Synchronized Multimedia Integration Language Document Object Model

W3C Working Draft *25 February, 2000*

This version:
http://www.w3.org/TR/2000/WD-smil-boston-dom-20000225
(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/TR/1999/WD-smil-boston-dom-19991115

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

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.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

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.

Table of contents

Expanded Table of Contents and Copyright

Expanded Table of Contents

Copyright Notice

W3C Document Copyright Notice and License

Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/" (Hypertext is preferred, but a textual representation is permitted.)
3. *If it exists*, the STATUS of the W3C document.

When space permits, inclusion of the full text of this **NOTICE** should be provided. We request that authorship attribution be provided in any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of W3C documents is granted pursuant to this license. However, subsequent to additional requirements documented in the Copyright FAQ, modifications or derivatives are sometimes granted by the W3C to individuals complying with those terms.

THIS DOCUMENT IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

W3C Software Copyright Notice and License

Note: This section is a copy of the W3C Software Copyright Notice and License and could be found at http://www.w3.org/Consortium/Legal/copyright-software-19980720

Copyright © 1994-2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

http://www.w3.org/Consortium/Legal/

This W3C work (including software, documents, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, and modify this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications, that you make:

1. Introduction and Requirements

1.1. 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 Level 2 Core [p.9]). 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.

1.2. 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 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.

2. 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.

2.1. 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.

2.1.1. 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).

2.1.2. 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.)

2.2. 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.

2.2.1. 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.

2.3. 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.
(***ED:*** Perhaps we should clarify this to refer more specifically to documents incorporating SMIL Layout.)

3. 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.

3.1. 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.

3.2. 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`.)

3.3. 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.

4. 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 "org.w3c.dom.smil". The version of this DOM version 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*).

4.1. 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, ElementSequentialTimeContainer {
};
```

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.

4.2. Structure extensions

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

4.3. Meta informations extensions

This SMIL Boston module doesn't have a corresponding SMIL Boston DOM module.
(***ED:*** Relation with a RDF DOM ?)

4.4. Layout extensions

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*.

IDL Definition

```
interface SMILLayoutElement : SMILElement {
  readonly attribute DOMString        type;
  readonly attribute boolean          resolved;
};
```

Attributes

`type` of type `DOMString`, readonly

The mime type of the layout langage used in this layout element.The default value of the type attribute is "text/smil-basic-layout".

`resolved` of type `boolean`, readonly

`true` if the player can understand the mime type, `false` otherwise.

Interface *ElementLayout*

This interface is used by SMIL elements root-layout, top-layout and region.

IDL Definition

```
interface ElementLayout {
          attribute DOMString        title;
                                       // 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.

`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 *SMILTopLayoutElement*

Declares layout properties for the top-layout element. See the *top-layout element definition*.

IDL Definition

```
interface SMILTopLayoutElement : SMILElement, ElementLayout {
};
```

Interface *SMILRootLayoutElement*

Declares layout properties for the root-layout element. See the *root-layout element definition*.

IDL Definition

```
interface SMILRootLayoutElement : SMILElement, ElementLayout {
};
```

Interface *SMILRegionElement*

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

IDL Definition

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

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

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

};
```

Attributes

`fit` of type `DOMString`

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*.
IDL Definition

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

Attributes
`region` of type `SMILRegionElement` [p.18]

4.5. 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.
(***ED:*** Still need to address the wallclock values.)
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.19]	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

  // restartTypes
  const unsigned short      RESTART_ALWAYS                  = 0;
  const unsigned short      RESTART_NEVER                   = 1;
  const unsigned short      RESTART_WHEN_NOT_ACTIVE         = 2;

           attribute unsigned short   restart;
                                         // raises(DOMException) on setting

  // fillTypes
  const unsigned short      FILL_REMOVE                     = 0;
  const unsigned short      FILL_FREEZE                     = 1;

           attribute unsigned short   fill;
                                         // raises(DOMException) on setting

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

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

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

Attributes

`begin` of type `TimeList` [p.23]

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.23]

The list of active *ends* for 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.

Definition group *restartTypes*

An integer indicating the value of the `restart` attribute.

Defined Constants

RESTART_ALWAYS

RESTART_NEVER

RESTART_WHEN_NOT_ACTIVE

`restart` of type `unsigned short`

A code representing the value of the *restart* attribute, as defined above. Default value is `RESTART_ALWAYS`.

Exceptions on setting

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

Definition group *fillTypes*

An integer indicating the value of the `fill` attribute.

Defined Constants

FILL_REMOVE

FILL_FREEZE

`fill` of type `unsigned short`

A code representing the value of the *fill* attribute, as defined above. Default value is `FILL_REMOVE`.

Exceptions on setting

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

`repeatCount` of type `float`

The *repeatCount* attribute causes the element to play repeatedly (loop) for the specified number of times. A negative value repeat the element indefinitely. Default value is 0 (unspecified).

Exceptions on setting

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

`repeatDur` of type `float`

The *repeatDur* 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 `"undefinite"`)
- 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 `"undefinite"`) • 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

`float`	`seekTo`	The desired position on the local timeline in milliseconds.

No Return Value
No Exceptions

Interface *ElementTimeManipulation*

This interface support use-cases commonly associated with animation.
(***ED:*** "accelerate" and "decelerate" are float values in the timing draft and percentage values even in this draft if both of them represent a percentage.)
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 *autoReverse* attribute controls 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 *ElementTimeContainer*

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

IDL Definition

```
interface ElementTimeContainer : ElementTime {
  readonly attribute NodeList         timeChildren;
  NodeList           getActiveChildrenAt(in float instant);
};
```

Attributes

`timeChildren` of type `NodeList`, readonly

A NodeList that contains all timed childrens of this node. If there are no timed children, the `Nodelist` is empty.
(***ED:*** 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 specified invocation.

Parameters

`float`	`instant`	The desired position on the local timeline in milliseconds.

Return Value

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

No Exceptions

Interface *ElementSyncBehavior*

The synchronization behavior extension.

IDL Definition

```
interface ElementSyncBehavior {
  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 *ElementParallelTimeContainer*

A `parallel` container defines a simple parallel time grouping in which multiple elements can play back at the same time.
(***ED:*** It may have to specify a repeat iteration. (?))
IDL Definition

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

  float              getImplicitDuration();
};
```

Attributes
`endSync` of type `DOMString`
Controls the end of the container.
(***ED:*** Need to address thr id-ref value.)
Exceptions on setting

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

Methods
`getImplicitDuration`
This method returns the implicit duration in seconds.
Return Value

`float`	The implicit duration in seconds or -1 if the implicit is unknown (indefinite?).

No Parameters
No Exceptions

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

  NodeList           getPausedElements();
};
```

Attributes

`endSync` of type `DOMString`

Controls the end of the container.
(***ED:*** Need to address thr id-ref value.)

Exceptions on setting

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

Methods

`getPausedElements`

This should support another method to get the ordered collection of paused elements (the paused stack) at a given point in time.

Return Value

`NodeList`	All paused elements at the current time.

No Parameters
No Exceptions

4.5.1. Timing Events

Interface *TimeEvent*

The `TimeEvent` interface provides specific contextual information associated with Time events.

IDL Definition

```
interface TimeEvent : events::Event {
  readonly attribute views::AbstractView  view;
  readonly attribute long             detail;
  void               initTimeEvent(in DOMString typeArg,
                                   in views::AbstractView viewArg,
                                   in long detailArg);
};
```

Attributes

`view` of type `views::AbstractView`, readonly

The `view` attribute identifies the `AbstractView` from which the event was generated.

`detail` of type `long`, readonly

Specifies some detail information about the `Event`, depending on the type of event.

Methods

`initTimeEvent`

The `initTimeEvent` method is used to initialize the value of a `TimeEvent` created through the `DocumentEvent` interface. This method may only be called before the `TimeEvent` has been dispatched via the `dispatchEvent` method, though it may be called multiple times during that phase if necessary. If called multiple times, the final invocation takes precedence.

Parameters

`DOMString`	`typeArg`	Specifies the event type.
`views::AbstractView`	`viewArg`	Specifies the `Event`'s `AbstractView`.
`long`	`detailArg`	Specifies the `Event`'s detail.

No Return Value

No Exceptions

The different types of events that can occur are:

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. when it restarts, but not when it repeats). 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() or beginElementAt() methods. 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.

- Bubbles: No
- Cancelable: No
- Context Info: None

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. 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() or endElementAt() methods. 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.

- Bubbles: No
- Cancelable: No
- Context Info: None

repeat

This event is raised when the element local timeline repeats. It will be raised each time the element repeats, after the first iteration.

The `Time` [p.19] event interface supports a property "detail" that indicates which repeat iteration is beginning. The value is a 0-based integer, but the repeat event is not raised for the first iteration and so the observed values will be >= 1. of the element ?

- Bubbles: No
- Cancelable: No
- Context Info: number of iterations

4.6. 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.

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        port;
                                     // raises(DOMException) on setting

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

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

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

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

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

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

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

};
```

Attributes

`abstractAttr` of type `DOMString`

See the *abstract* attribute from [SMIL Boston].

Exceptions on setting

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

`alt` of type `DOMString`

See the *alt* attribute from [SMIL Boston].

Exceptions on setting

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

`author` of type `DOMString`

See the *author* attribute from [SMIL Boston].

Exceptions on setting

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

`clipBegin` of type `DOMString`
See the *clipBegin* attribute from [SMIL Boston].
Exceptions on setting

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

`clipEnd` of type `DOMString`
See the *clipEnd* attribute from [SMIL Boston].
Exceptions on setting

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

`copyright` of type `DOMString`
See the *copyright* attribute from [SMIL Boston].
Exceptions on setting

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

`longdesc` of type `DOMString`
See the *longdesc* attribute from [SMIL Boston].
Exceptions on setting

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

`port` of type `DOMString`
See the *port* attribute from [SMIL Boston].
Exceptions on setting

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

`readIndex` of type `DOMString`
See the *readIndex* attribute from [SMIL Boston].
Exceptions on setting

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

`rtpformat` of type `DOMString`
See the *rtpformat* attribute from [SMIL Boston].
Exceptions on setting

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

`src` of type `DOMString`
See the *src* attribute from [SMIL Boston].
Exceptions on setting

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

`stripRepeat` of type `DOMString`
See the *stripRepeat* attribute from [SMIL Boston].
Exceptions on setting

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

`title` of type `DOMString`
See the *title* attribute from [SMIL Boston].
Exceptions on setting

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

`transport` of type `DOMString`
See the *transport* attribute from [SMIL Boston].
Exceptions on setting

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

`type` of type `DOMString`
See the *type* attribute from [SMIL Boston].
Exceptions on setting

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

Interface *SMILRefElement*

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

IDL Definition

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

4.7. Animations extensions

This module will include interfaces associated with animation extensions.

4.7.1. SMIL Animation

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

Interface *ElementTimeControl*

IDL Definition

```
interface ElementTimeControl {
  boolean             beginElement()
                                     raises(DOMException);
  boolean             beginElementAt(in float offset)
                                     raises(DOMException);
  boolean             endElement()
                                     raises(DOMException);
  boolean             endElementAt(in float offset)
                                     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 `begin` attribute is not set to `"indefinite"`)
- 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

`beginElementAt`

Causes this element to begin the local timeline (subject to sync constraints), at the passed offset from the current time when the method is called. If the offset is >= 0, the semantics are equivalent to an event-base begin with the specified offset. If the offset is < 0, the semantics are equivalent to beginElement(), but the element active duration is evaluated as though the element had begun at the passed (negative) offset from the current time when the method is called.

Parameters

`float`	`offset`	The offset in seconds at which to begin the element.

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 `beginElementAt` method. (the `begin` attribute is not set to `"indefinite"`) • 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 `beginElementAt` calls.

`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 `end` attribute is not set to `"indefinite"`) • The element is not active.

Exceptions

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

No Parameters

`endElementAt`

Causes this element to end the local timeline (subject to sync constraints) at the specified offset from the current time when the method is called.

Parameters

`float`	`offset`	The offset in seconds at which to end the element. Must be >= 0.

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 `endElementAt` method. (the `end` attribute is not set to `"indefinite"`)
- The element is not active.

Exceptions

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

4.7.2. SMIL Boston Animation

Interface *SMILAnimation*

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

```
interface SMILAnimation : SMILElement, ElementTargetAttributes, ElementTime, ElementTimeControl {
  // additiveTypes
  const unsigned short      ADDITIVE_REPLACE               = 0;
  const unsigned short      ADDITIVE_SUM                   = 1;

           attribute unsigned short   additive;
                                        // raises(DOMException) on setting

  // accumulateTypes
  const unsigned short      ACCUMULATE_NONE                = 0;
  const unsigned short      ACCUMULATE_SUM                 = 1;

           attribute unsigned short   accumulate;
                                        // raises(DOMException) on setting
```

```
   // calcModeTypes
   const unsigned short      CALCMODE_DISCRETE              = 0;
   const unsigned short      CALCMODE_LINEAR                = 1;
   const unsigned short      CALCMODE_PACED                 = 2;
   const unsigned short      CALCMODE_SPLINE                = 3;

            attribute unsigned short   calcMode;
                                         // raises(DOMException) on setting

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

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

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

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

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

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

};
```

Definition group *additiveTypes*

Defined Constants

ADDITIVE_REPLACE

ADDITIVE_SUM

Attributes

`additive` of type `unsigned short`

A code representing the value of the *additive* attribute, as defined above. Default value is `ADDITIVE_REPLACE`.

Exceptions on setting

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

Definition group *accumulateTypes*

Defined Constants

ACCUMULATE_NONE

ACCUMULATE_SUM

`accumulate` of type `unsigned short`

A code representing the value of the *accumulate* attribute, as defined above. Default value is `ACCUMULATE_NONE`.

Exceptions on setting

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

Definition group *calcModeTypes*

Defined Constants

CALCMODE_DISCRETE

CALCMODE_LINEAR

CALCMODE_PACED

CALCMODE_SPLINE

`calcMode` of type `unsigned short`

A code representing the value of the *calcMode* attribute, as defined above.

Exceptions on setting

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

`keySplines` of type `DOMString`

A `DOMString` representing the value of the *keySplines* attribute.
(***ED:*** Need an interface a point (x1,y1,x2,y2))

Exceptions on setting

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

`keyTimes` of type `TimeList` [p.23]

A list of the time value of the *keyTimes* attribute.

Exceptions on setting

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

`values` of type `DOMString`

A `DOMString` representing the value of the *values* attribute.

Exceptions on setting

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

`from` of type `DOMString`
A `DOMString` representing the value of the *from* attribute.
Exceptions on setting

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

`to` of type `DOMString`
A `DOMString` representing the value of the *to* attribute.
Exceptions on setting

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

`by` of type `DOMString`
A `DOMString` representing the value of the *by* attribute.
Exceptions on setting

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

Interface *ElementTargetAttributes*

This interface define the set of animation target extensions.
IDL Definition

```
interface ElementTargetAttributes {
           attribute DOMString        attributeName;
  // attributeTypes
  const unsigned short      ATTRIBUTE_TYPE_AUTO            = 0;
  const unsigned short      ATTRIBUTE_TYPE_CSS             = 1;
  const unsigned short      ATTRIBUTE_TYPE_XML             = 2;

           attribute unsigned short   attributeType;
};
```

Attributes
`attributeName` of type `DOMString`
The name of the target attribute.

Definition group *attributeTypes*
Defined Constants

ATTRIBUTE_TYPE_AUTO

ATTRIBUTE_TYPE_CSS

ATTRIBUTE_TYPE_XML

`attributeType` of type `unsigned short`
A code representing the value of the *attributeType* attribute, as defined above. Default value is `ATTRIBUTE_TYPE_CODE`.

Interface *SMILAnimateElement*

This interface represents the SMIL `animate` element.
IDL Definition

```
interface SMILAnimateElement : SMILAnimation {
};
```

Interface *SMILSetElement*

This interface represents the *set* element.
IDL Definition

```
interface SMILSetElement : ElementTimeControl, ElementTime, ElementTargetAttributes, SMILElement {
         attribute DOMString       to;
};
```

Attributes
`to` of type `DOMString`
Specifies the value for the attribute during the duration of this element.

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.
(***ED:*** Check with the SVG spec for better support)
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.

Interface *SMILAnimateColorElement*

This interface represents the SMIL `animateColor` element.
IDL Definition

```
interface SMILAnimateColorElement : SMILAnimation {
};
```

4.8. Transition extensions

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

4.9. Linking extensions

This module includes interfaces for hyperlinking elements.

4.10. Content Control extensions

This module includes interfaces for content control markup.

Interface *SMILSwitchElement*

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

```
interface SMILSwitchElement : SMILElement {
  Element             getSelectedElement();
};
```

Methods
`getSelectedElement`
Returns the slected element at runtime. `null` if the selected element is not yet available.
Return Value

`Element`	The selected `Element` for thisd `switch` element.

No Parameters
No Exceptions

Interface *ElementTest*

Defines the test attributes interface. See the *Test attributes definition*.

IDL Definition

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

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

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

  readonly attribute boolean        systemRequired;
  readonly attribute boolean        systemScreenSize;
  readonly attribute boolean        systemScreenDepth;
          attribute DOMString       systemOverdubOrSubtitle;
                                      // raises(DOMException) on setting

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

};
```

Attributes

`systemBitrate` of type `long`

The *systemBitrate* value.

Exceptions on setting

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

`systemCaptions` of type `boolean`

The *systemCaptions* value.

Exceptions on setting

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

`systemLanguage` of type `DOMString`

The *systemLanguage* value.

Exceptions on setting

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

`systemRequired` of type `boolean`, readonly
The result of the evaluation of the *systemRequired* attribute.

`systemScreenSize` of type `boolean`, readonly
The result of the evaluation of the *systemScreenSize* attribute.

`systemScreenDepth` of type `boolean`, readonly
The result of the evaluation of the *systemScreenDepth* attribute.

`systemOverdubOrSubtitle` of type `DOMString`
The value of the *systemOverdubOrSubtitle* attribute.
Exceptions on setting

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

`systemAudioDesc` of type `boolean`
The value of the *systemAudioDesc* attribute.
Exceptions on setting

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

4.11. 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.)

4.11.1. Media Player Level 1 Interface

4.11.2. Media Player Level 2 Interface

4.11.3. 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.47] .

The IDL files are also available as: http://www.w3.org/TR/2000/WD-smil-boston-dom-20000225/idl.zip

A.1: SYMM Document Object Model

smil.idl:

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

#include "dom.idl"
#include "views.idl"
#include "events.idl"

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

  interface SMILRegionElement;

  interface ElementLayout {
             attribute DOMString        title;
                                        // 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 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

  // restartTypes
  const unsigned short      RESTART_ALWAYS                 = 0;
  const unsigned short      RESTART_NEVER                  = 1;
  const unsigned short      RESTART_WHEN_NOT_ACTIVE        = 2;

           attribute unsigned short   restart;
                                      // raises(dom::DOMException) on setting

  // fillTypes
  const unsigned short      FILL_REMOVE                    = 0;
  const unsigned short      FILL_FREEZE                    = 1;

           attribute unsigned short   fill;
                                      // raises(dom::DOMException) on setting

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

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

  boolean            beginElement();
  boolean            endElement();
  void               pauseElement();
  void               resumeElement();
  void               seekElement(inout float 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 ElementTimeContainer : ElementTime {
  readonly attribute NodeList          timeChildren;
  NodeList            getActiveChildrenAt(in float instant);
};

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

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

  float               getImplicitDuration();
};

interface ElementSequentialTimeContainer : ElementTimeContainer {
};

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

  NodeList            getPausedElements();
};

interface ElementTimeControl {
  boolean             beginElement()
                                  raises(dom::DOMException);
  boolean             beginElementAt(in float offset)
                                  raises(dom::DOMException);
  boolean             endElement()
                                  raises(dom::DOMException);
  boolean             endElementAt(in float offset)
                                  raises(dom::DOMException);
};

interface ElementTargetAttributes {
           attribute DOMString         attributeName;
  // attributeTypes
  const unsigned short       ATTRIBUTE_TYPE_AUTO            = 0;
  const unsigned short       ATTRIBUTE_TYPE_CSS             = 1;
  const unsigned short       ATTRIBUTE_TYPE_XML             = 2;

           attribute unsigned short    attributeType;
};

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

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

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

  readonly attribute boolean          systemRequired;
  readonly attribute boolean          systemScreenSize;
  readonly attribute boolean          systemScreenDepth;
           attribute DOMString        systemOverdubOrSubtitle;
                                      // raises(dom::DOMException) on setting

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

};

interface SMILDocument : Document, ElementSequentialTimeContainer {
};

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

};

interface SMILLayoutElement : SMILElement {
  readonly attribute DOMString        type;
  readonly attribute boolean          resolved;
};

interface SMILTopLayoutElement : SMILElement, ElementLayout {
};

interface SMILRootLayoutElement : SMILElement, ElementLayout {
};

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

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

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

};

interface TimeEvent : events::Event {
  readonly attribute views::AbstractView  view;
  readonly attribute long             detail;
  void               initTimeEvent(in DOMString typeArg,
                                   in views::AbstractView viewArg,
                                   in long detailArg);
};

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        port;
                                  // raises(dom::DOMException) on setting

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

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

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

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

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

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

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

};

interface SMILRefElement : SMILMediaElement {
};

interface SMILAnimation : SMILElement, ElementTargetAttributes, ElementTime, ElementTimeControl {
  // additiveTypes
  const unsigned short      ADDITIVE_REPLACE               = 0;
  const unsigned short      ADDITIVE_SUM                   = 1;

           attribute unsigned short   additive;
                                  // raises(dom::DOMException) on setting

  // accumulateTypes
  const unsigned short      ACCUMULATE_NONE                = 0;
  const unsigned short      ACCUMULATE_SUM                 = 1;

           attribute unsigned short   accumulate;
                                  // raises(dom::DOMException) on setting

  // calcModeTypes
  const unsigned short      CALCMODE_DISCRETE              = 0;
  const unsigned short      CALCMODE_LINEAR                = 1;
  const unsigned short      CALCMODE_PACED                 = 2;
  const unsigned short      CALCMODE_SPLINE                = 3;

           attribute unsigned short   calcMode;
```

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

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

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

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

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

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

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

  };

  interface SMILAnimateElement : SMILAnimation {
  };

  interface SMILSetElement : ElementTimeControl, ElementTime, ElementTargetAttributes, SMILElement {
             attribute DOMString        to;
  };

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

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

  };

  interface SMILAnimateColorElement : SMILAnimation {
  };

  interface SMILSwitchElement : SMILElement {
    Element              getSelectedElement();
  };
};

#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.53] .

The Java files are also available as http://www.w3.org/TR/2000/WD-smil-boston-dom-20000225/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, ElementSequentialTimeContainer {
}
```

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/SMILLayoutElement.java:

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

public interface SMILLayoutElement extends SMILElement {
    public String getType();

    public boolean getResolved();

}
```

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

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

import org.w3c.dom.DOMException;

public interface ElementLayout {
```

```
    public String getTitle();
    public void setTitle(String title)
 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/SMILTopLayoutElement.java:

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

public interface SMILTopLayoutElement extends SMILElement, ElementLayout {
}
```

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

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

public interface SMILRootLayoutElement extends SMILElement, ElementLayout {
}
```

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

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

import org.w3c.dom.DOMException;

public interface SMILRegionElement extends SMILElement, ElementLayout {
    public String getFit();
    public void setFit(String fit)
 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;

    // restartTypes
    public static final short RESTART_ALWAYS            = 0;
    public static final short RESTART_NEVER             = 1;
    public static final short RESTART_WHEN_NOT_ACTIVE   = 2;

    public short getRestart();
    public void setRestart(short restart)
                     throws DOMException;

    // fillTypes
    public static final short FILL_REMOVE               = 0;
    public static final short FILL_FREEZE               = 1;

    public short getFill();
    public void setFill(short fill)
                     throws DOMException;

    public float getRepeatCount();
    public void setRepeatCount(float repeatCount)
                     throws DOMException;

    public float getRepeatDur();
    public void setRepeatDur(float repeatDur)
                     throws DOMException;

    public boolean beginElement();
```

```
    public boolean endElement();

    public void pauseElement();

    public void resumeElement();

    public void seekElement(float 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/ElementTimeContainer.java:

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

import org.w3c.dom.NodeList;

public interface ElementTimeContainer extends ElementTime {
    public NodeList getTimeChildren();

    public NodeList getActiveChildrenAt(float instant);

}
```

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

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

public interface ElementSyncBehavior {
    public String getSyncBehavior();

    public float getSyncTolerance();

    public String getDefaultSyncBehavior();

    public float getDefaultSyncTolerance();

    public boolean getSyncMaster();

}
```

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;

    public float getImplicitDuration();

}
```

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;
import org.w3c.dom.NodeList;

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

    public NodeList getPausedElements();

}
```

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

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

import org.w3c.dom.events.Event;
import org.w3c.dom.views.AbstractView;

public interface TimeEvent extends Event {
    public AbstractView getView();

    public int getDetail();

    public void initTimeEvent(String typeArg,
                              AbstractView viewArg,
                              int detailArg);

}
```

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 getPort();
    public void setPort(String port)
                              throws DOMException;
```

```
    public String getReadIndex();
    public void setReadIndex(String readIndex)
                              throws DOMException;

    public String getRtpformat();
    public void setRtpformat(String rtpformat)
                              throws DOMException;

    public String getSrc();
    public void setSrc(String src)
                              throws DOMException;

    public String getStripRepeat();
    public void setStripRepeat(String stripRepeat)
                              throws DOMException;

    public String getTitle();
    public void setTitle(String title)
                              throws DOMException;

    public String getTransport();
    public void setTransport(String transport)
                              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 beginElementAt(float offset)
                                  throws DOMException;

    public boolean endElement()
                              throws DOMException;
```

```
    public boolean endElementAt(float offset)
                                   throws DOMException;

}
```

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

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

import org.w3c.dom.DOMException;

public interface SMILAnimation extends SMILElement, ElementTargetAttributes, ElementTime, ElementTimeControl {
    // additiveTypes
    public static final short ADDITIVE_REPLACE          = 0;
    public static final short ADDITIVE_SUM              = 1;

    public short getAdditive();
    public void setAdditive(short additive)
                            throws DOMException;

    // accumulateTypes
    public static final short ACCUMULATE_NONE           = 0;
    public static final short ACCUMULATE_SUM            = 1;

    public short getAccumulate();
    public void setAccumulate(short accumulate)
                            throws DOMException;

    // calcModeTypes
    public static final short CALCMODE_DISCRETE         = 0;
    public static final short CALCMODE_LINEAR           = 1;
    public static final short CALCMODE_PACED            = 2;
    public static final short CALCMODE_SPLINE           = 3;

    public short getCalcMode();
    public void setCalcMode(short calcMode)
                            throws DOMException;

    public String getKeySplines();
    public void setKeySplines(String keySplines)
                            throws DOMException;

    public TimeList getKeyTimes();
    public void setKeyTimes(TimeList keyTimes)
                            throws DOMException;

    public String getValues();
    public void setValues(String values)
                            throws DOMException;

    public String getFrom();
    public void setFrom(String from)
                            throws DOMException;

    public String getTo();
    public void setTo(String to)
                            throws DOMException;

    public String getBy();
    public void setBy(String by)
                            throws DOMException;

}
```

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

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

public interface ElementTargetAttributes {
    public String getAttributeName();
    public void setAttributeName(String attributeName);

    // attributeTypes
    public static final short ATTRIBUTE_TYPE_AUTO       = 0;
    public static final short ATTRIBUTE_TYPE_CSS        = 1;
    public static final short ATTRIBUTE_TYPE_XML        = 2;

    public short getAttributeType();
    public void setAttributeType(short attributeType);

}
```

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

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

public interface SMILAnimateElement extends SMILAnimation {
}
```

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

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

public interface SMILSetElement extends ElementTimeControl, ElementTime, ElementTargetAttributes, SMILElement {
    public String getTo();
    public void setTo(String to);

}
```

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/SMILAnimateColorElement.java:

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

public interface SMILAnimateColorElement extends SMILAnimation {
}
```

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

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

import org.w3c.dom.Element;

public interface SMILSwitchElement extends SMILElement {
    public Element getSelectedElement();

}
```

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

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

import org.w3c.dom.DOMException;

public interface ElementTest {
    public int getSystemBitrate();
    public void setSystemBitrate(int systemBitrate)
                                  throws DOMException;

    public boolean getSystemCaptions();
    public void setSystemCaptions(boolean systemCaptions)
                                  throws DOMException;

    public String getSystemLanguage();
    public void setSystemLanguage(String systemLanguage)
                                  throws DOMException;

    public boolean getSystemRequired();

    public boolean getSystemScreenSize();

    public boolean getSystemScreenDepth();

    public String getSystemOverdubOrSubtitle();
    public void setSystemOverdubOrSubtitle(String systemOverdubOrSubtitle)
                                  throws DOMException;

    public boolean getSystemAudioDesc();
    public void setSystemAudioDesc(boolean systemAudioDesc)
                                  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.65] .

C.1: SMIL Document Object Model

Object **SMILDocument**
  SMILDocument has the all the properties and methods of **Document ElementSequentialTimeContainer** 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 **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**.
    resolved
      This property is of type **boolean**.
Object **ElementLayout**
  The **ElementLayout** object has the following properties:
    title
      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 **SMILTopLayoutElement**
  SMILTopLayoutElement has the all the properties and methods of **SMILElement ElementLayout** as well as the properties and methods defined below.
Object **SMILRootLayoutElement**
  SMILRootLayoutElement has the all the properties and methods of **SMILElement ElementLayout** as well as the properties and methods defined below.
Object **SMILRegionElement**
  SMILRegionElement has the all the properties and methods of **SMILElement ElementLayout** as well as the properties and methods defined below.
  The **SMILRegionElement** object has the following properties:

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

Class **Time**
The **Time** class has the following constants:
Time.SMIL_TIME_INDEFINITE
This constant is of type **short** and its value is **0**.
Time.SMIL_TIME_OFFSET
This constant is of type **short** and its value is **1**.
Time.SMIL_TIME_SYNC_BASED
This constant is of type **short** and its value is **2**.
Time.SMIL_TIME_EVENT_BASED
This constant is of type **short** and its value is **3**.
Time.SMIL_TIME_WALLCLOCK
This constant is of type **short** and its value is **4**.
Time.SMIL_TIME_MEDIA_MARKER
This constant is of type **short** and its value is **5**.

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.

Class **ElementTime**
The **ElementTime** class has the following constants:
ElementTime.RESTART_ALWAYS
This constant is of type **short** and its value is **0**.
ElementTime.RESTART_NEVER
This constant is of type **short** and its value is **1**.
ElementTime.RESTART_WHEN_NOT_ACTIVE
This constant is of type **short** and its value is **2**.
ElementTime.FILL_REMOVE
This constant is of type **short** and its value is **0**.
ElementTime.FILL_FREEZE
This constant is of type **short** and its value is **1**.

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**.
restart
This property is of type **short**.
fill
This property is of type **short**.
repeatCount
This property is of type **float**.
repeatDur
This property is of type **float**.

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 **float**.

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 **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:

timeChildren
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 **float**.

Object **ElementSyncBehavior**

The **ElementSyncBehavior** 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 **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**.

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

getImplicitDuration()
This method returns a **float**.

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**.
The **ElementExclusiveTimeContainer** object has the following methods:
getPausedElements()
This method returns a **NodeList**.
Object **TimeEvent**
TimeEvent has the all the properties and methods of **Event** as well as the properties and methods defined below.
The **TimeEvent** object has the following properties:
view
This property is of type **AbstractView**.
detail
This property is of type **long**.
The **TimeEvent** object has the following methods:
initTimeEvent(typeArg, viewArg, detailArg)
This method returns a **void**. The **typeArg** parameter is of type **DOMString**. The **viewArg** parameter is of type **views::AbstractView**. The **detailArg** parameter is of type **long**.
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**.
port
This property is of type **String**.
readIndex
This property is of type **String**.
rtpformat
This property is of type **String**.
src
This property is of type **String**.
stripRepeat
This property is of type **String**.

title
This property is of type **String**.
transport
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**.
beginElementAt(offset)
This method returns a **boolean**. The **offset** parameter is of type **float**.
endElement()
This method returns a **boolean**.
endElementAt(offset)
This method returns a **boolean**. The **offset** parameter is of type **float**.

Class **SMILAnimation**
The **SMILAnimation** class has the following constants:
SMILAnimation.ADDITIVE_REPLACE
This constant is of type **short** and its value is **0**.
SMILAnimation.ADDITIVE_SUM
This constant is of type **short** and its value is **1**.
SMILAnimation.ACCUMULATE_NONE
This constant is of type **short** and its value is **0**.
SMILAnimation.ACCUMULATE_SUM
This constant is of type **short** and its value is **1**.
SMILAnimation.CALCMODE_DISCRETE
This constant is of type **short** and its value is **0**.
SMILAnimation.CALCMODE_LINEAR
This constant is of type **short** and its value is **1**.
SMILAnimation.CALCMODE_PACED
This constant is of type **short** and its value is **2**.
SMILAnimation.CALCMODE_SPLINE
This constant is of type **short** and its value is **3**.

Object **SMILAnimation**
SMILAnimation has the all the properties and methods of **SMILElement** **ElementTargetAttributes** **ElementTime** **ElementTimeControl** as well as the properties and methods defined below.
The **SMILAnimation** object has the following properties:
additive
This property is of type **short**.
accumulate
This property is of type **short**.

calcMode
This property is of type **short**.
keySplines
This property is of type **String**.
keyTimes
This property is of type **TimeList**.
values
This property is of type **String**.
from
This property is of type **String**.
to
This property is of type **String**.
by
This property is of type **String**.

Class **ElementTargetAttributes**
The **ElementTargetAttributes** class has the following constants:
ElementTargetAttributes.ATTRIBUTE_TYPE_AUTO
This constant is of type **short** and its value is **0**.
ElementTargetAttributes.ATTRIBUTE_TYPE_CSS
This constant is of type **short** and its value is **1**.
ElementTargetAttributes.ATTRIBUTE_TYPE_XML
This constant is of type **short** and its value is **2**.

Object **ElementTargetAttributes**
The **ElementTargetAttributes** object has the following properties:
attributeName
This property is of type **String**.
attributeType
This property is of type **short**.

Object **SMILAnimateElement**
SMILAnimateElement has the all the properties and methods of **SMILAnimation** as well as the properties and methods defined below.

Object **SMILSetElement**
SMILSetElement has the all the properties and methods of **ElementTimeControl ElementTime ElementTargetAttributes SMILElement** as well as the properties and methods defined below.
The **SMILSetElement** object has the following properties:
to
This property is of type **String**.

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 **SMILAnimateColorElement**

SMILAnimateColorElement has the all the properties and methods of **SMILAnimation** as well as the properties and methods defined below.

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 methods:

getSelectedElement()

This method returns a **Element**.

Object **ElementTest**

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

systemBitrate

This property is of type **long**.

systemCaptions

This property is of type **boolean**.

systemLanguage

This property is of type **String**.

systemRequired

This property is of type **boolean**.

systemScreenSize

This property is of type **boolean**.

systemScreenDepth

This property is of type **boolean**.

systemOverdubOrSubtitle

This property is of type **String**.

systemAudioDesc

This property is of type **boolean**.

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

SMIL 1.0
: W3C (World Wide Web Consortium) *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*. See http://www.w3.org/TR/1998/REC-smil-19980615.

SMIL Boston
: W3C (World Wide Web Consortium) *Synchronized Multimedia Integration Language (SMIL) Boston Specification*. See http://www.w3.org/TR/2000/WD-smil-boston-20000225.

SMIL Animation
: W3C (World Wide Web Consortium) *Synchronized Multimedia Integration Language (SMIL) Animation*. See http://www.w3.org/TR/2000/WD-smil-animation-20000128.

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