::DOMException) on setting

           attribute DOMString        marker;
                                       // raises(dom::DOMException) on setting

};

interface TimeList {
  Time               item(in unsigned long index);
  readonly attribute unsigned long    length;
};

interface ElementTime {
           attribute TimeList         begin;
                                       // raises(dom::DOMException) on setting

           attribute TimeList         end;
                                       // raises(dom::DOMException) on setting

           attribute float            dur;
                                       // raises(dom::DOMException) on setting

           attribute float            repeatCount;
                                       // raises(dom::DOMException) on setting

           attribute long             repeatDur;
                                       // raises(dom::DOMException) on setting

  boolean            beginElement();
  boolean            endElement();
  void               pauseElement();
  void               resumeElement();
  void               seekElement(inout DOMString seekTo);
};

interface ElementTimeManipulation {
           attribute float            speed;
                                       // raises(dom::DOMException) on setting

           attribute float            accelerate;
                                       // raises(dom::DOMException) on setting

           attribute float            decelerate;
                                       // raises(dom::DOMException) on setting

           attribute boolean          autoReverse;
                                       // raises(dom::DOMException) on setting

};

interface ElementTimeSynchronization {
  readonly attribute DOMString        syncBehavior;
  readonly attribute float            syncTolerance;
  readonly attribute DOMString        defaultSyncBehavior;
  readonly attribute float            defaultSyncTolerance;
  readonly attribute boolean          syncMaster;
};
```

```
interface ElementTimeContainer : ElementTime {
  readonly attribute NodeList         timeChildrens;
  NodeList           getActiveChildrenAt(inout DOMString instant);
};

interface ElementParallelTimeContainer : ElementTimeContainer {
           attribute DOMString        endSync;
                                      // raises(dom::DOMException) on setting

};

interface ElementSequentialTimeContainer : ElementTimeContainer {
};

interface ElementExclusiveTimeContainer : ElementTimeContainer {
           attribute DOMString        endSync;
                                      // raises(dom::DOMException) on setting

};

interface ElementTimeControl {
  boolean            beginElement()
                                      raises(dom::DOMException);
  boolean            endElement()
                                      raises(dom::DOMException);
};

interface ElementAnimation : ElementTime, ElementTimeControl {
           attribute Element          targetElement;
                                      // raises(dom::DOMException) on setting

           attribute DOMString        href;
                                      // raises(dom::DOMException) on setting

};

interface ElementTest {
           attribute DOMString        systemBitrate;
                                      // raises(dom::DOMException) on setting

           attribute DOMString        systemCaptions;
                                      // raises(dom::DOMException) on setting

           attribute DOMString        systemLanguage;
                                      // raises(dom::DOMException) on setting

           attribute DOMString        systemOverdubOrCaption;
                                      // raises(dom::DOMException) on setting

           attribute DOMString        systemRequired;
                                      // raises(dom::DOMException) on setting

           attribute DOMString        systemScreenSize;
                                      // raises(dom::DOMException) on setting

           attribute DOMString        systemScreenDepth;
```

```
                                    // raises(dom::DOMException) on setting

    };

    interface SMILDocument : Document {
    };

    interface SMILElement : Element {
              attribute DOMString        id;
                                    // raises(dom::DOMException) on setting

    };

    interface SMILMetaElement : SMILElement {
              attribute DOMString        content;
                                    // raises(dom::DOMException) on setting

              attribute DOMString        name;
                                    // raises(dom::DOMException) on setting

              attribute DOMString        skipContent;
                                    // raises(dom::DOMException) on setting

    };

    interface SMILLayoutElement : SMILElement {
              attribute DOMString        type;
                                    // raises(dom::DOMException) on setting

    };

    interface SMILRootLayoutElement : SMILElement {
              attribute DOMString        title;
                                    // raises(dom::DOMException) on setting

              attribute DOMString        skipContent;
                                    // raises(dom::DOMException) on setting

              attribute DOMString        backgroundColor;
                                    // raises(dom::DOMException) on setting

              attribute long             height;
                                    // raises(dom::DOMException) on setting

              attribute long             width;
                                    // raises(dom::DOMException) on setting

    };

    interface SMILRegionElement : SMILElement {
              attribute DOMString        title;
                                    // raises(dom::DOMException) on setting

              attribute DOMString        skipContent;
                                    // raises(dom::DOMException) on setting

              attribute DOMString        fit;
```

```
                                    // raises(dom::DOMException) on setting

          attribute DOMString        backgroundColor;
                                    // raises(dom::DOMException) on setting

          attribute long             height;
                                    // raises(dom::DOMException) on setting

          attribute long             width;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        top;
                                    // raises(dom::DOMException) on setting

          attribute long             zIndex;
                                    // raises(dom::DOMException) on setting

};

interface SMILMediaElement : ElementTime, SMILElement {
          attribute DOMString        abstractAttr;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        alt;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        author;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        clipBegin;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        clipEnd;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        copyright;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        longdesc;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        src;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        title;
                                    // raises(dom::DOMException) on setting

          attribute DOMString        type;
                                    // raises(dom::DOMException) on setting

};

interface SMILRefElement : SMILMediaElement {
};

interface SMILAnimateElement : ElementAnimation, SMILElement {
          attribute TimeList         keyTimes;
```

```
                                        // raises(dom::DOMException) on setting

           attribute TimeList          keySplines;
                                        // raises(dom::DOMException) on setting

  };

  interface SMILSetElement : ElementAnimation, SMILElement {
  };

  interface SMILAnimateMotionElement : SMILAnimateElement {
           attribute DOMString         path;
                                        // raises(dom::DOMException) on setting

           attribute DOMString         origin;
                                        // raises(dom::DOMException) on setting

  };

  interface SMILSwitchElement : SMILElement {
           attribute DOMString         title;
                                        // raises(dom::DOMException) on setting

  };
};

#endif // _SMIL_IDL_
```

Appendix B: Java Language Binding

This appendix contains the complete Java bindings for the SYMM Object Model. The definitions are divided into SMIL [p.51] .

The Java files are also available as
http://www.w3.org/TR/1999/WD-smil-boston-dom-19991115/java-binding.zip

B.1: SMIL Document Object Model

org/w3c/dom/smil/SMILDocument.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.Document;

public interface SMILDocument extends Document {
}
```

org/w3c/dom/smil/SMILElement.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;
import org.w3c.dom.Element;

public interface SMILElement extends Element {
  public String             getId();
  public void               setId(String id)
 throws DOMException;
}
```

org/w3c/dom/smil/SMILMetaElement.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface SMILMetaElement extends SMILElement {
  public String             getContent();
  public void               setContent(String content)
 throws DOMException;
  public String             getName();
  public void               setName(String name)
 throws DOMException;
  public String             getSkipContent();
  public void               setSkipContent(String skipContent)
 throws DOMException;
}
```

org/w3c/dom/smil/SMILLayoutElement.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface SMILLayoutElement extends SMILElement {
  public String             getType();
  public void               setType(String type)
 throws DOMException;
}
```

org/w3c/dom/smil/SMILRootLayoutElement.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface SMILRootLayoutElement extends SMILElement {
  public String             getTitle();
  public void               setTitle(String title)
 throws DOMException;
  public String             getSkipContent();
  public void               setSkipContent(String skipContent)
 throws DOMException;
  public String             getBackgroundColor();
  public void               setBackgroundColor(String backgroundColor)
 throws DOMException;
  public int                getHeight();
  public void               setHeight(int height)
 throws DOMException;
  public int                getWidth();
  public void               setWidth(int width)
 throws DOMException;
}
```

org/w3c/dom/smil/SMILRegionElement.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface SMILRegionElement extends SMILElement {
  public String             getTitle();
  public void               setTitle(String title)
 throws DOMException;
  public String             getSkipContent();
  public void               setSkipContent(String skipContent)
 throws DOMException;
  public String             getFit();
  public void               setFit(String fit)
 throws DOMException;
  public String             getBackgroundColor();
  public void               setBackgroundColor(String backgroundColor)
 throws DOMException;
```

```
  public int                getHeight();
  public void               setHeight(int height)
 throws DOMException;
  public int                getWidth();
  public void               setWidth(int width)
 throws DOMException;
  public String             getTop();
  public void               setTop(String top)
 throws DOMException;
  public int                getZIndex();
  public void               setZIndex(int zIndex)
 throws DOMException;
}
```

org/w3c/dom/smil/SMILRegionInterface.java:

```
package org.w3c.dom.smil;

public interface SMILRegionInterface {
  public SMILRegionElement  getRegion();
  public void               setRegion(SMILRegionElement region);
}
```

org/w3c/dom/smil/Time.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;
import org.w3c.dom.Element;

public interface Time {
  public boolean            getResolved();
  public double             getResolvedOffset();
  // TimeTypes
  public static final short           SMIL_TIME_INDEFINITE = 0;
  public static final short           SMIL_TIME_OFFSET     = 1;
  public static final short           SMIL_TIME_SYNC_BASED = 2;
  public static final short           SMIL_TIME_EVENT_BASED = 3;
  public static final short           SMIL_TIME_WALLCLOCK  = 4;
  public static final short           SMIL_TIME_MEDIA_MARKER = 5;

  public short              getTimeType();
  public double             getOffset();
  public void               setOffset(double offset)
 throws DOMException;
  public Element            getBaseElement();
  public void               setBaseElement(Element baseElement)
 throws DOMException;
  public boolean            getBaseBegin();
  public void               setBaseBegin(boolean baseBegin)
 throws DOMException;
  public String             getEvent();
  public void               setEvent(String event)
 throws DOMException;
```

```
  public String              getMarker();
  public void                setMarker(String marker)
 throws DOMException;
}
```

org/w3c/dom/smil/TimeList.java:

```
package org.w3c.dom.smil;

public interface TimeList {
  public Time                item(int index);
  public int                 getLength();
}
```

org/w3c/dom/smil/ElementTime.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface ElementTime {
  public TimeList            getBegin();
  public void                setBegin(TimeList begin)
                                  throws DOMException;
  public TimeList            getEnd();
  public void                setEnd(TimeList end)
                                  throws DOMException;
  public float               getDur();
  public void                setDur(float dur)
                                  throws DOMException;
  public float               getRepeatCount();
  public void                setRepeatCount(float repeatCount)
                                  throws DOMException;
  public int                 getRepeatDur();
  public void                setRepeatDur(int repeatDur)
                                  throws DOMException;
  public boolean             beginElement();
  public boolean             endElement();
  public void                pauseElement();
  public void                resumeElement();
  public void                seekElement(String seekTo);
}
```

org/w3c/dom/smil/ElementTimeManipulation.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface ElementTimeManipulation {
  public float               getSpeed();
  public void                setSpeed(float speed)
                                         throws DOMException;
  public float               getAccelerate();
  public void                setAccelerate(float accelerate)
```

```
                                        throws DOMException;
  public float              getDecelerate();
  public void               setDecelerate(float decelerate)
                                        throws DOMException;
  public boolean            getAutoReverse();
  public void               setAutoReverse(boolean autoReverse)
                                        throws DOMException;
}
```

org/w3c/dom/smil/ElementTimeSynchronization.java:

```
package org.w3c.dom.smil;

public interface ElementTimeSynchronization {
  public String             getSyncBehavior();
  public float              getSyncTolerance();
  public String             getDefaultSyncBehavior();
  public float              getDefaultSyncTolerance();
  public boolean            getSyncMaster();
}
```

org/w3c/dom/smil/ElementTimeContainer.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.NodeList;

public interface ElementTimeContainer extends ElementTime {
  public NodeList           getTimeChildrens();
  public NodeList           getActiveChildrenAt(String instant);
}
```

org/w3c/dom/smil/ElementParallelTimeContainer.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface ElementParallelTimeContainer extends ElementTimeContainer {
  public String             getEndSync();
  public void               setEndSync(String endSync)
                                                throws DOMException;
}
```

org/w3c/dom/smil/ElementSequentialTimeContainer.java:

```
package org.w3c.dom.smil;

public interface ElementSequentialTimeContainer extends ElementTimeContainer {
}
```

org/w3c/dom/smil/ElementExclusiveTimeContainer.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface ElementExclusiveTimeContainer extends ElementTimeContainer {
  public String             getEndSync();
  public void               setEndSync(String endSync)
                                        throws DOMException;
}
```

org/w3c/dom/smil/SMILMediaElement.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface SMILMediaElement extends ElementTime, SMILElement {
  public String             getAbstractAttr();
  public void               setAbstractAttr(String abstractAttr)
                                        throws DOMException;
  public String             getAlt();
  public void               setAlt(String alt)
                                        throws DOMException;
  public String             getAuthor();
  public void               setAuthor(String author)
                                        throws DOMException;
  public String             getClipBegin();
  public void               setClipBegin(String clipBegin)
                                        throws DOMException;
  public String             getClipEnd();
  public void               setClipEnd(String clipEnd)
                                        throws DOMException;
  public String             getCopyright();
  public void               setCopyright(String copyright)
                                        throws DOMException;
  public String             getLongdesc();
  public void               setLongdesc(String longdesc)
                                        throws DOMException;
  public String             getSrc();
  public void               setSrc(String src)
                                        throws DOMException;
  public String             getTitle();
  public void               setTitle(String title)
                                        throws DOMException;
  public String             getType();
  public void               setType(String type)
                                        throws DOMException;
}
```

org/w3c/dom/smil/SMILRefElement.java:

```
package org.w3c.dom.smil;

public interface SMILRefElement extends SMILMediaElement {
}
```

org/w3c/dom/smil/ElementTimeControl.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface ElementTimeControl {
  public boolean            beginElement()
                                          throws DOMException;
  public boolean            endElement()
                                        throws DOMException;
}
```

org/w3c/dom/smil/ElementAnimation.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;
import org.w3c.dom.Element;

public interface ElementAnimation extends ElementTime, ElementTimeControl {
  public Element            getTargetElement();
  public void               setTargetElement(Element targetElement)
                                        throws DOMException;
  public String             getHref();
  public void               setHref(String href)
                                        throws DOMException;
}
```

org/w3c/dom/smil/SMILAnimateElement.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface SMILAnimateElement extends ElementAnimation, SMILElement {
  public TimeList           getKeyTimes();
  public void               setKeyTimes(TimeList keyTimes)
                                        throws DOMException;
  public TimeList           getKeySplines();
  public void               setKeySplines(TimeList keySplines)
                                        throws DOMException;
}
```

org/w3c/dom/smil/SMILSetElement.java:

```
package org.w3c.dom.smil;

public interface SMILSetElement extends ElementAnimation, SMILElement {
}
```

org/w3c/dom/smil/SMILAnimateMotionElement.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface SMILAnimateMotionElement extends SMILAnimateElement {
  public String             getPath();
  public void               setPath(String path)
                                      throws DOMException;
  public String             getOrigin();
  public void               setOrigin(String origin)
                                      throws DOMException;
}
```

org/w3c/dom/smil/SMILSwitchElement.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface SMILSwitchElement extends SMILElement {
  public String             getTitle();
  public void               setTitle(String title)
                                      throws DOMException;
}
```

org/w3c/dom/smil/ElementTest.java:

```
package org.w3c.dom.smil;

import org.w3c.dom.DOMException;

public interface ElementTest {
  public String             getSystemBitrate();
  public void               setSystemBitrate(String systemBitrate)
                                      throws DOMException;
  public String             getSystemCaptions();
  public void               setSystemCaptions(String systemCaptions)
                                      throws DOMException;
  public String             getSystemLanguage();
  public void               setSystemLanguage(String systemLanguage)
                                      throws DOMException;
  public String             getSystemOverdubOrCaption();
  public void               setSystemOverdubOrCaption(String systemOverdubOrCaption)
                                      throws DOMException;
  public String             getSystemRequired();
  public void               setSystemRequired(String systemRequired)
```

```
                                           throws DOMException;
    public String               getSystemScreenSize();
    public void                 setSystemScreenSize(String systemScreenSize)
                                           throws DOMException;
    public String               getSystemScreenDepth();
    public void                 setSystemScreenDepth(String systemScreenDepth)
                                           throws DOMException;
  }
```

org/w3c/dom/smil/ElementTest.java:

Appendix C: ECMA Script Language Binding

This appendix contains the complete ECMA Script binding for the SYMM Object Model definitions. The definitions are divided into SMIL [p.61] .

C.1: SMIL Document Object Model

Object **SMILDocument**

SMILDocument has the all the properties and methods of **Document** as well as the properties and methods defined below.

Object **SMILElement**

SMILElement has the all the properties and methods of **Element** as well as the properties and methods defined below.

The **SMILElement** object has the following properties:

id

This property is of type **String**.

Object **SMILMetaElement**

SMILMetaElement has the all the properties and methods of **SMILElement** as well as the properties and methods defined below.

The **SMILMetaElement** object has the following properties:

content

This property is of type **String**.

name

This property is of type **String**.

skipContent

This property is of type **String**.

Object **SMILLayoutElement**

SMILLayoutElement has the all the properties and methods of **SMILElement** as well as the properties and methods defined below.

The **SMILLayoutElement** object has the following properties:

type

This property is of type **String**.

Object **SMILRootLayoutElement**

SMILRootLayoutElement has the all the properties and methods of **SMILElement** as well as the properties and methods defined below.

The **SMILRootLayoutElement** object has the following properties:

title

This property is of type **String**.

skipContent

This property is of type **String**.

backgroundColor

This property is of type **String**.

height

This property is of type **long**.

width
This property is of type **long**.

Object **SMILRegionElement**

SMILRegionElement has the all the properties and methods of **SMILElement** as well as the properties and methods defined below.

The **SMILRegionElement** object has the following properties:

title
This property is of type **String**.

skipContent
This property is of type **String**.

fit
This property is of type **String**.

backgroundColor
This property is of type **String**.

height
This property is of type **long**.

width
This property is of type **long**.

top
This property is of type **String**.

zIndex
This property is of type **long**.

Object **SMILRegionInterface**

The **SMILRegionInterface** object has the following properties:

region
This property is of type **SMILRegionElement**.

Object **Time**

The **Time** object has the following properties:

resolved
This property is of type **boolean**.

resolvedOffset
This property is of type **double**.

timeType
This property is of type **short**.

offset
This property is of type **double**.

baseElement
This property is of type **Element**.

baseBegin
This property is of type **boolean**.

event
This property is of type **String**.

marker
This property is of type **String**.

Object **TimeList**

The **TimeList** object has the following properties:

length

This property is of type **int**.

The **TimeList** object has the following methods:

item(index)

This method returns a **Time**. The **index** parameter is of type **unsigned long**. This object can also be dereferenced using square bracket notation (e.g. obj[1]). Dereferencing with an integer `index` is equivalent to invoking the **item** method with that index.

Object **ElementTime**

The **ElementTime** object has the following properties:

begin

This property is of type **TimeList**.

end

This property is of type **TimeList**.

dur

This property is of type **float**.

repeatCount

This property is of type **float**.

repeatDur

This property is of type **long**.

The **ElementTime** object has the following methods:

beginElement()

This method returns a **boolean**.

endElement()

This method returns a **boolean**.

pauseElement()

This method returns a **void**.

resumeElement()

This method returns a **void**.

seekElement(seekTo)

This method returns a **void**. The **seekTo** parameter is of type **DOMString**.

Object **ElementTimeManipulation**

The **ElementTimeManipulation** object has the following properties:

speed

This property is of type **float**.

accelerate

This property is of type **float**.

decelerate

This property is of type **float**.

autoReverse

This property is of type **boolean**.

Object **ElementTimeSynchronization**

The **ElementTimeSynchronization** object has the following properties:

syncBehavior

This property is of type **String**.

syncTolerance
This property is of type **float**.
defaultSyncBehavior
This property is of type **String**.
defaultSyncTolerance
This property is of type **float**.
syncMaster
This property is of type **boolean**.

Object **ElementTimeContainer**
ElementTimeContainer has the all the properties and methods of **ElementTime** as well as the properties and methods defined below.
The **ElementTimeContainer** object has the following properties:
timeChildrens
This property is of type **NodeList**.
The **ElementTimeContainer** object has the following methods:
getActiveChildrenAt(instant)
This method returns a **NodeList**. The **instant** parameter is of type **DOMString**.

Object **ElementParallelTimeContainer**
ElementParallelTimeContainer has the all the properties and methods of **ElementTimeContainer** as well as the properties and methods defined below.
The **ElementParallelTimeContainer** object has the following properties:
endSync
This property is of type **String**.

Object **ElementSequentialTimeContainer**
ElementSequentialTimeContainer has the all the properties and methods of **ElementTimeContainer** as well as the properties and methods defined below.

Object **ElementExclusiveTimeContainer**
ElementExclusiveTimeContainer has the all the properties and methods of **ElementTimeContainer** as well as the properties and methods defined below.
The **ElementExclusiveTimeContainer** object has the following properties:
endSync
This property is of type **String**.

Object **SMILMediaElement**
SMILMediaElement has the all the properties and methods of **ElementTime SMILElement** as well as the properties and methods defined below.
The **SMILMediaElement** object has the following properties:
abstractAttr
This property is of type **String**.
alt
This property is of type **String**.
author
This property is of type **String**.
clipBegin
This property is of type **String**.
clipEnd
This property is of type **String**.

copyright
This property is of type **String**.
longdesc
This property is of type **String**.
src
This property is of type **String**.
title
This property is of type **String**.
type
This property is of type **String**.

Object **SMILRefElement**
SMILRefElement has the all the properties and methods of **SMILMediaElement** as well as the properties and methods defined below.

Object **ElementTimeControl**
The **ElementTimeControl** object has the following methods:
beginElement()
This method returns a **boolean**.
endElement()
This method returns a **boolean**.

Object **ElementAnimation**
ElementAnimation has the all the properties and methods of **ElementTime ElementTimeControl** as well as the properties and methods defined below.
The **ElementAnimation** object has the following properties:
targetElement
This property is of type **Element**.
href
This property is of type **String**.

Object **SMILAnimateElement**
SMILAnimateElement has the all the properties and methods of **ElementAnimation SMILElement** as well as the properties and methods defined below.
The **SMILAnimateElement** object has the following properties:
keyTimes
This property is of type **TimeList**.
keySplines
This property is of type **TimeList**.

Object **SMILSetElement**
SMILSetElement has the all the properties and methods of **ElementAnimation SMILElement** as well as the properties and methods defined below.

Object **SMILAnimateMotionElement**
SMILAnimateMotionElement has the all the properties and methods of **SMILAnimateElement** as well as the properties and methods defined below.
The **SMILAnimateMotionElement** object has the following properties:
path
This property is of type **String**.
origin
This property is of type **String**.

Object **SMILSwitchElement**

SMILSwitchElement has the all the properties and methods of **SMILElement** as well as the properties and methods defined below.

The **SMILSwitchElement** object has the following properties:

title

This property is of type **String**.

Object **ElementTest**

The **ElementTest** object has the following properties:

systemBitrate

This property is of type **String**.

systemCaptions

This property is of type **String**.

systemLanguage

This property is of type **String**.

systemOverdubOrCaption

This property is of type **String**.

systemRequired

This property is of type **String**.

systemScreenSize

This property is of type **String**.

systemScreenDepth

This property is of type **String**.

Acknowledgments

This document has been prepared by the Synchronized Multimedia Working Group (WG) of the World Wide Web Consortium. The WG includes the following individuals:

- Jeff Ayars, RealNetworks
- Dick Bulterman, Oratrix (Invited Expert)
- Wayne Carr, Intel
- Wo Chang, NIST
- Aaron Cohen, Intel
- Ken Day, Macromedia
- Geoff Freed, WGBH
- Mark Hakkinen, Productivity Works
- Lynda Hardman, CWI
- Masayuki Hiyama, Glocomm
- Erik Hodge, RealNetworks
- Philipp Hoschka, W3C
- Eric Hyche, RealNetworks
- Jack Jansen, Oratrix (Invited Expert)
- Muriel Jourdan, INRIA
- Keisuke Kamimura, Glocomm
- Kenichi Kubota, Panasonic
- Nabil Layaïda, INRIA
- Rob Lanphier, RealNetworks
- Philippe Le Hégaret, W3C
- Pietro Marchisio, CSELT
- Thierry Michel, W3C
- Sjoerd Mullender, Oratrix (Invited Expert)
- Debbie Newman, Microsoft
- Jacco van Ossenbruggen, CWI
- Didier Pillet, France Telecom
- Hanan Rosenthal, Canon
- Lloyd Rutledge, CWI
- Bridie Saccocio, RealNetworks
- Patrick Schmitz, Microsoft
- Warner ten Kate, Philips
- Daniel Weber, Matsushita
- Gary Wiemann, National Security Agency
- Ted Wugofski, Gateway (Invited Expert)
- Jin Yu, Compaq

Acknowledgments

References

DOM Requirements

Document Object Model Requirements for Synchronized Multimedia. see http://www.w3.org/AudioVideo/Group/DOM/DOM_reqts, (W3C Members only).

SMIL 1.0

W3C (World Wide Web Consortium) *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*. See http://www.w3.org/TR/REC-smil.

SMIL Boston

W3C (World Wide Web Consortium) *Synchronized Multimedia Integration Language (SMIL) Boston Specification*. See http://www.w3.org/TR/smil-boston.

SMIL Animation

W3C (World Wide Web Consortium) *Synchronized Multimedia Integration Language (SMIL) Animation*. See http://www.w3.org/TR/smil-animation.

DOM Level 2

W3C (World Wide Web Consortium) *Document Object Model (DOM) Level 2 Specification*. See http://www.w3.org/TR/WD-DOM-Level-2.

XML Namespaces

W3C (World Wide Web Consortium) *Namespaces in XML*. See http://www.w3.org/TR/REC-xml-names.

XLink

W3C (World Wide Web Consortium) *XML Linking Language (XLink)*. See http://www.w3.org/TR/xlink.

References

Index