SISO-STD-009-01

Standard for
Simulation Reference Markup
Language
Engine Specification – Level 0

Version 1.0

9 March 2017

Prepared by
Simulation Reference Markup
Language
Product Development Group
# Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Section</th>
<th>Date (MM/DD/YYYY)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>All</td>
<td>08/14/2016</td>
<td>Circulation Package Version</td>
</tr>
</tbody>
</table>
Participants

At the time this product was submitted to the Standards Activity Committee (SAC) for approval, the Simulation Reference Markup Language Product Development Group (PDG) had the following membership and was assigned the following SAC Technical Activity Director:

Product Development Group

Robert Lutz (Chair)
Jane Bachman (Vice-Chair)
Curtis Blais (Secretary)

John Stevens (SAC Technical Activity Director)

Arjan Lemmers
David Ronnfeldt
Graham Shanks
Eugene Stoudenmire
Steven Weiss

The PDG would like to especially acknowledge those individuals that significantly contributed to the preparation of this product as follows:

PDG Drafting Group

Steven W. Reichenthal (Editor)

Jane Bachman
Curtis Blais
Paul Gustavson
John Jinkerson
Craig Lammers
Jeffrey Steinman

The PDG would like also to recognize the contributions of these other individuals, including current and former SISO members, who contributed to the development of this document.

Jeff Abbott
Rob Albright
Shelby Barrett
William Beavin
Paul Birkel
Jonathan Boan
Peter Bosch
Steve Boswell
Francis Bowers
Kevin Brandt
Pat Burgess
Cord Cardinal
Pablo Cases
Michel Charbonneau
Tram Chase
Guan Chaun
Bang Choon
Darrell Collins
Jeff Covelli
Daran Crush
Dannie Cutts
D Jacinto
Bradford Dillman
Arren Dorman
Arno Duvenhage
John Fay
Terrill Frantz
Masakazu Furuichi
Michael Gagliano
Juan Garcia
James Globe
Edward Gordon
Jean-Baptiste Guillerit
Paul Gustavson
Per Gustavsson
Kevin Heffner
Reinhard Herzog
Frank Hill
Jim Hollenbach
George Hughes
Theresa Hughes
Russell Hutt
Jae Jun Hwang
Bruce Jackson

Stephen Jones
Rajive Joshi
Anna Karphammer
Rosemarie Keener
Michel Keuning
Sam Knight
Jerry Knipfer
Lee Krause
Lee Lacy
Jay Levenson
Bryan Linkous
Kok Seng Low
Van Lowe
Paul Lowe
Marianela Garcia Lozano
Franklin Lue
Farid Mamaghani
Peter Mandl
Lee Marden
Edward McCall
Michael McGarity
Robert McGraw
The following individuals comprised the ballot group for this product.

**Ballot Group**

<table>
<thead>
<tr>
<th>Jane Bachman</th>
<th>Timothy Calderwood</th>
<th>Laurent Prignac</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curtis Blais</td>
<td>Robert Lutz</td>
<td>Marcy Stutzman</td>
</tr>
</tbody>
</table>

When the Standards Activity Committee approved this product on 9 February 2017, it had the following membership:

**Standards Activity Committee**

| Katherine L. Morse (Chair)                          |
| Jean-Louis Igarza (Vice Chair)                     |
| Lance Marrou (Secretary)                            |

<table>
<thead>
<tr>
<th>Brad Dillman</th>
<th>Patrice Le Leydour</th>
<th>John Stevens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kevin Gupton</td>
<td>Chris McGroarty</td>
<td>Simone Youngblood</td>
</tr>
<tr>
<td>Paul Gustavson</td>
<td>Angus McLean</td>
<td></td>
</tr>
<tr>
<td>Aerial Kreiner</td>
<td>William Oates</td>
<td></td>
</tr>
</tbody>
</table>

When the Executive Committee approved this product on 9 March 2017, it had the following membership:

**Executive Committee**

| Michael O’Connor (Chair)                              |
| Bob Lutz (Vice Chair)                                 |
| Jeff Abbott (Secretary)                               |

<table>
<thead>
<tr>
<th>Jim Coolahan</th>
<th>Lana McGlynn</th>
<th>Robert Siegfried</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Daly</td>
<td>Katherine L. Morse</td>
<td>Eric Whittington</td>
</tr>
<tr>
<td>John Diem</td>
<td>Roy Scrudder</td>
<td></td>
</tr>
</tbody>
</table>
Introduction

This standard specifies the structure and behavior of engines that can execute models represented in the Simulation Reference Markup Language (SRML).

SRML is an XML-based data representation that provides the modeling and simulation community with:

- A common mechanism for loading and executing simulations.
- The ability to create an executable simulation model that can be shared between various devices.
- A way to improve model consistency between federated simulations.
- The ability to reuse simulations throughout the commercial, educational, and governmental communities, and across simulation domains.
# Table of Contents

1 **OVERVIEW** .................................................................................................................. 26
  1.1 Scope .......................................................................................................................... 26
  1.2 Purpose ......................................................................................................................... 26
  1.3 Objectives ..................................................................................................................... 27
  1.4 Intended Audience ........................................................................................................ 27
  1.5 Document Conventions ............................................................................................... 27

2 **REFERENCES** .................................................................................................................... 28
  2.1 SISO References ........................................................................................................... 28
  2.2 Other References .......................................................................................................... 28

3 **DEFINITIONS, ACRONYMS, AND ABBREVIATIONS** ............................................ 29
  3.1 Definitions ..................................................................................................................... 29
  3.2 Acronyms and Abbreviations ....................................................................................... 31

4 **SRML ENGINE CONCEPTS** ....................................................................................... 32
  4.1 SRML Simulation Context ............................................................................................ 32
  4.2 SRML Run-time Environment ..................................................................................... 32
  4.3 Item Management .......................................................................................................... 32
  4.4 Event Management ....................................................................................................... 33
  4.5 Conceptual Meta-Model ............................................................................................... 34

5 **DOM BASE** .................................................................................................................... 36
  5.1 Item Objects from XML ............................................................................................... 36
  5.2 Dynamic DOM .............................................................................................................. 36

6 **SRML ENGINE OBJECT MODEL** ............................................................................... 37

7 **SIMULATOR INTERFACE** ............................................................................................. 38
  7.1 Simulation Construction Services ................................................................................. 38
    7.1.1 CreateSimulation method ....................................................................................... 38
      7.1.1.1 Supplied arguments ......................................................................................... 38
      7.1.1.2 Returned arguments ......................................................................................... 38
      7.1.1.3 Preconditions ................................................................................................... 38
      7.1.1.4 Postconditions ................................................................................................ 38
      7.1.1.5 Exceptions ...................................................................................................... 38

8 **ITEM INTERFACE** ........................................................................................................ 39
  8.1 Intrinsic Item Properties .............................................................................................. 39
    8.1.1 ItemID property ........................................................................................................ 39
    8.1.2 Object property ....................................................................................................... 39
8.3.3 Simulation property ......................................................... 39
8.3.2 Items property ............................................................... 40
8.3.1 Links property ............................................................... 40
8.2.10 EventSinks property ..................................................... 40
8.2.9 ItemClass property ....................................................... 40
8.2.8 Self property ............................................................... 40
8.2.7 Node property .............................................................. 40
8.2.6 Location property ......................................................... 40
8.2.5 LocationFixedIND .......................................................... 40
8.2.4 SourceURI ................................................................. 40
8.2.3 Quantity ................................................................. 40
8.2.2 TransformURI ............................................................. 40
8.2.1 ItemScopeID ............................................................... 40
8.1.10 ScopeClosedIND .......................................................... 40
8.1.9 LocalID ................................................................. 40
8.1.8 ItemName ................................................................. 40
8.1.7 EventSinks ............................................................... 40
8.1.6 Links ................................................................. 40
8.1.5 ID ................................................................. 40
8.1.4 ItemClass ............................................................... 40
8.1.3 Node ................................................................. 40
8.1.2 Links ................................................................. 40
8.1.1 ID ................................................................. 40
8.1.0 Item ................................................................. 40

8.2 Model–Defined Item Properties ........................................... 41
8.2.1 Properties for special data types ........................................ 41
8.2.2 SourceURI–typed property ............................................... 42
8.2.3 Quantity–typed property .................................................. 42
8.2.4 TransformURI–typed property .......................................... 42
8.2.5 ItemScopeID–typed property ........................................... 42
8.2.6 ScopeClosedIND–typed property ........................................ 43
8.2.7 LocalID–typed property .................................................. 43
8.2.8 ItemName–typed property ................................................ 43
8.2.9 LocationFixedIND–typed property ..................................... 44
8.2.10 ID–typed property ....................................................... 44

8.3 Item Retrieval Services ...................................................... 44
8.3.1 Item method ............................................................... 44
8.3.1.1 Supplied arguments .................................................. 45
8.3.1.2 Returned arguments .................................................. 45
8.3.1.3 Preconditions ......................................................... 45
8.3.1.4 Postconditions ....................................................... 45
8.3.1.5 Exceptions ........................................................... 45
8.3.2 GetItem method ........................................................... 45
8.3.2.1 Supplied arguments .................................................. 45
8.3.2.2 Returned arguments .................................................. 45
8.3.2.3 Preconditions ......................................................... 45
8.3.2.4 Postconditions ....................................................... 45
8.3.2.5 Exceptions ........................................................... 45
8.3.3 FindItem method .......................................................... 46
8.3.3.1 Supplied arguments .................................................. 46
8.3.3.2 Returned arguments .................................................. 46
8.3.3.3 Preconditions ................................................................. 46
8.3.3.4 Postconditions ............................................................... 46
8.3.3.5 Exceptions ................................................................. 46
8.3.4 FindItems method ............................................................ 46
  8.3.4.1 Supplied arguments ...................................................... 46
  8.3.4.2 Returned arguments .................................................... 46
  8.3.4.3 Preconditions ............................................................. 46
  8.3.4.4 Postconditions ........................................................... 46
  8.3.4.5 Exceptions ............................................................... 47
8.4 Item Construction Services .................................................. 47
  8.4.1 CreateItem method .......................................................... 47
    8.4.1.1 Supplied arguments ................................................ 47
    8.4.1.2 Returned arguments ................................................. 47
    8.4.1.3 Preconditions ........................................................ 47
    8.4.1.4 Postconditions ....................................................... 47
    8.4.1.5 Exceptions ........................................................... 47
  8.4.2 CreateItems method ....................................................... 47
    8.4.2.1 Supplied arguments ................................................ 48
    8.4.2.2 Returned arguments ................................................. 48
    8.4.2.3 Preconditions ........................................................ 48
    8.4.2.4 Postconditions ....................................................... 48
    8.4.2.5 Exceptions ........................................................... 48
8.5 Event Services ................................................................. 48
  8.5.1 SendEvent method .......................................................... 48
    8.5.1.1 Supplied arguments ................................................ 48
    8.5.1.2 Returned arguments ................................................. 48
    8.5.1.3 Preconditions ........................................................ 48
    8.5.1.4 Postconditions ....................................................... 49
    8.5.1.5 Exceptions ........................................................... 49
  8.5.2 SendItemEvent method ................................................... 49
    8.5.2.1 Supplied arguments ................................................ 49
    8.5.2.2 Returned arguments ................................................. 49
    8.5.2.3 Preconditions ........................................................ 49
    8.5.2.4 Postconditions ....................................................... 49
    8.5.2.5 Exceptions ........................................................... 49
  8.5.3 PostEvent method .......................................................... 50
    8.5.3.1 Supplied arguments ................................................ 50
    8.5.3.2 Returned arguments ................................................. 50
    8.5.3.3 Preconditions ........................................................ 50
    8.5.3.4 Postconditions ....................................................... 50
    8.5.3.5 Exceptions ........................................................... 50
  8.5.4 PostItemEvent method .................................................... 50
    8.5.4.1 Supplied arguments ................................................ 50
    8.5.4.2 Returned arguments ................................................. 51
    8.5.4.3 Preconditions ........................................................ 51
8.5.4.4 Postconditions ................................................................. 51
8.5.4.5 Exceptions ................................................................. 51
8.5.5 ScheduleEvent method ................................................. 51
  8.5.5.1 Supplied arguments .................................................. 51
  8.5.5.2 Returned arguments ................................................ 51
  8.5.5.3 Preconditions ......................................................... 51
  8.5.5.4 Postconditions ....................................................... 51
  8.5.5.5 Exceptions ........................................................... 51
8.5.6 ScheduleItemEvent method ........................................... 52
  8.5.6.1 Supplied arguments .................................................. 52
  8.5.6.2 Returned arguments ................................................ 52
  8.5.6.3 Preconditions ......................................................... 52
  8.5.6.4 Postconditions ....................................................... 52
  8.5.6.5 Exceptions ........................................................... 52
8.5.7 CreateEvent method ..................................................... 52
  8.5.7.1 Supplied arguments .................................................. 53
  8.5.7.2 Returned arguments ................................................ 53
  8.5.7.3 Preconditions ......................................................... 53
  8.5.7.4 Postconditions ....................................................... 53
  8.5.7.5 Exceptions ........................................................... 53
8.5.8 BroadcastEvent method ................................................ 53
  8.5.8.1 Supplied arguments .................................................. 53
  8.5.8.2 Returned arguments ................................................ 54
  8.5.8.3 Postconditions ....................................................... 54
  8.5.8.4 Exceptions ........................................................... 54
8.5.9 ScheduleBroadcast method ........................................... 54
  8.5.9.1 Supplied arguments .................................................. 54
  8.5.9.2 Returned arguments ................................................ 54
  8.5.9.3 Preconditions ......................................................... 54
  8.5.9.4 Postconditions ....................................................... 54
  8.5.9.5 Exceptions ........................................................... 55
8.6 Model–Defined Item Behavior .......................................... 55

9 SIMULATION INTERFACE .................................................... 56
  9.1 Simulation Object Construction ...................................... 56
  9.2 Simulation Object Structure ........................................... 56
  9.3 Simulation Object States ................................................ 56
  9.4 Simulation Information Services ...................................... 57
    9.4.1 State property ........................................................ 57
    9.4.2 StateChange notification .......................................... 57
      9.4.2.1 Supplied arguments ............................................ 57
      9.4.2.2 Returned arguments .......................................... 58
      9.4.2.3 Preconditions .................................................. 58
      9.4.2.4 Postconditions ................................................ 58
      9.4.2.5 Exceptions ...................................................... 58
9.4.3 Document property .................................................................58
9.4.4 ItemCount property .................................................................58
9.4.5 ItemClasses property .................................................................58
9.4.6 ItemIDs method .................................................................58
  9.4.6.1 Supplied arguments ...............................................................58
  9.4.6.2 Returned arguments ...............................................................58
  9.4.6.3 Preconditions .................................................................58
  9.4.6.4 Postconditions .................................................................58
  9.4.6.5 Exceptions .................................................................59
9.5 Simulation Construction Services ................................................59
  9.5.1 Load method .................................................................59
    9.5.1.1 Supplied arguments ......................................................59
    9.5.1.2 Returned arguments ......................................................59
    9.5.1.3 Preconditions ...............................................................59
    9.5.1.4 Postconditions ...............................................................59
    9.5.1.5 Exceptions .................................................................59
  9.5.2 LoadXML method ..............................................................59
    9.5.2.1 Supplied arguments ......................................................59
    9.5.2.2 Returned arguments ......................................................60
    9.5.2.3 Preconditions ...............................................................60
    9.5.2.4 Postconditions ...............................................................60
    9.5.2.5 Exceptions .................................................................60
  9.5.3 LoadItems method ...............................................................60
    9.5.3.1 Supplied arguments ......................................................60
    9.5.3.2 Returned arguments ......................................................60
    9.5.3.3 Preconditions ...............................................................60
    9.5.3.4 Postconditions ...............................................................60
    9.5.3.5 Exceptions .................................................................60
  9.5.4 LoadItemSource notification ..................................................60
    9.5.4.1 Supplied arguments ......................................................61
    9.5.4.2 Returned arguments ......................................................61
    9.5.4.3 Preconditions ...............................................................61
    9.5.4.4 Postconditions ...............................................................61
    9.5.4.5 Exceptions .................................................................61
  9.5.5 DeleteItem method ..............................................................61
    9.5.5.1 Supplied arguments ......................................................61
    9.5.5.2 Returned arguments ......................................................61
    9.5.5.3 Preconditions ...............................................................61
    9.5.5.4 Postconditions ...............................................................61
    9.5.5.5 Exceptions .................................................................61
  9.5.6 DestroyItem method ............................................................62
    9.5.6.1 Supplied arguments ......................................................62
    9.5.6.2 Returned arguments ......................................................62
    9.5.6.3 Preconditions ...............................................................62
    9.5.6.4 Postconditions ...............................................................62
9.5.6.5 Exceptions ................................................................. 62
9.5.7 ReleaseAll method .......................................................... 62
  9.5.7.1 Supplied arguments .................................................. 62
  9.5.7.2 Returned arguments .................................................. 62
  9.5.7.3 Preconditions ......................................................... 62
  9.5.7.4 Postconditions ....................................................... 62
  9.5.7.5 Exceptions ............................................................ 62
9.5.8 UnloadItems method ..................................................... 62
  9.5.8.1 Supplied arguments ................................................. 63
  9.5.8.2 Returned arguments ............................................... 63
  9.5.8.3 Preconditions ....................................................... 63
  9.5.8.4 Postconditions ....................................................... 63
  9.5.8.5 Exceptions ............................................................ 63
9.5.9 Save method ............................................................... 63
  9.5.9.1 Supplied arguments ................................................. 63
  9.5.9.2 Returned arguments ............................................... 63
  9.5.9.3 Preconditions ....................................................... 63
  9.5.9.4 Postconditions ....................................................... 63
  9.5.9.5 Exceptions ............................................................ 63
9.5.10 ArgumentSerializer property ...................................... 63
  9.5.10.1 Supplied arguments .............................................. 64
  9.5.10.2 Returned arguments .............................................. 64
  9.5.10.3 Preconditions ....................................................... 64
  9.5.10.4 Postconditions ....................................................... 64
  9.5.10.5 Exceptions ............................................................ 64
9.5.11 Reset method ........................................................... 64
  9.5.11.1 Supplied arguments .............................................. 64
  9.5.11.2 Returned arguments .............................................. 64
  9.5.11.3 Preconditions ....................................................... 64
  9.5.11.4 Postconditions ....................................................... 64
  9.5.11.5 Exceptions ............................................................ 64
9.5.12 Idle notification ....................................................... 64
  9.5.12.1 Supplied arguments .............................................. 64
  9.5.12.2 Returned arguments .............................................. 65
  9.5.12.3 Preconditions ....................................................... 65
  9.5.12.4 Postconditions ....................................................... 65
  9.5.12.5 Exceptions ............................................................ 65
9.5.13 Error property .......................................................... 65
9.5.14 IsItemLocationFixed method ...................................... 65
  9.5.14.1 Supplied arguments .............................................. 65
  9.5.14.2 Returned arguments .............................................. 65
  9.5.14.3 Preconditions ....................................................... 65
  9.5.14.4 Postconditions ....................................................... 65
  9.5.14.5 Exceptions ............................................................ 65
9.5.15 SetItemLocation method ............................................ 65
  9.5.15.1 Supplied arguments .............................................. 65
9.5.15.2 Returned arguments .................................................. 66
9.5.15.3 Preconditions ......................................................... 66
9.5.15.4 Postconditions ....................................................... 66
9.5.15.5 Exceptions ............................................................ 66

9.6 Item Monitoring Services .................................................. 66

9.6.1 ItemCreated notification ............................................... 66
  9.6.1.1 Supplied arguments ............................................. 66
  9.6.1.2 Returned arguments ............................................. 66
  9.6.1.3 Preconditions ...................................................... 66
  9.6.1.4 Postconditions .................................................... 66
  9.6.1.5 Exceptions ........................................................ 66

9.6.2 ItemDestroyed notification .......................................... 66
  9.6.2.1 Supplied arguments ............................................. 66
  9.6.2.2 Returned arguments ............................................. 67
  9.6.2.3 Preconditions ...................................................... 67
  9.6.2.4 Postconditions .................................................... 67
  9.6.2.5 Exceptions ........................................................ 67

9.6.3 ItemLocationChanged notification .................................. 67
  9.6.3.1 Supplied arguments ............................................. 67
  9.6.3.2 Returned arguments ............................................. 67
  9.6.3.3 Preconditions ...................................................... 67
  9.6.3.4 Postconditions .................................................... 67
  9.6.3.5 Exceptions ........................................................ 67

9.6.4 ItemPropertyChanged notification .................................. 67
  9.6.4.1 Supplied arguments ............................................. 67
  9.6.4.2 Returned arguments ............................................. 68
  9.6.4.3 Preconditions ...................................................... 68
  9.6.4.4 Postconditions .................................................... 68
  9.6.4.5 Exceptions ........................................................ 68

9.6.5 BeforeItemDeleted notification .................................... 68
  9.6.5.1 Supplied arguments ............................................. 68
  9.6.5.2 Returned arguments ............................................. 68
  9.6.5.3 Preconditions ...................................................... 68
  9.6.5.4 Postconditions .................................................... 68
  9.6.5.5 Exceptions ........................................................ 68

9.7 Item Linking Services .................................................... 68

9.7.1 AddLink method .......................................................... 68
  9.7.1.1 Supplied arguments ............................................. 69
  9.7.1.2 Returned arguments ............................................. 69
  9.7.1.3 Preconditions ...................................................... 69
  9.7.1.4 Postconditions .................................................... 69
  9.7.1.5 Exceptions ........................................................ 69

9.7.2 GetLinkNode method ................................................... 69
  9.7.2.1 Supplied arguments ............................................. 69
  9.7.2.2 Returned arguments ............................................. 69
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.7.2.3</td>
<td>Preconditions</td>
<td>69</td>
</tr>
<tr>
<td>9.7.2.4</td>
<td>Postconditions</td>
<td>69</td>
</tr>
<tr>
<td>9.7.2.5</td>
<td>Exceptions</td>
<td>69</td>
</tr>
<tr>
<td>9.7.3</td>
<td>RemoveLink method</td>
<td>70</td>
</tr>
<tr>
<td>9.7.3.1</td>
<td>Supplied arguments</td>
<td>70</td>
</tr>
<tr>
<td>9.7.3.2</td>
<td>Returned arguments</td>
<td>70</td>
</tr>
<tr>
<td>9.7.3.3</td>
<td>Preconditions</td>
<td>70</td>
</tr>
<tr>
<td>9.7.3.4</td>
<td>Postconditions</td>
<td>70</td>
</tr>
<tr>
<td>9.7.3.5</td>
<td>Exceptions</td>
<td>70</td>
</tr>
<tr>
<td>9.8</td>
<td>Item Dynamics Services</td>
<td>70</td>
</tr>
<tr>
<td>9.8.1</td>
<td>InvokeMethod method</td>
<td>70</td>
</tr>
<tr>
<td>9.8.1.1</td>
<td>Supplied arguments</td>
<td>70</td>
</tr>
<tr>
<td>9.8.1.2</td>
<td>Returned arguments</td>
<td>70</td>
</tr>
<tr>
<td>9.8.1.3</td>
<td>Preconditions</td>
<td>70</td>
</tr>
<tr>
<td>9.8.1.4</td>
<td>Postconditions</td>
<td>70</td>
</tr>
<tr>
<td>9.8.1.5</td>
<td>Exceptions</td>
<td>70</td>
</tr>
<tr>
<td>9.8.2</td>
<td>MemberExists method</td>
<td>71</td>
</tr>
<tr>
<td>9.8.2.1</td>
<td>Supplied arguments</td>
<td>71</td>
</tr>
<tr>
<td>9.8.2.2</td>
<td>Returned arguments</td>
<td>71</td>
</tr>
<tr>
<td>9.8.2.3</td>
<td>Preconditions</td>
<td>71</td>
</tr>
<tr>
<td>9.8.2.4</td>
<td>Postconditions</td>
<td>71</td>
</tr>
<tr>
<td>9.8.2.5</td>
<td>Exceptions</td>
<td>71</td>
</tr>
<tr>
<td>9.8.3</td>
<td>AddProperty method</td>
<td>71</td>
</tr>
<tr>
<td>9.8.3.1</td>
<td>Supplied arguments</td>
<td>71</td>
</tr>
<tr>
<td>9.8.3.2</td>
<td>Returned arguments</td>
<td>71</td>
</tr>
<tr>
<td>9.8.3.3</td>
<td>Preconditions</td>
<td>71</td>
</tr>
<tr>
<td>9.8.3.4</td>
<td>Postconditions</td>
<td>71</td>
</tr>
<tr>
<td>9.8.3.5</td>
<td>Exceptions</td>
<td>72</td>
</tr>
<tr>
<td>9.8.4</td>
<td>RemoveMember method</td>
<td>72</td>
</tr>
<tr>
<td>9.8.4.1</td>
<td>Supplied arguments</td>
<td>72</td>
</tr>
<tr>
<td>9.8.4.2</td>
<td>Returned arguments</td>
<td>72</td>
</tr>
<tr>
<td>9.8.4.3</td>
<td>Preconditions</td>
<td>72</td>
</tr>
<tr>
<td>9.8.4.4</td>
<td>Postconditions</td>
<td>72</td>
</tr>
<tr>
<td>9.8.4.5</td>
<td>Exceptions</td>
<td>72</td>
</tr>
<tr>
<td>9.9</td>
<td>Runtime Services</td>
<td>72</td>
</tr>
<tr>
<td>9.9.1</td>
<td>AddExtension method</td>
<td>72</td>
</tr>
<tr>
<td>9.9.1.1</td>
<td>Supplied arguments</td>
<td>72</td>
</tr>
<tr>
<td>9.9.1.2</td>
<td>Returned arguments</td>
<td>72</td>
</tr>
<tr>
<td>9.9.1.3</td>
<td>Preconditions</td>
<td>72</td>
</tr>
<tr>
<td>9.9.1.4</td>
<td>Postconditions</td>
<td>73</td>
</tr>
<tr>
<td>9.9.1.5</td>
<td>Exceptions</td>
<td>73</td>
</tr>
<tr>
<td>9.9.2</td>
<td>RemoveExtension method</td>
<td>73</td>
</tr>
<tr>
<td>9.9.2.1</td>
<td>Supplied arguments</td>
<td>73</td>
</tr>
<tr>
<td>9.9.2.2</td>
<td>Returned arguments</td>
<td>73</td>
</tr>
<tr>
<td>9.9.2.3</td>
<td>Preconditions</td>
<td>73</td>
</tr>
</tbody>
</table>
9.9.2.4 Postconditions.................................................................73
9.9.2.5 Exceptions .................................................................73
9.9.3 ClearExtensions method ....................................................73
  9.9.3.1 Supplied arguments....................................................73
  9.9.3.2 Returned arguments....................................................73
  9.9.3.3 Preconditions ...........................................................73
  9.9.3.4 Postconditions ..........................................................74
  9.9.3.5 Exceptions ...............................................................74
9.10 Time Services ....................................................................74
  9.10.1 CurrentTime property ....................................................74
    9.10.1.1 Supplied arguments ................................................74
    9.10.1.2 Returned arguments ...............................................74
    9.10.1.3 Preconditions .........................................................74
    9.10.1.4 Postconditions .......................................................74
    9.10.1.5 Exceptions .............................................................74
  9.10.2 EndTime property ........................................................74
    9.10.2.1 Supplied arguments ................................................74
    9.10.2.2 Returned arguments ...............................................74
    9.10.2.3 Preconditions .........................................................74
    9.10.2.4 Postconditions .......................................................74
    9.10.2.5 Exceptions .............................................................75
  9.10.3 CurrentTimeSet notification ..........................................75
    9.10.3.1 Supplied arguments ................................................75
    9.10.3.2 Returned arguments ...............................................75
    9.10.3.3 Preconditions .........................................................75
    9.10.3.4 Postconditions .......................................................75
    9.10.3.5 Exceptions .............................................................75
9.11 Event Definition Services ................................................75
  9.11.1 EventClasses property ................................................75
    9.11.1.1 Supplied arguments ................................................75
    9.11.1.2 Returned arguments ...............................................75
    9.11.1.3 Preconditions .........................................................75
    9.11.1.4 Postconditions .......................................................75
    9.11.1.5 Exceptions .............................................................76
  9.11.2 SetEventClassPriority method .......................................76
    9.11.2.1 Supplied arguments ................................................76
    9.11.2.2 Returned arguments ...............................................76
    9.11.2.3 Preconditions .........................................................76
    9.11.2.4 Postconditions .......................................................76
    9.11.2.5 Exceptions .............................................................76
  9.11.3 RetractEvent method ....................................................76
    9.11.3.1 Supplied arguments ................................................76
    9.11.3.2 Returned arguments ...............................................76
    9.11.3.3 Preconditions .........................................................76
    9.11.3.4 Postconditions .......................................................76
9.11.3.5 Exceptions ................................................................. 76
9.11.4 RetractAllEvents method ........................................... 77
  9.11.4.1 Supplied arguments .............................................. 77
  9.11.4.2 Returned arguments ............................................. 77
  9.11.4.3 Preconditions .................................................... 77
  9.11.4.4 Postconditions .................................................. 77
  9.11.4.5 Exceptions ....................................................... 77
9.12 Event Information Services ........................................... 77
  9.12.1 EventCount property .............................................. 77
    9.12.1.1 Supplied arguments ......................................... 77
    9.12.1.2 Returned arguments ........................................ 77
    9.12.1.3 Preconditions ................................................ 77
    9.12.1.4 Postconditions .............................................. 77
    9.12.1.5 Exceptions .................................................... 77
  9.12.2 GetEvent method ................................................ 78
    9.12.2.1 Supplied arguments ......................................... 78
    9.12.2.2 Returned arguments ........................................ 78
    9.12.2.3 Preconditions ................................................ 78
    9.12.2.4 Postconditions .............................................. 78
    9.12.2.5 Exceptions .................................................... 78
  9.12.3 PeekEvent method ................................................. 78
    9.12.3.1 Supplied arguments ......................................... 78
    9.12.3.2 Returned arguments ........................................ 78
    9.12.3.3 Preconditions ................................................ 78
    9.12.3.4 Postconditions .............................................. 78
    9.12.3.5 Exceptions .................................................... 78
9.13 Event Execution Services ............................................ 79
  9.13.1 DoNextEvent method .............................................. 79
    9.13.1.1 Supplied arguments ......................................... 79
    9.13.1.2 Returned arguments ........................................ 79
    9.13.1.3 Preconditions ................................................ 79
    9.13.1.4 Postconditions .............................................. 79
    9.13.1.5 Exceptions .................................................... 79
  9.13.2 Run method ........................................................ 79
    9.13.2.1 Supplied arguments ......................................... 79
    9.13.2.2 Returned arguments ........................................ 79
    9.13.2.3 Preconditions ................................................ 79
    9.13.2.4 Postconditions .............................................. 79
    9.13.2.5 Exceptions .................................................... 80
  9.13.3 Abort method ..................................................... 80
    9.13.3.1 Supplied arguments ......................................... 80
    9.13.3.2 Returned arguments ........................................ 80
    9.13.3.3 Preconditions ................................................ 80
    9.13.3.4 Postconditions .............................................. 80
    9.13.3.5 Exceptions .................................................... 80
9.13.4 CurrentEvent property ................................................................. 80
9.13.5 ForwardEvent method ................................................................. 80
  9.13.5.1 Supplied arguments ............................................................. 80
  9.13.5.2 Returned arguments ............................................................ 80
  9.13.5.3 Preconditions ......................................................................... 81
  9.13.5.4 Postconditions ....................................................................... 81
  9.13.5.5 Exceptions ............................................................................. 81
9.14 Event Sink Services ....................................................................... 81
  9.14.1 AddEventSink method ............................................................... 81
    9.14.1.1 Supplied arguments ......................................................... 81
    9.14.1.2 Returned arguments .......................................................... 82
    9.14.1.3 Preconditions .................................................................. 82
    9.14.1.4 Postconditions ................................................................ 82
    9.14.1.5 Exceptions ..................................................................... 82
  9.14.2 GetEventSinkNode method ......................................................... 82
    9.14.2.1 Supplied arguments .......................................................... 82
    9.14.2.2 Returned arguments .......................................................... 82
    9.14.2.3 Preconditions .................................................................. 82
    9.14.2.4 Postconditions ................................................................ 82
    9.14.2.5 Exceptions ..................................................................... 82
  9.14.3 RemoveEventSink method ......................................................... 82
    9.14.3.1 Supplied arguments .......................................................... 82
    9.14.3.2 Returned arguments .......................................................... 82
    9.14.3.3 Preconditions .................................................................. 83
    9.14.3.4 Postconditions ................................................................ 83
    9.14.3.5 Exceptions ..................................................................... 83
9.15 Event Monitoring Services .......................................................... 83
  9.15.1 BeforeEvent notification ......................................................... 83
    9.15.1.1 Supplied arguments .......................................................... 83
    9.15.1.2 Returned arguments .......................................................... 83
    9.15.1.3 Preconditions .................................................................. 83
    9.15.1.4 Postconditions ................................................................ 83
    9.15.1.5 Exceptions ..................................................................... 83
  9.15.2 BeforeItemEvent notification ................................................... 83
    9.15.2.1 Supplied arguments .......................................................... 83
    9.15.2.2 Returned arguments .......................................................... 84
    9.15.2.3 Preconditions .................................................................. 84
    9.15.2.4 Postconditions ................................................................ 84
    9.15.2.5 Exceptions ..................................................................... 84
  9.15.3 EventSent notification ............................................................... 84
    9.15.3.1 Supplied arguments .......................................................... 84
    9.15.3.2 Returned arguments .......................................................... 84
    9.15.3.3 Preconditions .................................................................. 84
    9.15.3.4 Postconditions ................................................................ 84
    9.15.3.5 Exceptions ..................................................................... 84
9.15.4 ItemEventSent notification ................................................................. 84
  9.15.4.1 Supplied arguments .................................................................... 84
  9.15.4.2 Returned arguments ................................................................ 85
  9.15.4.3 Preconditions ........................................................................... 85
  9.15.4.4 Postconditions ........................................................................ 85
  9.15.4.5 Exceptions ............................................................................... 85
9.15.5 EventPosted notification ............................................................... 85
  9.15.5.1 Supplied arguments ................................................................ 85
  9.15.5.2 Returned arguments ................................................................ 85
  9.15.5.3 Preconditions .......................................................................... 85
  9.15.5.4 Postconditions ........................................................................ 85
  9.15.5.5 Exceptions ............................................................................... 85
9.15.6 EventScheduled notification ........................................................ 85
  9.15.6.1 Supplied arguments ................................................................ 85
  9.15.6.2 Returned arguments ................................................................ 86
  9.15.6.3 Preconditions .......................................................................... 86
  9.15.6.4 Postconditions ........................................................................ 86
  9.15.6.5 Exceptions ............................................................................... 86
9.15.7 EventRetracted notification .......................................................... 86
  9.15.7.1 Supplied arguments ................................................................ 86
  9.15.7.2 Returned arguments ................................................................ 86
  9.15.7.3 Preconditions .......................................................................... 86
  9.15.7.4 Postconditions ........................................................................ 86
  9.15.7.5 Exceptions ............................................................................... 86
9.15.8 EventOccurred notification .......................................................... 86
  9.15.8.1 Supplied arguments ................................................................ 86
  9.15.8.2 Returned arguments ................................................................ 87
  9.15.8.3 Preconditions .......................................................................... 87
  9.15.8.4 Postconditions ........................................................................ 87
  9.15.8.5 Exceptions ............................................................................... 87
9.15.9 ItemEventOccurred notification ................................................... 87
  9.15.9.1 Supplied arguments ................................................................ 87
  9.15.9.2 Returned arguments ................................................................ 87
  9.15.9.3 Preconditions .......................................................................... 87
  9.15.9.4 Postconditions ........................................................................ 87
  9.15.9.5 Exceptions ............................................................................... 87
9.16 Statistical Services ........................................................................... 87
  9.16.1 RandomSeed property ................................................................. 87
  9.16.2 Random method ........................................................................ 87
    9.16.2.1 Supplied arguments ............................................................... 88
    9.16.2.2 Returned arguments ............................................................... 88
    9.16.2.3 Preconditions ....................................................................... 88
    9.16.2.4 Postconditions ..................................................................... 88
    9.16.2.5 Exceptions .......................................................................... 88
  9.16.3 RandomizeSeed method .............................................................. 88
9.16.3.1 Supplied arguments ........................................................................................................ 88
9.16.3.2 Returned arguments ....................................................................................................... 88
9.16.3.3 Preconditions ................................................................................................................ 88
9.16.3.4 Postconditions ............................................................................................................. 88
9.16.3.5 Exceptions .................................................................................................................. 88

9.16.4 GetRandomNumberGenerator method ........................................................................... 88
9.16.4.1 Supplied arguments ....................................................................................................... 89
9.16.4.2 Returned arguments ..................................................................................................... 89
9.16.4.3 Preconditions ................................................................................................................ 89
9.16.4.4 Postconditions ............................................................................................................. 89
9.16.4.5 Exceptions .................................................................................................................. 89

9.16.5 LoadRandomNumberGenerator notification ................................................................... 89
9.16.5.1 Supplied arguments ....................................................................................................... 89
9.16.5.2 Returned arguments ..................................................................................................... 89
9.16.5.3 Preconditions ................................................................................................................ 89
9.16.5.4 Postconditions ............................................................................................................. 89
9.16.5.5 Exceptions .................................................................................................................. 90

9.16.6 CreateRandomUniform method .................................................................................... 90
9.16.6.1 Supplied arguments ....................................................................................................... 91
9.16.6.2 Returned arguments ..................................................................................................... 91
9.16.6.3 Preconditions ................................................................................................................ 91
9.16.6.4 Postconditions ............................................................................................................. 91
9.16.6.5 Exceptions .................................................................................................................. 91

9.16.7 CreateRandomExponential method ............................................................................... 91
9.16.7.1 Supplied arguments ....................................................................................................... 92
9.16.7.2 Returned arguments ..................................................................................................... 92
9.16.7.3 Preconditions ................................................................................................................ 92
9.16.7.4 Postconditions ............................................................................................................. 92
9.16.7.5 Exceptions .................................................................................................................. 92

9.16.8 CreateRandomTriangular method ............................................................................... 92
9.16.8.1 Supplied arguments ....................................................................................................... 93
9.16.8.2 Returned arguments ..................................................................................................... 93
9.16.8.3 Preconditions ................................................................................................................ 93
9.16.8.4 Postconditions ............................................................................................................. 93
9.16.8.5 Exceptions .................................................................................................................. 93

9.16.9 CreateRandomNormal method .................................................................................... 93
9.16.9.1 Supplied arguments ....................................................................................................... 94
9.16.9.2 Returned arguments ..................................................................................................... 94
9.16.9.3 Preconditions ................................................................................................................ 94
9.16.9.4 Postconditions ............................................................................................................. 94
9.16.9.5 Exceptions .................................................................................................................. 94

9.16.10 CreateTimeAverage method ....................................................................................... 94
9.16.10.1 Supplied arguments .................................................................................................... 94
9.16.10.2 Returned arguments .................................................................................................. 94
9.16.10.3 Preconditions ............................................................................................................. 94
9.16.10.4 Postconditions .......................................................................................................... 94

Copyright © 2017 SISO. All rights reserved
This is an approved SISO standard.
9.16.10.5 Exceptions ................................................................................................. 94
9.16.11 CreateBasicStatistics method................................................................. 94
  9.16.11.1 Supplied arguments ............................................................................. 95
  9.16.11.2 Returned arguments ........................................................................... 95
  9.16.11.3 Preconditions ...................................................................................... 95
  9.16.11.4 Postconditions .................................................................................... 95
  9.16.11.5 Exceptions ......................................................................................... 95
9.17 Support Services ......................................................................................... 95
  9.17.1 CreateArray method ............................................................................... 95
    9.17.1.1 Supplied arguments ......................................................................... 95
    9.17.1.2 Returned arguments ....................................................................... 95
    9.17.1.3 Preconditions ................................................................................... 95
    9.17.1.4 Postconditions ................................................................................. 95
    9.17.1.5 Exceptions ....................................................................................... 95
  9.17.2 CreateCollection method ...................................................................... 95
    9.17.2.1 Supplied arguments ........................................................................ 96
    9.17.2.2 Returned arguments ....................................................................... 96
    9.17.2.3 Preconditions .................................................................................. 96
    9.17.2.4 Postconditions ................................................................................. 96
    9.17.2.5 Exceptions ....................................................................................... 96

10 ITEM EVENT INTERFACE .................................................................................. 97
  10.1 EventID property ...................................................................................... 97
  10.2 Name property ......................................................................................... 97
  10.3 Time property .......................................................................................... 97
  10.4 Priority property ...................................................................................... 97
  10.5 Format property ...................................................................................... 97
  10.6 Cascade property ..................................................................................... 97
  10.7 Source property ....................................................................................... 97
  10.8 Target property ......................................................................................... 97
  10.9 Cancel property ....................................................................................... 98
  10.10 Arguments zproperty ............................................................................. 98

11 ITEMS COLLECTION INTERFACE .................................................................. 99
  11.1 Collection Management Services ............................................................ 99
    11.1.1 Item method ....................................................................................... 99
      11.1.1.1 Supplied arguments ..................................................................... 99
      11.1.1.2 Returned arguments ................................................................... 99
      11.1.1.3 Preconditions .............................................................................. 99
      11.1.1.4 Postconditions ............................................................................ 99
      11.1.1.5 Exceptions ................................................................................... 99
    11.1.2 Count property ................................................................................... 99
    11.1.3 Exists method .................................................................................... 99
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1.3.1</td>
<td>Supplied arguments</td>
</tr>
<tr>
<td>11.1.3.2</td>
<td>Returned arguments</td>
</tr>
<tr>
<td>11.1.3.3</td>
<td>Preconditions</td>
</tr>
<tr>
<td>11.1.3.4</td>
<td>Postconditions</td>
</tr>
<tr>
<td>11.1.3.5</td>
<td>Exceptions</td>
</tr>
<tr>
<td>13.1.4</td>
<td>IndexOf method</td>
</tr>
</tbody>
</table>

### 12 ITEM EVENTS COLLECTION INTERFACE

12.1 Collection Management Services

12.1.1 Item method

12.1.1.1 Supplied arguments
12.1.1.2 Returned arguments
12.1.1.3 Preconditions
12.1.1.4 Postconditions
12.1.1.5 Exceptions

12.1.2 Count property

### 13 COLLECTION INTERFACE

13.1 Collection Management Services

13.1.1 DuplicateKeyAction property

13.1.2 Item method

13.1.2.1 Supplied arguments
13.1.2.2 Returned arguments
13.1.2.3 Preconditions
13.1.2.4 Postconditions
13.1.2.5 Exceptions

13.1.3 GetItem method

13.1.3.1 Supplied arguments
13.1.3.2 Returned arguments
13.1.3.3 Preconditions
13.1.3.4 Postconditions
13.1.3.5 Exceptions

13.1.4 Count property

13.1.5 Exists method

13.1.5.1 Supplied arguments
13.1.5.2 Returned arguments
13.1.5.3 Preconditions
13.1.5.4 Postconditions
13.1.5.5 Exceptions

13.1.6 IndexOf method

13.1.6.1 Supplied arguments
14 ARRAY INTERFACE

14.1 Array Management Services

14.1.1 Item method

14.1.1.1 Supplied arguments

14.1.1.2 Returned arguments

14.1.1.3 Preconditions

14.1.1.4 Postconditions

14.1.1.5 Exceptions

14.1.2 Count property

14.1.3 Add method

14.1.3.1 Supplied arguments

14.1.3.2 Returned arguments

14.1.3.3 Preconditions

14.1.3.4 Postconditions

14.1.3.5 Exceptions

14.1.4 Insert method

14.1.4.1 Supplied arguments
16 BASIC STATISTICS INTERFACE .................................................. 111

16.1 Basic Statistics Services ...................................................... 111
   16.1.1 AddValue method ...................................................... 111
       16.1.1.1 Supplied arguments ........................................... 111
       16.1.1.2 Returned arguments ......................................... 111
       16.1.1.3 Preconditions ............................................... 111
       16.1.1.4 Postconditions .............................................. 111
       16.1.1.5 Exceptions .................................................. 111
   16.1.2 Mean property ....................................................... 111
   16.1.3 Variance property .................................................. 111
   16.1.4 StandardDeviation property .................................... 111
   16.1.5 Count property .................................................... 111
   16.1.6 Maximum property ............................................... 111
   16.1.7 Minimum property ................................................ 112
   16.1.8 Reset method ...................................................... 112
       16.1.8.1 Supplied arguments ........................................... 112
       16.1.8.2 Returned arguments ......................................... 112
       16.1.8.3 Preconditions ............................................... 112
       16.1.8.4 Postconditions .............................................. 112

15 RANDOM NUMBER GENERATOR INTERFACE.................................. 110

15.1 Random Number Generation Services .................................. 110
   15.1.1 RandomValue method .............................................. 110
       15.1.1.1 Supplied arguments ........................................... 110
       15.1.1.2 Returned arguments ......................................... 110
       15.1.1.3 Preconditions ............................................... 110
       15.1.1.4 Postconditions .............................................. 110
       15.1.1.5 Exceptions .................................................. 110
17 TIME AVERAGE INTERFACE .............................................................................113

17.1 Time Average Services .............................................................................113

17.1.1 AddValue method .............................................................................113

17.1.1.1 Supplied arguments ..................................................................113

17.1.1.2 Returned arguments ..................................................................113

17.1.1.3 Preconditions .........................................................................113

17.1.1.4 Postconditions .........................................................................113

17.1.1.5 Exceptions .............................................................................113

17.1.2 Initialize method .............................................................................113

17.1.2.1 Supplied arguments ..................................................................113

17.1.2.2 Returned arguments ..................................................................113

17.1.2.3 Preconditions .........................................................................113

17.1.2.4 Postconditions .........................................................................113

17.1.2.5 Exceptions .............................................................................114

17.1.3 LastValue property ..........................................................................114

17.1.4 CurrentTime property ......................................................................114

17.1.5 Average property .............................................................................114

17.1.6 Count property ................................................................................114

17.1.7 Maximum property ..........................................................................114

17.1.8 Reset method ..................................................................................114

17.1.8.1 Supplied arguments ..................................................................114

17.1.8.2 Returned arguments ..................................................................114

17.1.8.3 Preconditions .........................................................................114

17.1.8.4 Postconditions .........................................................................114

17.1.8.5 Exceptions .............................................................................114

17.1.9 Skip method ....................................................................................115

17.1.9.1 Supplied arguments ..................................................................115

17.1.9.2 Returned arguments ..................................................................115

17.1.9.3 Preconditions .........................................................................115

17.1.9.4 Postconditions .........................................................................115

17.1.9.5 Exceptions .............................................................................115

18 ERROR INTERFACE .....................................................................................116

18.1 Description Property .............................................................................116

18.2 XPath PROPERTY ..................................................................................116

18.3 Script Property ......................................................................................116

18.4 Number Property ................................................................................116

18.5 Line Property ......................................................................................116

18.6 Column Property ................................................................................116

18.7 Clear Method .........................................................................................116

18.7.1 Supplied arguments ........................................................................116
19 ARGUMENT SERIALIZER INTERFACE ......................................................... 118

19.1 Argument Serialization Services .................................................................. 118
  19.1.1 Load method ....................................................................................... 118
    19.1.1.1 Supplied arguments ................................................................. 118
    19.1.1.2 Returned arguments ............................................................... 118
    19.1.1.3 Preconditions .......................................................................... 118
    19.1.1.4 Postconditions ........................................................................ 118
    19.1.1.5 Exceptions ............................................................................... 118
  19.1.2 Save method ....................................................................................... 118
    19.1.2.1 Supplied arguments ................................................................. 118
    19.1.2.2 Returned arguments ............................................................... 118
    19.1.2.3 Preconditions .......................................................................... 119
    19.1.2.4 Postconditions ........................................................................ 119
    19.1.2.5 Exceptions ............................................................................... 119

ANNEX A – TABLE OF RANDOM NUMBER MULTIPLIERS .................................. 120

ANNEX B – SRML ENGINE IDL BINDING ......................................................... 121

TABLE OF FIGURES

Figure 1: Simulation Context (Informative) ............................................................ 32
Figure 2: SRML Item Management Conceptual Model (Informative) ....................... 33
Figure 3: SRML Event Model (Informative) ............................................................ 34
Figure 4: Conceptual Meta Model ...................................................................... 35
Figure 5: SRML Engine Object Model .................................................................. 37
Figure 6: Simulation Interface .......................................................................... 56
Figure 7: Simulation State Diagram (Informative) ............................................... 57

TABLE OF TABLES

Table 1: Intrinsic Item Properties ..................................................................... 39
Table 2: Item Properties for Special Data Types ................................................... 41

LISTINGS

Listing 1: SRML IDL Binding: ........................................................................... 121
1 OVERVIEW

This standard specifies the structure and behavior of engines that can execute models represented in the Simulation Reference Markup Language (SRML). SRML is an Extensible Markup Language (XML)-based data representation that is similar to HyperText Markup Language (HTML) in that it may be loaded and executed by end users using standardized software. Moreover, an SRML engine is analogous to a web browser in that it can load content containing executable behavior.

1.1 Scope

Three related documents comprise the SRML standard:

1. SISO-STD-009-00 [9] is the SRML specification that defines syntax and semantics of SRML markup for representing SRML models that can be executed in an SRML engine.
2. SISO-STD-009-01 (this document) is the SRML engine specification that defines the required operation of an SRML engine.

The primary intention for this document is to present a concise definition of an SRML engine, suitable for developing or using an SRML engine.

An SRML engine shall consist of software that implements the following capabilities:

1. Processing simulation structures from arbitrary XML documents, which may include markup from multiple schemas.
2. Processing behavioral scripts through language engines.
3. Constructing object structures from groups of objects in an XML–based compositional hierarchy.
4. Producing item class structures from an XML–based item class hierarchy.
5. Producing event class structures from an XML–based event class hierarchy.
6. Enabling the execution state of a simulation to be persisted and restored.
7. Producing quantities of individual items from a single object specification.
8. Establishing links between individual items.
9. Consuming global simulation control parameters.

Although an SRML engine provides features for processing scripts from arbitrary programming languages, an SRML engine is not required to directly process the syntax from any particular programming language. Therefore, the processing of languages such as JavaScript or other interpreted or compiled languages is beyond the scope of the SRML engine. However, SRML engine implementations do need to process the Script elements in SRML so as to enable language processing to occur as defined in this specification.

1.2 Purpose

This standard provides a specification that defines the capabilities provided by an SRML engine. The purpose of an SRML engine is to provide a software execution environment for simulations and other content to be served, received, and processed using web technologies. The engine specification promotes a common runtime environment, whereby an SRML model may operate regardless of product or vendor. To accomplish this, the SRML engine specification defines a generalized set of structural and behavioral constructs typically found in simulation products, while also providing a means for extension.
1.3 Objectives

The primary objective of this specification is to provide a clear description of an SRML engine that enables simulation developers and engine builders to produce, reuse, and integrate simulations across a broad range of tools and environments.

Additional goals are as follows:

1. Specify a flexible reference standard runtime environment for executing simulations.
2. Provide enough expressive power to load and operate most any XML for the purposes of simulation.
3. Choose constructs that are simple, yet expressive, so that models can be executed and maintained using minimal software.
4. Specify a sufficient set of features and provide the ability to extend.
5. Support downloadable model composition.
6. Utilize widely accepted standards such as DOM, and plug-in scripting languages, while not preventing lower-level languages from being used.
7. Define semantics that are neutral and flexible, to enable simulation software vendors to implement SRML engines.
8. Select constructs that minimize the ambiguity with reserved names found commonly in programming languages or in XML. For example the names: ‘class’, ‘event’, ‘entity’, ‘object’, and ‘attribute’ are avoided.
9. Provide software definition similar to a web browser, with individual object behavior and dynamic DOM functionality, yet targeted for simulations.
10. Provide object-oriented capabilities for representing generalization and containment relationships, as well as multiplicity.
11. Support interoperability with other applications and plug-in functionality.
12. Maintain device and presentation independence.
13. Define a system that can interoperate with (and simplify the development of) distributed simulation systems.

1.4 Intended Audience

The primary audience for this standard is the Modeling and Simulation (M&S) community; those who are interested in the modeling, interoperability, reusability, componentization, and composition of systems and simulations. Other communities with similar interests are encouraged to use this specification in their domains.

1.5 Document Conventions

The key words must, must not, required, shall, shall not, should, should not, recommended, may, and optional in this specification are to be interpreted as described in SISO–ADM–005–2011 [4].

In this document, the words define and declare refer to the use of SRML engine constructs, whereas the word specify refers to the definition of the SRML engine constructs themselves.
2 REFERENCES

2.1 SISO References

<table>
<thead>
<tr>
<th>#</th>
<th>Document Number</th>
<th>Title</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>SISO-ADM-001-2014</td>
<td>Policy for Numbering of SISO Products</td>
<td>15 May 2014</td>
</tr>
<tr>
<td>2.</td>
<td>SISO-ADM-002-2011</td>
<td>SISO Policies &amp; Procedures</td>
<td>11 Apr 2011</td>
</tr>
<tr>
<td>4.</td>
<td>SISO-ADM-005-2011</td>
<td>Policy for the Style and Format of SISO Documents</td>
<td>13 Jun 2011</td>
</tr>
<tr>
<td>7.</td>
<td>SISO-STD-003-2006</td>
<td>Standard for Base Object Model (BOM)</td>
<td>8 May 2006</td>
</tr>
</tbody>
</table>

2.2 Other References

<table>
<thead>
<tr>
<th>#</th>
<th>Document Number</th>
<th>Title</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.</td>
<td>W3C REC-xpath20-20070123</td>
<td>XML Path Language (XPath) 2.0</td>
<td>23 Jan 2007</td>
</tr>
<tr>
<td>12.</td>
<td>IETF RFC 3986</td>
<td>Uniform Resource Identifier (URI): Generic Syntax</td>
<td>2005</td>
</tr>
</tbody>
</table>
### 3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

English words are used in accordance with their definitions in the latest edition of Webster’s New Collegiate Dictionary except when special SISO Product-related technical terms are required.

#### 3.1 Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>A person who develops an SRML model.</td>
</tr>
<tr>
<td>Behavior</td>
<td>A set of operations performed by an item as actions or reactions to events.</td>
</tr>
<tr>
<td>Class definitions</td>
<td>A set of item class and/or event class definitions.</td>
</tr>
<tr>
<td>Document order</td>
<td>The order in which objects occur in the XML representation of a document [11]—corresponds to a preorder traversal of DOM nodes.</td>
</tr>
<tr>
<td>Domain</td>
<td>A field of interest or study for which a simulation is constructed, such as transportation, logistics, communication, marketing, physics, etc. A domain includes a specific vocabulary, for example the vocabulary for a transportation domain includes vehicle, cargo, passenger, and route.</td>
</tr>
<tr>
<td>Event class</td>
<td>A description of the common characteristics of all item events having the same name.</td>
</tr>
<tr>
<td>Event queue</td>
<td>An ordered list of events that have been scheduled or posted.</td>
</tr>
<tr>
<td>Fully composed interface</td>
<td>An interface that includes the union of members from a core interface and the members of the author–defined interface.</td>
</tr>
<tr>
<td>Host</td>
<td>Any software that serves as the environment for an SRML engine.</td>
</tr>
<tr>
<td>Interface</td>
<td>A set of members.</td>
</tr>
<tr>
<td>Item class</td>
<td>A named description of the common structure and behavior of all items to which it applies.</td>
</tr>
<tr>
<td>Item element</td>
<td>The XML element representation of an item object.</td>
</tr>
<tr>
<td>Item event</td>
<td>Information describing something that happens to an item at a point in time during the execution of simulation. Every item event has a name.</td>
</tr>
<tr>
<td>Item object</td>
<td>An object instance in the run–time environment of an SRML engine which provides the members of the Item interface.</td>
</tr>
<tr>
<td>Item scope</td>
<td>A location that provides a boundary whereby the items it contains may be uniquely identified.</td>
</tr>
<tr>
<td>Item</td>
<td>Any representation of the state of a particular object instance in a simulation in terms of its attribute values at a point in time. Item refers to an instance of an item object in the run–time environment of an SRML engine.</td>
</tr>
<tr>
<td>Item-derived element</td>
<td>Any XML element that an SRML engine uses to derive an item. Elements such as Item and Component are Item-derived objects, whereas Property, Script, Link, and EventSink are not.</td>
</tr>
<tr>
<td>Link</td>
<td>A property of an item that references another item. An item may have links to multiple other items.</td>
</tr>
<tr>
<td>Local identifier</td>
<td>A name used once within an item scope to identify an item.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Location</td>
<td>Refers to an item’s role when it serves as a place where other items reside. A location is derived from a DOM “parent node”. An item may be at only one location at a time, but may be transitively at many locations.</td>
</tr>
<tr>
<td>Member</td>
<td>A property, method, or notification defined for an item.</td>
</tr>
<tr>
<td>Method</td>
<td>An operation defined for an item. A method may have parameters and a return value. As a service, a method may be invoked to perform an action and/or return a result.</td>
</tr>
<tr>
<td>Model content</td>
<td>A portion of an SRML model that describes the items and their links at a point in time. Model content may be specified using constructs from the SRML namespace or in conjunction with markup from other namespaces.</td>
</tr>
<tr>
<td>Namespace qualification</td>
<td>The explicit association between a name and its namespace, by reference to the namespace’s URI.</td>
</tr>
<tr>
<td>Namespace</td>
<td>A collection of object, attribute, data type, and other names to be used in XML documents, and identified by a Uniform Resource Identifier (URI) [12].</td>
</tr>
<tr>
<td>Notification</td>
<td>A service which may invoke a method on a subscriber.</td>
</tr>
<tr>
<td>Object reference</td>
<td>A value that represents and provides access to an object. Scripts use object references to manipulate objects.</td>
</tr>
<tr>
<td>Object</td>
<td>The runtime instantiation of one or more interfaces.</td>
</tr>
<tr>
<td>Owner (owning item)</td>
<td>An item that includes a particular occurrence of a property, script, link, or event sink.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Defines the name and data type allowed when invoking an operation.</td>
</tr>
<tr>
<td>Property</td>
<td>A named feature or characteristic of an item’s overall state.</td>
</tr>
<tr>
<td>Property–deriving attribute</td>
<td>Any XML attribute that a simulator uses to derive a property.</td>
</tr>
<tr>
<td>Reference object</td>
<td>One of several objects in the SRML namespace that is intended to be used as a template for authors when defining the objects in their own namespaces.</td>
</tr>
<tr>
<td>Runtime environment</td>
<td>The defined structures and operations available and instantiated during the lifetime of an SRML simulation.</td>
</tr>
<tr>
<td>Script</td>
<td>Any text that describes executable instructions that an SRML engine uses to execute the behavior of an item.</td>
</tr>
<tr>
<td>Simulation object</td>
<td>In SRML, “Simulation object” is a specific term that refers to the top-level item in a simulation’s runtime environment. A Simulation object is represented as an item. All objects in the simulation are referred to as “items”.</td>
</tr>
<tr>
<td>Simulation structure</td>
<td>Refers to the organization or augmentation of model content for the purpose of running simulations.</td>
</tr>
<tr>
<td>SRML engine</td>
<td>Any software that is capable of loading and executing SRML models.</td>
</tr>
<tr>
<td>SRML model</td>
<td>An XML representation of a simulation that uses constructs from the SRML namespace mixed with model content from other namespaces to describe the structure and behavior of the items to be simulated. At runtime, refers to the executing representation that corresponds to the XML representation.</td>
</tr>
<tr>
<td>SRML simulation</td>
<td>The execution of one or more SRML models within an SRML engine.</td>
</tr>
<tr>
<td>Subscriber</td>
<td>An object that implements an outbound interface for receiving notifications.</td>
</tr>
<tr>
<td>Target</td>
<td>An item referenced by a link.</td>
</tr>
</tbody>
</table>
3.2 Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym or Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>A person who develops an SRML model</td>
</tr>
<tr>
<td>BOM</td>
<td>Base Object Model</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>EXCOM</td>
<td>Executive Committee</td>
</tr>
<tr>
<td>HLA</td>
<td>High Level Architecture</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>M&amp;S</td>
<td>Modeling and Simulation</td>
</tr>
<tr>
<td>MathML</td>
<td>Math Markup Language</td>
</tr>
<tr>
<td>PDG</td>
<td>Product Development Group</td>
</tr>
<tr>
<td>SAC</td>
<td>Standards Activity Committee</td>
</tr>
<tr>
<td>SISO</td>
<td>Simulation Interoperability Standards Organization</td>
</tr>
<tr>
<td>SRML</td>
<td>Simulation Reference Markup Language</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>XSD</td>
<td>XML Schema Definition</td>
</tr>
</tbody>
</table>
4 SRML ENGINE CONCEPTS

4.1 SRML Simulation Context

An SRML engine may be designed to operate in one or more specific environments. Figure 1 shows a simplified view of how an SRML engine would function within its environment. Here, a host application uses the services provided by the two–way interface of an SRML engine.

![Simulation Context Diagram]

4.2 SRML Run–time Environment

An SRML engine shall provide a run–time environment for SRML models which includes item and event management capability, random number generation, basic math and statistics such as an arithmetic average of a set of values, and extensibility in the form of external plug-in components and models.

An SRML engine shall provide a root “Simulation” object that encapsulates an individual execution (a run or replication) and is primarily responsible for providing the run–time environment.

Since a Simulation is an individual object that embodies a particular execution of a simulation pertaining to one or more models, multiple simultaneous simulations can exist on the same computer or be distributed across a network as a client, server, or peer.

4.3 Item Management

The “items” that comprise SRML models shall be represented as objects in the run-time environment. Each item shall have an interface that is generated from an XML element. Each item represents a unique instance. A correspondence shall exist between an item declared in the SRML model representation and an item that exists in the run–time environment.
Figure 2 illustrates these fundamental item management concepts and the relationships that exist among items, item classes, XML objects, and XML attributes.

XML attributes shall be used to generate an item’s properties. Properties may be defined directly on an item, or in a related item class. A data type defines the valid values that may be assigned to a property. Every item also has system-generated properties such as its ItemID property that contains its unique identifier, and its Node property that references its underlying object. Links are properties that connect items. Any item can be the owner or target or a link.

The behavior of an item shall be defined by operations placed within SRML Script elements, or by plug-in components, which are compiled objects that have interfaces usable by an SRML engine. Scripts may be defined directly on an item, or in a related item class. Each script has a “type”, which identifies the programming language in which the script is written.

An item may serve as a location for other items, and exist at a location along with other items. This location–item relationship shall be derived from the XML parent–child relationship.

An item may belong to an item class, which is analogous to a class in object-oriented terms, which provides the characteristics of its instances with an inheritance mechanism.

4.4 Event Management

An SRML engine shall use “item events” to facilitate communications among items during the execution of a simulation. An item event may be generated by an item at run–time, and carry temporal information to other items through its properties and arguments. Item events shall also be storable and retrievable before or after the execution of a simulation. SRML provides the ItemEvent object for this purpose.
Figure 3 depicts the event–oriented concepts in SRML. An item event’s properties shall include its name, and may include the source item that originated the item event, the target item that is the intended recipient of the item event, and the time in which the item event should occur.

For example, an item event describing a random collision between two “particle” items may be received by all items that have registered to receive collisions, hence no target and no time. The arguments of the item event might include the two items that collided. The name and characteristics of all “collision” item events can be defined in SRML with an “event class”.

An event class defines static characteristics that apply to all item events of the same name. Those characteristics include the script method invocation parameters to be mapped to item event arguments, the relative priority of an event class, and the identification of the super–classes of the event class. For example, a “collision” event class could identify a “coincidence” event class as its super–class. An event class’s priority characteristic helps the simulation engine decide which item event occurs first when two are scheduled simultaneously.

Item events shall be received by items through the invocation of methods defined in scripts. An item may include multiple “event sinks” that each subscribe to the ranges of event classes specified by an event sink’s properties. Event sinks may also subscribe to changes taking place on items, such as property changes, location changes, creation, and destruction.

4.5 Conceptual Meta-Model

The intention of this specification is to facilitate SRML engines to be developed for a variety of software environments and programming languages. With that in mind, the chosen vocabulary used in this specification is general, and may require mapping to specific applications. The following conceptual meta-model describes the assumptions and terminology used throughout the specification.

Objects implement named interfaces. A given object may implement several named interfaces. Interfaces provide related services in the form of properties, methods, and notifications. Services are grouped by a related purpose. Services have descriptions which specify their functions. Services may have public or private visibility, and may have named arguments. Properties are interface members that contain portions of an object’s state data, which may (or may not) be readable or writable. State data includes simple values such as numbers and strings, or composite values such as structures, object references, operation
references, and scripts. Unless otherwise specified, the default values of optional service arguments (parameters) shall imply the absence of data. Methods are inbound services (functions or procedures) that may alter the state of an object and/or simply calculate a value. Notifications are outbound services that an object may use to signal state changes or other communications with subscribing recipients. One member may be designated as a default member of an interface, which allows that member to be used without qualification. Some interfaces do not require implementation by any object instances.

![Figure 4: Conceptual Meta Model](image)

Programming language classes, modules, or other specific constructs are not part of this meta-model, but may be required for a particular implementation.
5 DOM BASE

SRML engine objects shall provide DOM interfaces.

5.1 Item Objects from XML

An SRML engine shall have the capability to load any well-formed XML.

An item object shall be created for every XML element, such that:

1. The item’s named properties match in name and data type each of the XML attributes.
2. The sub-items shall be connected to their locations for at all levels of nesting corresponding to the from parent-child XML elements.
3. Changes to items shall be reflected as changes in the DOM.

5.2 Dynamic DOM

Changes to the values of item properties shall be dynamically reflected in the item’s XML representation through the associated DOM objects.
6 SRML Engine Object Model

An SRML engine shall provide the primary interfaces: Item and Simulation. An SRML simulation shall be represented as a hierarchy of Item objects that corresponds to the structure of an SRML model. The top-level item in a simulation shall be a Simulation object.

![SRML Engine Object Model Diagram]

Figure 5: SRML Engine Object Model

An SRML engine shall provide the capability to instantiate Simulation objects. Simulation objects shall provide the capability to instantiate item objects. An item object shall be created from each item-derived element in the corresponding SRML model. Every item shall have an associated DOM node. A Simulation object’s DOM node shall be mapped to the root element only if the root element is a Simulation element. Otherwise the DOM node shall be the DOM Document node.
7 SIMULATOR INTERFACE

7.1 Simulation Construction Services

7.1.1 CreateSimulation method

The CreateSimulation method shall create and return a new simulation object. The returned simulation designator shall be the simulation object that has been created.

7.1.1.1 Supplied arguments

a) None

7.1.1.2 Returned arguments

a) Simulation designator.

7.1.1.3 Preconditions

a) None.

7.1.1.4 Postconditions

a) A new simulation object has been returned.

7.1.1.5 Exceptions

a) None.
8 ITEM INTERFACE

Item objects shall be created by a simulation from item-deriving elements in an SRML model. These include the following SRML elements: Simulations, Sim, Components, Comp, ItemClasses, ItemClass, EventClasses, EventClass, Item, and model-defined elements. Item objects may also be created dynamically.

An item’s interface shall be a combination of intrinsic and model-defined properties and methods.

8.1 Intrinsic Item Properties

Each item object shall have the intrinsic item properties as summarized in Table 1. Properties not listed as public may be public. Properties that are not public shall be visible to the scripts defined on the item.

<table>
<thead>
<tr>
<th>#</th>
<th>Property Name</th>
<th>Description</th>
<th>Visibility</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>ItemID</td>
<td>Engine–defined unique identifier.</td>
<td>Public</td>
<td>String containing an Unsigned Positive Integer</td>
</tr>
<tr>
<td>2.</td>
<td>Object</td>
<td>The item in its fully composed public interface.</td>
<td>Public</td>
<td>Composed Item</td>
</tr>
<tr>
<td>3.</td>
<td>Simulation</td>
<td>The item’s simulation owner.</td>
<td>Public</td>
<td>Composed Simulation</td>
</tr>
<tr>
<td>4.</td>
<td>Items</td>
<td>A collection of the item’s direct sub–items.</td>
<td>Public</td>
<td>Items Collection</td>
</tr>
<tr>
<td>5.</td>
<td>Links</td>
<td>Collection of the item’s links.</td>
<td>Links Collection</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>EventSinks</td>
<td>Collection of the item’s event sinks.</td>
<td>Event Sink Collection</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>ItemClass</td>
<td>The item’s ItemClass.</td>
<td>Composed Item Class</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Self</td>
<td>The item in its fully composed private interface.</td>
<td>Composed Item</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Node</td>
<td>The Item’s DOM Node.</td>
<td>DOM Node</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Location</td>
<td>The item’s location in the DOM, which itself is an item.</td>
<td>Composed Item</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>Script</td>
<td>The Item’s script management interface.</td>
<td>[Reserved]</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>ItemEvents</td>
<td>Collection of the item’s currently scheduled or posted events.</td>
<td>Public</td>
<td>Item events Collection</td>
</tr>
</tbody>
</table>

8.1.1 ItemID property

The ItemID property shall contain the item’s unique item identifier. This property shall be read–only. The item identifier shall be an unsigned non zero integer value which shall not change during the lifetime of the item, and shall not be re-used if an item is destroyed.

8.1.2 Object property

The Object property shall contain a self-reference to the item with its flattened interface. This property shall be read–only. The item designator shall be an object reference that remains valid for referencing the item during the item’s lifetime.

8.1.3 Simulation property

The Simulation property shall contain a reference to the simulation in which item exists. This property shall be read–only. The simulation designator shall be an object reference that remains valid for referencing the simulation during the time that the item exists in the simulation.
8.1.4 **Items property**

The Items property shall contain an Items collection containing the direct sub-items of a particular item. This property shall be read-only. The list of item designators shall be an Items collection that remains valid for accessing the sub-items during the lifetime of the item.

8.1.5 **Links property**

The Links property shall contain a Links collection containing the item’s links. This property shall be read-only. The returned list of item designators shall be a Collection that remains valid for accessing the links during the lifetime of the item.

8.1.6 **EventSinks property**

The EventSinks property shall contain a collection of the item’s event sinks. This property shall be read-only. The returned list of event sink designators shall be a Collection that remains valid for accessing the event sinks during the lifetime of the item.

8.1.7 **ItemClass property**

The ItemClass property shall contain a reference to the item class of the item. This property shall be read-only. The returned item class designator shall be an object reference that remains valid for referencing the item class during the time that the item exists in the simulation.

8.1.8 **Self property**

The Self property shall contain a reference to the item. This property shall be read-only. The returned item designator shall be an object reference that remains valid for referencing the item during the lifetime of the item.

8.1.9 **Node property**

The Node property shall contain a reference to the item’s DOM Node. This property shall be read-only. The DOM Node item designator shall be an object reference that remains valid for referencing the item’s DOM node during the lifetime of the item.

8.1.10 **Location property**

The Location property shall contain the item’s location in the simulation. This property shall be read-write. The location item designator shall be the object reference of the item for which this item is a sub-item. The location item designator shall not be null.

8.1.10.1 **Supplied arguments**

- a) Item designator.
- b) Location item designator.

8.1.10.2 **Returned arguments**

- a) None.

8.1.10.3 **Preconditions**

- a) None.
8.1.10.4 Postconditions

a) The item has been set to the designated location.

8.1.10.5 Exceptions

a) None.

8.1.11 Script property

The Script property shall contain a reference to the item’s script. This property shall be read–only. The Script designator shall be an object reference that remains valid for referencing the item’s script during the lifetime of the item. This member is reserved.

8.1.12 ItemEvents property

The ItemEvents property shall contain a collection of events that are currently scheduled or posted where the item is the target. This property shall be read–only. The list of item event designators shall be an ItemEvents collection that remains valid for accessing the item events during the lifetime of the item.

8.2 Model–Defined Item Properties

Each item object shall have a property derived from the XML representation of the model for:

1. Each attribute of the item’s corresponding element.
2. Each attribute of the prototype element of the corresponding item class.
3. Each property child element of the corresponding item class that has Attribute or Element placement.
4. Each Link child element.
5. Each Links child element.
6. Each child element that has a value for the Name attribute and a LocationFixed attribute set to true.
7. Each super–class of the item class in a left to right pre–order traversal, the same prototype, property, Link, and Links definitions as the primary item class which haven’t already been defined.

8.2.1 Properties for special data types

Table 2: Item Properties for Special Data Types

<table>
<thead>
<tr>
<th>Default property Name</th>
<th>Description</th>
<th>Data type</th>
<th>Property name is overridable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Source</td>
<td>External source.</td>
<td>SourceURI</td>
<td>Yes</td>
</tr>
<tr>
<td>2. Quantity</td>
<td>Replication count.</td>
<td>Quantity</td>
<td>Yes</td>
</tr>
<tr>
<td>3. Transform</td>
<td>XSLT.</td>
<td>TransformURI</td>
<td>Yes</td>
</tr>
<tr>
<td>4. ItemScope</td>
<td>Item scope identifier.</td>
<td>ItemScopeID</td>
<td>Yes</td>
</tr>
<tr>
<td>5. ScopeClosed</td>
<td>Scope closed indicator.</td>
<td>ScopeClosedIND</td>
<td>Yes</td>
</tr>
<tr>
<td>6. LocalID</td>
<td>The item’s local identifier.</td>
<td>LocalID</td>
<td>Yes</td>
</tr>
<tr>
<td>7. Name</td>
<td>Location–specific identifier.</td>
<td>ItemName</td>
<td>Yes</td>
</tr>
<tr>
<td>8. LocationFixed</td>
<td>Location fixed indicator.</td>
<td>LocationFixedIND</td>
<td>Yes</td>
</tr>
<tr>
<td>9. ID</td>
<td>Simulation–wide unique identifier.</td>
<td>xs:ID</td>
<td>Yes</td>
</tr>
</tbody>
</table>
8.2.2 SourceURI–typed property

An item object may have one SourceURI–typed property if authored in an SRML model. This property may neither be read nor written after the item has been constructed. The engine shall notify the host using LoadItemSource prior to attempting to create the item. If LoadItemSource returns an item, then the returned item shall be used. Otherwise, if a SourceURI resolves to a file, then the item’s object shall be merged with first matching object (in document order) in the file. If the SourceURI resolves to a class name available on the host system, then an instance of that class shall be instantiated as the item. After an object is loaded, the SourceURI–typed attribute shall be either removed from the corresponding DOM node or set to empty by the SRML engine when the value referenced a document. Otherwise the attribute value shall contain a value which allows the item’s document to be saved and re–loaded in an equivalent state.

The name of the SourceURI–typed property shall be derived from the from the Name attribute of the Property element in the nearest scope that has a Type attribute of “SourceURI”, if present. The name “Source” is the default SourceURI–typed property name.

8.2.3 Quantity–typed property

An item object may have one Quantity–typed property if authored in an SRML model. This property shall be read-only and shall always return the value (1) after an item has been constructed. The value of this property represents the post–condition of the following duplication process. While loading an item, if the value of the underlying Quantity–typed attribute is not blank, an SRML engine shall evaluate the expression contained in the value, duplicate the element to create enough siblings to obtain the specified quantity of items, and then shall set all of their Quantity property values to 1. After an object is loaded, the Quantity–typed attribute may either be removed from the corresponding DOM node, or set to empty by the SRML engine.

The name of the Quantity–typed property shall be derived from the Name attribute of the property element in the associated ItemClass that has the Type attribute set to “Quantity” and Placement set to “Instance”, if present. The name “Quantity” is the default Quantity–typed property name.

8.2.4 TransformURI–typed property

An item object may have one TransformURI–typed property if authored in an SRML model. This property may neither be read nor written after an item has been constructed. The value of this property represents the post–condition of the following operation. While loading an item, an SRML engine shall apply the XSLT transformation to the underlying object using the TransformURI value to locate the XSLT instructions. The transformation will take place prior to any other loading operation on the object. After an object is loaded, the TransformURI–typed attribute may be removed from the corresponding DOM node, or set to empty by the SRML engine.

The name of the TransformURI–typed property shall be derived from the Name attribute of the property object in the nearest scope that has a Type attribute of “TransformURI”, if present. The name “Transform” is the default TransformURI–typed property name.

8.2.5 ItemScopeID–typed property

An item object may have one ItemScopeID–typed property containing the string value that corresponds to the value supplied by the ItemScopeID–type attribute on the underlying element. This property shall be read–write after an item has been constructed. If such an attribute does not exist, the property shall initially contain an empty string. This property shall contain an item scope, which is a base location where nested items can be identified by their unique scope–wide local identifiers.

An SRML engine shall evaluate the value of the ItemScopeID–typed attribute as follows:

- “True” (not case sensitive) – an unnamed item scope shall be created for the item.
• ‘False’ (not case sensitive), or is empty, or not specified – an item scope shall not be created for the item.
• Any non-Boolean string value – a named item scope shall be created for the item. The name shall be a local identifier for the item within the outer item scope.

The name of the ItemScopeID–typed property shall be derived from the Name attribute of the property object in the nearest scope that has a Type attribute of “ItemScopeID”, if present. The name “ItemScope” is the default ItemScopeID–typed property name.

A global item scope shall be created at the root of a simulation.

8.2.6 ScopeClosedIND–typed property

An item object shall have one ScopeClosedIND–typed property containing the boolean value that corresponds to the value supplied by the ScopeClosedIND–type attribute on the underlying element. This property shall be read–write after an item has been constructed. If such an attribute does not exist, the property value shall be initially false. This property shall be used to set an item’s scope as either open or closed. The ScopeClosedIND–typed property with a value of true shall establish the item scope such that it does not include locally identified items from outer item scopes. Otherwise, the scope shall contain locally identified items from outer item scopes.

The name of the ScopeClosedIND–typed property shall be derived from the Name attribute of the property object in the nearest scope that has a Type attribute of “ScopeClosedIND”, if present. The name “ScopeClosed” is the default ScopeClosedIND–typed property name.

8.2.7 LocalID–typed property

An item object shall have one LocalID–typed property containing a string value that corresponds to the value supplied by the LocalID–type attribute on the underlying element. This property shall be read–write after an item has been constructed. If such an attribute does not exist, the property value shall initially be an empty string. When a value is set, this property shall be used to identify the item within the nearest enclosing item scope, which is either the nearest containing item with an enabled ItemScopeID–enabled property, or the global scope.

When loading an item, an SRML engine shall evaluate the name expression contained within the LocalID–typed attribute of the underlying object and produce the property value. Further, the SRML engine shall validate that the resulting property value conforms to the NDName specification. If two items with the same value in their LocalID–typed property exist in the same item scope, the engine shall favor the last one added (in document order, or execution order). An SRML engine shall enforce that an item does not have both a property of type ID, and a property of type LocalID.

The name of the LocalID–typed property shall be derived from the Name attribute of the property object in the nearest scope that has a Type attribute of “LocalID”, if present. The name “LocalID” is the default LocalID–typed property name.

8.2.8 ItemName–typed property

An item object shall have one ItemName–typed property containing a string value that corresponds to the value supplied by the ItemName–type attribute on the underlying element. This property shall be read–write after an item has been constructed. If such an attribute does not exist, the property value shall initially be an empty string. When a value is set, this property shall be used to identify the item within its location, which is the item derived from the parent object.

When loading an item, an SRML engine shall evaluate the name expression contained within the ItemName–typed attribute of the underlying object and produce the property value. Further, the SRML engine shall validate that the resulting property value conforms to the NDName specification.
If two items with the same value in their ItemName–typed property exist at the same location, the engine shall favor the last one added (in document order, or execution order).

The Name of the ItemName–typed property shall be derived from the ItemName attribute of the property object in the nearest scope that has a Type attribute of “ItemName”, if present. The name “Name” is the default ItemName–typed property name.

8.2.9 LocationFixedIND–typed property

An item object shall have one LocationFixedIND–typed property containing a boolean value that corresponds to the value supplied by the LocationFixedIND–type attribute on the underlying element. This property shall be read–write after an item has been constructed. If such an attribute does not exist, the property value shall initially be false. When the value is true, this property shall be used to fix an item’s location. If the item’s ItemName–typed property contains a value, an SRML engine shall establish a named property in the item’s enclosing item (location), named by the value in the ItemName–typed attribute. For example if "itemX" is location fixed, then it can be accessed through "location.ItemX".

The name of the LocationFixedIND–typed property shall be derived from the LocationFixedIND attribute of the property object in the nearest scope that has a Type attribute of “LocationFixedIND”. The name “LocationFixed” is the default LocationFixedIND–typed property name.

8.2.10 ID–typed property

An item object shall have one ID–typed property containing a string value that corresponds to the value supplied by the ID–type attribute on the underlying element. This property shall be read–write after an item has been constructed. If such an attribute does not exist, the property value shall initially be an empty string. When a value is set, this property shall be used to identify the item in the entire simulation, which is either the nearest containing item with an enabled ItemScopeID–enabled property, or the global scope.

When loading an item, an SRML engine shall evaluate the name expression contained within the ID–typed attribute of the underlying object and produce the property value. Further, the SRML engine shall validate that the resulting property value conforms to the NDName specification.

If two items with the same value in their ID–typed property exist in the same item scope, the engine shall favor the last one added (in document order, or execution order). An SRML engine shall enforce that an item does not have both a property of type ID, and a property of type LocalID.

An SRML engine shall derive the name of the ID–typed property from the Name attribute of the property object in the nearest scope that has a Type attribute of “ID”. The name “ID” is the default ID–typed property name.

8.3 Item Retrieval Services

8.3.1 Item method

The Item method shall return the item in scope at a specified index. The item index argument shall be either an index number or a name identifying a simulation item. The returned item designator shall be the returned item instance. If the index is a number, then this method shall return the item that has the unique ItemID that matches the index within the scope in which the item resides. Furthermore, if the index is a name, then this method shall return the item that has the LocalID typed property that matches the name. The name may be referenced in an inner item scope using dot (.) notation. If the item index argument does not resolve to an item, this property shall raise the exception: item not found. Where possible, this shall be implemented as the default member.
8.3.1.1 **Supplied arguments**
   a) Item index.

8.3.1.2 **Returned arguments**
   a) Item designator.

8.3.1.3 **Preconditions**
   a) None.

8.3.1.4 **Postconditions**
   a) An item designator has been returned.

8.3.1.5 **Exceptions**
   a) Item not found.

8.3.2 **GetItem method**

The GetItem method shall return the item in scope at a specified index. The item index argument shall be either an index number or a name identifying an item. The optional default object designator argument shall be an object to be returned if the supplied item designator argument does not identify an item in the simulation. The optional default object designator's implied value is a null reference. The returned item designator shall be the returned object reference. If the index is a number, then this method shall return the item that has the unique ItemID that matches the index within the scope in which the item resides. Furthermore, if the index is a name, then this method shall return the item that has the LocalID typed property that matches the name. The name may be referenced in an inner item scope using dot (.) notation. Where possible, this shall be implemented as the default member.

8.3.2.1 **Supplied arguments**
   a) Item index.
   b) Optional default object designator.

8.3.2.2 **Returned arguments**
   a) Item designator.

8.3.2.3 **Preconditions**
   a) None.

8.3.2.4 **Postconditions**
   a) An item or null reference has been returned.

8.3.2.5 **Exceptions**
   a) None.
8.3.3 FindItem method

The FindItem method shall return the first item found from a specified query. The XPath expression argument shall be an XPath query string. The optional directive to cache the query argument shall be if true, specifies that the query should be cached. Default is false. The returned item designator shall be the returned item instance. If no item is found at the specified location, this method shall return a null reference.

8.3.3.1 Supplied arguments

a) XPath expression.
   b) Directive to cache the query.

8.3.3.2 Returned arguments

a) Item designator.

8.3.3.3 Preconditions

a) None.

8.3.3.4 Postconditions

a) An item or null reference has been returned.

8.3.3.5 Exceptions

a) None.

8.3.4 FindItems method

The FindItems method shall return a collection of items found from a specified query. The XPath expression argument shall be an XPath query string. The optional directive to cache the query argument shall be if true, specifies that the query should be cached. Default is false. The returned list of item designators shall be a Collection containing the items that were found. If no items are found at the specified location, this method shall return a Collection containing no items.

8.3.4.1 Supplied arguments

a) XPath expression.
   b) Directive to cache the query.

8.3.4.2 Returned arguments

a) List of item designators.

8.3.4.3 Preconditions

a) None.

8.3.4.4 Postconditions

a) A list of item designators has been returned.
8.3.4.5 Exceptions

a) None.

8.4 Item Construction Services

8.4.1 CreateItem method

The CreateItem method shall create an item at a location and returns a reference to that item. The prototype item designator argument shall be the item or ItemClass that serves as a prototype for the new item. The optional location item designator argument shall be the location where the item should be created. The optional host provided data argument shall be context data available while creating the item. The before item designator argument shall be the index number or the name of a sub-item before which the item shall be placed. The after item designator argument shall be the index number or the name of a sub-item after which the item shall be placed. If both a before item designator and an after item designator is provided, this method shall raise the exception: Cannot simultaneously have both before and after arguments. The returned item designator shall be the item that has been created. If the engine does not have enough memory to create the item, this method shall raise the exception: Out of Memory.

8.4.1.1 Supplied arguments

a) Prototype item designator.
b) Location item designator.
c) Host provided data.
d) Before item designator.
e) After item designator.

8.4.1.2 Returned arguments

a) Item designator.

8.4.1.3 Preconditions

a) None.

8.4.1.4 Postconditions

a) A new item has been created at the specified location.
b) The new item has been returned.

8.4.1.5 Exceptions

a) Out of memory.
b) Cannot simultaneously have both before and after arguments.

8.4.2 CreateItems method

The CreateItems method shall create a specified number of items at a location and returns a collection that references those items. The prototype item designator argument shall be the item or ItemClass that serves as a prototype for the new item. The required quantity argument shall be the number of items to create. The optional location item designator argument shall be the location where the item should be created. The host provided data argument shall be context data available while creating the item. The before item designator argument shall be the index number or the name of a sub-item before which the item shall be placed. The after item designator argument shall be the index number or the name of a sub-item after which the item shall be placed. If both a before item designator and an after item designator is provided, this method shall raise the exception: Cannot simultaneously have both before and after arguments.
arguments. The returned list of item designators shall be the collection that has been created. If the engine does not have enough memory to create the item, this method shall raise the exception: Out of Memory.

8.4.2.1 Supplied arguments

a) Prototype item designator.
b) Required quantity.
c) Location item designator.
d) Host provided data.
e) Before item designator.
f) After item designator.

8.4.2.2 Returned arguments

a) List of item designators.

8.4.2.3 Preconditions

a) None.

8.4.2.4 Postconditions

a) The specified number of items has been created.
b) The list of item designators has been returned.

8.4.2.5 Exceptions

a) Out of memory.
b) Cannot simultaneously have both before and after arguments.

8.5 Event Services

An event is received when a simulation invokes a correspondingly named method on a target item.

8.5.1 SendEvent method

The SendEvent method shall synchronously send an event to a target item and return the result of the method underlying the event. The target item designator argument shall be the target item that will receive the event. All items will receive the event if the target item designator is a null reference. The event class designator argument shall identify the event to be sent. The optional variable argument list argument shall be a variable number of event argument values. The returned event invocation return value shall be the return value from the method underlying the event. If the item does not have a method corresponding to the event, no error occurs. The receiving method shall have a name that matches the event name, and an argument list that accepts the sequence of values in the variable argument list.

8.5.1.1 Supplied arguments

a) Target item designator.
b) Event class designator.
c) Variable argument list.

8.5.1.2 Returned arguments

a) Event invocation return value.
8.5.1.3  **Preconditions**

a) None.

8.5.1.4  **Postconditions**

a) The event has been sent.
b) The event invocation return value has been returned.

8.5.1.5  **Exceptions**

a) None.

8.5.2  **SendItemEvent method**

The SendItemEvent method shall synchronously send an item event to a target item and return the result from the last method to receive the event. The target item designator argument shall be the first item to receive the event. The event class designator argument shall identify the event to be sent. The cascade directive argument shall be one of the following values to specify how the event should cascade after it has been processed by the target.

- Do not cascade (default).
- Cascade up.
- Cascade down.

The optional variable argument list argument shall be a variable number of event argument values. The returned event invocation return value shall be the return value from the last method to receive the event. If the item does not have a method corresponding to the event, no error shall occur. The receiving method shall have a name that matches the event name, and an argument list that accepts a single item event argument. The item event provided to the receiving method shall have the source item designator set to the item that implements this method.

8.5.2.1  **Supplied arguments**

a) Target item designator.
b) Event class designator.
c) Cascade directive.
d) Variable argument list.

8.5.2.2  **Returned arguments**

a) Event invocation return value.

8.5.2.3  **Preconditions**

a) None.

8.5.2.4  **Postconditions**

a) The item event has been sent.
b) The event invocation return value has been returned.

8.5.2.5  **Exceptions**

a) None.
8.5.3 **PostEvent method**

The PostEvent method shall post an event to a target item, and return a retraction handle. The target item designator argument shall be the target item that will receive the event. All items will receive the event if the target item designator is a null reference. The event class designator argument shall identify the event to be sent. The optional variable argument list argument shall be a variable number of event argument values. The returned queued event handle shall be a handle that can be used with RetractEvent. If the item does not have a method corresponding to the event, no error shall occur. The receiving method shall have a name that matches the event name, and an argument list that accepts the sequence of values in the variable argument list.

8.5.3.1 **Supplied arguments**

a) Target item designator.
b) Event class designator.
c) Variable argument list.

8.5.3.2 **Returned arguments**

a) Queued event handle.

8.5.3.3 **Preconditions**

a) None.

8.5.3.4 **Postconditions**

a) The event has been posted.
b) The queued event handle has been returned.

8.5.3.5 **Exceptions**

a) None.

8.5.4 **PostItemEvent method**

The PostItemEvent method shall post an item event to a target item and return a retraction handle. The target item designator argument shall be the target item that will receive the event. All items will receive the event if the target item designator is a null reference. The event class designator argument shall identify the event to be sent. The cascade directive argument shall be one of the following values to specify how the event should cascade after it has been processed by the target.

- Do not cascade (default).
- Cascade up.
- Cascade down.

The optional variable argument list argument shall be a variable number of event argument values. The returned queued event handle shall be a handle that can be used with RetractEvent. If the item does not have a method corresponding to the event, no error shall occur. The receiving method shall have a name that matches the event name, and an argument list that accepts a single item event argument. The item event provided to the receiving method shall have the source item designator set to the item that implements this method.

8.5.4.1 **Supplied arguments**

a) Target item designator.
b) Event class designator.
c) Cascade directive.
d) Variable argument list.

8.5.4.2 Returned arguments

a) Queued event handle.

8.5.4.3 Preconditions

a) None.

8.5.4.4 Postconditions

a) The event has been posted.
b) The queued event handle has been returned.

8.5.4.5 Exceptions

a) None.

8.5.5 ScheduleEvent method

The ScheduleEvent method shall schedule an event to a target item, and return a retraction handle. The target item designator argument shall be the target item that will receive the event. All items will receive the event if the target item designator is a null reference. The event class designator argument shall identify the event to be sent. The scheduled time value argument shall be the date and time when the event is to occur. The optional variable argument list argument shall be a variable number of event argument values. The returned queued event handle shall be a handle that can be used with RetractEvent. If the item does not have a method corresponding to the event, no error shall occur. The receiving method shall have a name that matches the event name, and an argument list that accepts the sequence of values in the variable argument list.

8.5.5.1 Supplied arguments

a) Target item designator.
b) Event class designator.
c) Scheduled time value.
d) Variable argument list.

8.5.5.2 Returned arguments

a) Queued event handle.

8.5.5.3 Preconditions

a) None.

8.5.5.4 Postconditions

a) The event has been scheduled.
b) The queued event handle has been returned.

8.5.5.5 Exceptions

a) None.
8.5.6 ScheduleItemEvent method

The ScheduleItemEvent method shall schedule an item event to a target item and return a retraction handle. The target item designator argument shall be the target item that will receive the event. All items will receive the event if the target item designator is a null reference. The event class designator argument shall identify the event to be sent. The cascade directive argument shall be one of the following values to specify how the event should cascade after it has been processed by the target.

- Do not cascade (default).
- Cascade up.
- Cascade down.

The scheduled time value argument shall be the date and time when the event is to occur. The optional variable argument list argument shall be a variable number of event argument values. The returned queued event handle shall be a handle that can be used with RetractEvent. If the item does not have a method corresponding to the event, no error shall occur. The receiving method shall have a name that matches the event name, and an argument list that accepts a single item event. The item event provided to the receiving method shall have the source item designator set to the item that implements this method.

8.5.6.1 Supplied arguments

a) Target item designator.
b) Event class designator.
c) Cascade directive.
d) Scheduled time value.
e) Variable argument list.

8.5.6.2 Returned arguments

a) Queued event handle.

8.5.6.3 Preconditions

a) None.

8.5.6.4 Postconditions

a) The event has been scheduled.
b) The queued event handle has been returned.

8.5.6.5 Exceptions

a) None.

8.5.7 CreateEvent method

The CreateEvent method shall create and return a new ItemEvent instance. The event class designator argument shall be the name or instance of the event class to be created. The optional variable argument list argument shall be a variable number of event argument values. The returned item event designator shall be the ItemEvent that has been created. The event class designator argument may be the name of an event class. The returned item event shall have the source item designator set to the item that implements this method. If the engine does not have enough memory to create the event, this method shall raise the exception: Out of Memory.
8.5.7.1 **Supplied arguments**
   a) Event class designator.
   b) Variable argument list.

8.5.7.2 **Returned arguments**
   a) Item event designator.

8.5.7.3 **Preconditions**
   a) None.

8.5.7.4 **Postconditions**
   a) The new event has been returned.

8.5.7.5 **Exceptions**
   a) Out of memory.

8.5.8 **BroadcastEvent method**

The BroadcastEvent method shall broadcast an event starting at target item and return either a retraction handle or the return value from the method underlying the event. The dispatch directive argument shall be one of the following values to specify whether the event should be sent, posted, or scheduled.
   - The event will be sent immediately.
   - The event will be posted.
   - The event will be scheduled.

The target item designator argument shall be the target item that will receive the event. The event class designator argument shall identify the event to be sent. The cascade directive argument shall be one of the following values to specify how the event should cascade after it has been processed by the target.
   - Do not cascade (default).
   - Cascade up.
   - Cascade down.

The optional scheduled time value argument shall be the date and time when the event is to occur. This parameter is required if the dispatch directive is for the message to be scheduled otherwise it must be omitted. The optional variable argument list argument shall be a variable number of event argument values. The returned event invocation return value or queued event handle shall be if the dispatch directive is for the message to be sent, the result is the return value from the last method to receive the event and return a value. If the dispatch directive is for the event to be posted or scheduled, the result shall be a handle that can be used with RetractEvent. If the target item does not have a method corresponding to the event, no error shall occur.

8.5.8.1 **Supplied arguments**
   a) Dispatch directive.
   b) Target item designator.
   c) Event class designator.
   d) Cascade directive.
   e) Scheduled time value.
   f) Variable argument list.
8.5.8.2 Returned arguments

a) Event invocation return value or queued event handle.
b) Preconditions
c) None.

8.5.8.3 Postconditions

a) The event has been sent, scheduled, or posted.
b) The event invocation return value or queued event handle has been returned.

8.5.8.4 Exceptions

a) None.

8.5.9 ScheduleBroadcast method

The ScheduleBroadcast method shall schedule an event to be broadcasted starting at target item and return a retraction handle. The target item designator argument shall be the target item that will receive the event. The event class designator argument shall identify the event to be sent. The cascade directive argument shall be one of the following values to specify how the event should cascade after it has been processed by the target.

- Do not cascade (default).
- Cascade up.
- Cascade down.

The scheduled time value argument shall be the date and time when the event is to occur. The optional variable argument list argument shall be a variable number of event argument values. The returned queued event handle shall be a handle that can be used with RetractEvent. If the item does not have a method corresponding to the event, no error shall occur. The receiving method shall have a name that matches the event name, and an argument list that accepts the sequence of values in the variable argument list.

8.5.9.1 Supplied arguments

a) Target item designator.
b) Event class designator.
c) Cascade directive.
d) Scheduled time value.
e) Variable argument list.

8.5.9.2 Returned arguments

a) Queued event handle.

8.5.9.3 Preconditions

a) None.

8.5.9.4 Postconditions

a) Event has been scheduled.
b) The queued event handle has been returned.
8.5.9.5 **Exceptions**

a) None.

**8.6 Model–Defined Item Behavior**

Model–defined run–time item behavior shall be composed from Script elements combined using the following precedence (lowest number has highest precedence):

1. Script child elements of the item’s corresponding element.
2. Script child elements of the prototype element of the corresponding item class.
3. Script child elements of the corresponding item class that have Instance or Isolated placement.
4. Script child elements in super–classes of the item class in a left to right pre–order traversal, the same manner as the primary item class.

Where possible, a script shall have direct, unqualified access to the following in order of precedence:

1. The members of the item.
2. The members of external objects added to the item as extensions.
3. The members of external objects that have been added to the simulation as extensions.
4. The named external objects that have been added as extensions.
5. The members of the owning simulation.
9 SIMULATION INTERFACE

All Simulation objects shall implement the Simulation interface, which is an extension of the Item interface.

![Simulation Interface Diagram]

Figure 6: Simulation Interface

Every Simulation object shall behave as an item, and be substitutable as an item. Therefore the following conditions must remain true:
- A simulation object reference may be treated as an item object reference.
- Any variable that is compatible with an item shall be compatible with a simulation.

9.1 Simulation Object Construction

A Simulation object may be directly instantiated without using the CreateSimulation method of the Simulator interface.

9.2 Simulation Object Structure

Each Simulation object shall own and manage the items it creates.

Each Simulation object shall maintain an event queue.

9.3 Simulation Object States

A Simulation object begins in the Unloaded state. When one of the Load methods is invoked, the simulation enters the Loading state until completion, wherein it enters the Loaded state. When either Run or DoNextEvent is invoked, the Simulation object enters the Started state. When the simulation’s event queue becomes empty, the Simulation object enters the Ended state. If an unrecoverable error occurs the Simulation object shall enter the Error state. If the Abort method is invoked at any time, the Aborted state is entered.
9.4 Simulation Information Services

9.4.1 State property

The State property shall contain the state of the simulation. This property shall be read-only. The returned simulation current state designator shall be a value from the following list, specifying the state of the simulation.

- The simulation is in an unloaded state.
- The simulation is loading.
- The simulation is loaded but hasn’t been started.
- The simulation has been started.
- The simulation has ended normally.
- The simulation has an error.

9.4.2 StateChanged notification

The StateChanged notification shall occur immediately after a change in the state of the simulation. The simulation new state designator argument shall be the new state. The simulation previous state designator argument shall be the previous state.

9.4.2.1 Supplied arguments

a) Simulation new state designator.
b) Simulation previous state designator.
9.4.2.2 Returned arguments
   a) None.

9.4.2.3 Preconditions
   a) None.

9.4.2.4 Postconditions
   a) Notification subscribers have been notified that the simulation state has changed.

9.4.2.5 Exceptions
   a) None.

9.4.3 Document property
The Document property shall contain or set the simulation’s XML document object. This is a read-write property. The DOM document designator shall be the DOM Document object that corresponds to the XML representation of the simulation.

9.4.4 ItemCount property
The ItemCount property shall contain the total number of items in the simulation. This property shall be read-only. The returned total item count shall be the total number of items in the simulation.

9.4.5 ItemClasses property
The ItemClasses property shall contain an Items collection containing the item classes in the simulation. This property shall be read-only. The returned list of item class designators shall be the collection of item classes.

9.4.6 ItemIDs method
The ItemIDs property shall contain the ItemID at a specified index in the simulation's namespace. This property shall be read-only. The item designator argument shall be an index number or a name identifying a simulation item. The returned ItemID shall be the item ID value of the item.

9.4.6.1 Supplied arguments
   a) Item designator.

9.4.6.2 Returned arguments
   a) ItemID.

9.4.6.3 Preconditions
   a) None.

9.4.6.4 Postconditions
   a) None.
9.4.6.5 **Exceptions**

a) None.

**9.5 Simulation Construction Services**

**9.5.1 Load method**

The Load method shall load the specified XML document. The resource path designator argument shall be the path to an XML document. The optional host provided data argument shall be data that is available during load. The optional load directives argument shall be any combination of the following values that specify the manner in which the file will be loaded.

- The items will be loaded.
- The events will be loaded.
- The items and events will be loaded (this is the default).

The optional name designator shall be the name of the Simulation element to be found within the XML document.

**9.5.1.1 Supplied arguments**

a) Resource path designator.
b) Host provided data.
c) Load directives.
d) Name designator.

**9.5.1.2 Returned arguments**

a) None.

**9.5.1.3 Preconditions**

a) None.

**9.5.1.4 Postconditions**

a) The simulation has been loaded.

**9.5.1.5 Exceptions**

a) Resource not found.
b) Simulation name not found.

**9.5.2 LoadXML method**

The LoadXML method shall load the simulation from an XML expression. The xml expression argument shall be the XML string to be loaded. The optional host provided data argument shall be data that is available during load. The optional load directives argument shall be any combination of the following values that specify the manner in which the file will be loaded. Default is srmlLoadDefault.

- The events will be loaded.
- The Items will be loaded.

**9.5.2.1 Supplied arguments**

a) XML expression.
b) Host provided data.
c) Load directives.

9.5.2.2 Returned arguments

a) None.

9.5.2.3 Preconditions

a) None.

9.5.2.4 Postconditions

a) The XML expression has been loaded.

9.5.2.5 Exceptions

a) None.

9.5.3 LoadItems method

The LoadItems method shall load the simulation from the document. The optional host provided data argument shall be data provided to the load context. The optional load directives argument shall be any combination of the following values that specify the manner in which the file will be loaded. Default is srmlLoadDefault.

- The events will be loaded.
- The Items will be loaded.

9.5.3.1 Supplied arguments

a) Host provided data.
b) Load directives.

9.5.3.2 Returned arguments

a) None.

9.5.3.3 Preconditions

a) None.

9.5.3.4 Postconditions

a) The simulation has been loaded.

9.5.3.5 Exceptions

a) None.

9.5.4 LoadItemSource notification

The LoadItemSource notification shall signal that an item needs to be loaded from its source. This notification shall occur when the simulator needs to load the object identified by the Source attribute of an XML element. The DOM node designator argument shall be the node containing the Source attribute that needs to be loaded. The URI scheme name argument shall be the portion of the Source attribute value that precedes the first colon (:) character, which contains the URI scheme name used to locate the object
to be loaded. The URI scheme data argument shall be the portion of the Source attribute value that follows the first character sequence that begins with a colon (:) optionally concatenated with up to two consecutive forward slash (/) characters, which defines the object to be loaded. The external item designator argument shall be the output parameter for storing an object reference to the loaded object.

9.5.4.1  **Supplied arguments**

a)  DOM node designator.

b)  URI scheme name.

c)  URI scheme data.

d)  External item designator.

9.5.4.2  **Returned arguments**

a)  External item designator.

9.5.4.3  **Preconditions**

a)  None.

9.5.4.4  **Postconditions**

a)  External item designator may have been set.

9.5.4.5  **Exceptions**

a)  None.

9.5.5  **DeleteItem method**

The DeleteItem method shall delete an item. The item designator argument shall be the item to be deleted, which can be a reference to the item or the item's ItemID value. Calling this method defers the destruction of the item until after the current event has completed.

9.5.5.1  **Supplied arguments**

a)  Item designator.

9.5.5.2  **Returned arguments**

a)  None.

9.5.5.3  **Preconditions**

a)  None.

9.5.5.4  **Postconditions**

a)  The item has been marked for deletion.

9.5.5.5  **Exceptions**

a)  None.
9.5.6 **DestroyItem method**

The DestroyItem method shall destroy an item. The item designator argument shall be the item to be destroyed, which can be a reference to the item or the item’s ItemID value. The item is destroyed immediately.

9.5.6.1 **Supplied arguments**

a) Item designator.

9.5.6.2 **Returned arguments**

a) None.

9.5.6.3 **Preconditions**

a) None.

9.5.6.4 **Postconditions**

a) The item was destroyed.

9.5.6.5 **Exceptions**

a) None.

9.5.7 **ReleaseAll method**

The ReleaseAll method shall release all objects held by the simulation, and return the simulation to the Unloaded state. This method shall also set the Document to a null reference.

9.5.7.1 **Supplied arguments**

a) None.

9.5.7.2 **Returned arguments**

a) None.

9.5.7.3 **Preconditions**

a) None.

9.5.7.4 **Postconditions**

a) All objects held by the simulation have been released.

9.5.7.5 **Exceptions**

a) None.

9.5.8 **UnloadItems method**

The UnloadItems method shall unload all of the items, and return the simulation to the Unloaded state.
9.5.8.1 Supplied arguments
   a) None.

9.5.8.2 Returned arguments
   a) None.

9.5.8.3 Preconditions
   a) None.

9.5.8.4 Postconditions
   a) All items held by the simulation have been released.

9.5.8.5 Exceptions
   a) None.

9.5.9 Save method

The Save method shall save the simulation to a file. The resource path designator argument shall be the path to an XML document where the simulation is to be saved. The optional save directives argument shall be any combination of the following values that specify the manner in which the file will be saved.
   - Only the DOM document is updated by the save.
   - The events will be saved. (default)
   - The items will be saved. (default)

9.5.9.1 Supplied arguments
   a) Resource path designator.
   b) Save directives.

9.5.9.2 Returned arguments
   a) None.

9.5.9.3 Preconditions
   a) None.

9.5.9.4 Postconditions
   a) The simulation has been saved.

9.5.9.5 Exceptions
   a) None.

9.5.10 ArgumentSerializer property

The ArgumentSerializer property shall contain an ArgumentSerializer object reference to assist in the saving and loading of item event arguments. This property shall be read-write. This property may contain a null reference.
9.5.10.1 Supplied arguments
   a) None.

9.5.10.2 Returned arguments
   a) None.

9.5.10.3 Preconditions
   a) None.

9.5.10.4 Postconditions
   a) The simulation is in the Loaded state.

9.5.10.5 Exceptions
   a) None.

9.5.11 Reset method
The Reset method shall reset the document to its initial state and load the simulation from the document.

9.5.11.1 Supplied arguments
   a) None.

9.5.11.2 Returned arguments
   a) None.

9.5.11.3 Preconditions
   a) None.

9.5.11.4 Postconditions
   a) The simulation is in the Loaded state.

9.5.11.5 Exceptions
   a) None.

9.5.12 Idle notification
The Idle notification shall signal that an asynchronous operation is waiting to complete. This notification shall occur when the simulation is waiting for an asynchronous operation to complete. The directive to cancel the operation argument shall be used to request that the simulation stop waiting for the asynchronous operation.

9.5.12.1 Supplied arguments
   a) Directive to cancel the operation.
9.5.12.2 **Returned arguments**
   a) Directive to cancel the operation.

9.5.12.3 **Preconditions**
   a) None.

9.5.12.4 **Postconditions**
   a) Notification subscribers have been notified that the simulation is waiting.

9.5.12.5 **Exceptions**
   a) None.

9.5.13 **Error property**

The Error property shall contain an object describing the simulation’s most recent error. This property shall be read-only. The returned error designator shall be an Error object.

9.5.14 **IsItemLocationFixed method**

The IsItemLocationFixed method shall return true if an item has a LocationFixed property set to true. The item designator argument shall be the item to be tested. The returned location fixed indicator shall be the return value of true signifies that varItem has a LocationFixed property value set to true.

9.5.14.1 **Supplied arguments**
   a) Item designator.

9.5.14.2 **Returned arguments**
   a) Location fixed indicator.

9.5.14.3 **Preconditions**
   a) None.

9.5.14.4 **Postconditions**
   a) The location fixed indicator has been returned.

9.5.14.5 **Exceptions**
   a) None.

9.5.15 **SetItemLocation method**

The SetItemLocation method shall set an item’s location. The item designator argument shall be the item that will change location. The location item designator argument shall be the item that will become the location for the item that will change location.

9.5.15.1 **Supplied arguments**
   a) Item designator.
b) Location item designator.

9.5.15.2 Returned arguments

a) None.

9.5.15.3 Preconditions

a) None.

9.5.15.4 Postconditions

a) The item has been set to the designated location.

9.5.15.5 Exceptions

a) None.

9.6 Item Monitoring Services

9.6.1 ItemCreated notification

The ItemCreated notification shall signal that an item was created. This notification shall occur after an event has been created. The ItemID argument shall be the value of the ItemID of the item that was created.

9.6.1.1 Supplied arguments

a) ItemID.

9.6.1.2 Returned arguments

a) None.

9.6.1.3 Preconditions

a) None.

9.6.1.4 Postconditions

a) Notification subscribers have been notified that the item has been created.

9.6.1.5 Exceptions

a) None.

9.6.2 ItemDestroyed notification

The ItemDestroyed notification shall signal that an item was destroyed. This notification shall occur after an item has been destroyed. The ItemID argument shall be the value of the ItemID of the item that was created.

9.6.2.1 Supplied arguments

a) ItemID.
9.6.2.2 Returned arguments
   a) None.

9.6.2.3 Preconditions
   a) None.

9.6.2.4 Postconditions
   a) Notification subscribers have been notified that the item has been destroyed.

9.6.2.5 Exceptions
   a) None.

9.6.3 ItemLocationChanged notification

The ItemLocationChanged notification shall signal that an item's location has changed. This notification shall occur after an item's location has changed. The item designator argument shall be the item that has changed location. The previous location designator argument shall be the item's previous location.

9.6.3.1 Supplied arguments
   a) Item designator.
   b) Previous location designator.

9.6.3.2 Returned arguments
   a) None.

9.6.3.3 Preconditions
   a) None.

9.6.3.4 Postconditions
   a) Notification subscribers have been notified that the item's location has been changed.

9.6.3.5 Exceptions
   a) None.

9.6.4 ItemPropertyChanged notification

The ItemPropertyChanged notification shall signal that an item's property has changed. This notification shall occur after an item's property has changed. The item designator argument shall be the item that has had a property change. The property name argument shall be the property name that has changed. The property value argument shall be the new value of the property.

9.6.4.1 Supplied arguments
   a) Item designator.
   b) Property name.
   c) Property value.
9.6.4.2 **Returned arguments**

a) None.

9.6.4.3 **Preconditions**

a) None.

9.6.4.4 **Postconditions**

a) Notification subscribers have been notified that the item’s property has changed.

9.6.4.5 **Exceptions**

a) None.

9.6.5 **BeforeItemDeleted notification**

The BeforeItemDeleted notification shall signal that an item is about to be deleted. This notification shall occur before an item is deleted. The ItemID argument shall be the value of the ItemID of the item that was created. The directive to cancel the operation argument shall be if true, the deletion will be canceled.

9.6.5.1 **Supplied arguments**

a) ItemID.

b) Directive to cancel the operation.

9.6.5.2 **Returned arguments**

a) Directive to cancel the operation.

9.6.5.3 **Preconditions**

a) None.

9.6.5.4 **Postconditions**

a) Notification subscribers have been notified that the item has been deleted.

b) Deletion was possibly canceled.

9.6.5.5 **Exceptions**

a) None.

9.7 **Item Linking Services**

9.7.1 **AddLink method**

The AddLink method shall add a link from a source item to a target item, and returns the link handle. The source item designator argument shall be the item that will receive the link. The optional target item designator argument shall be the link target item. The optional link name argument shall be the link name. The optional links collection name argument shall be the name of the links collection that will contain the link. The optional directive to place the link(s) in IDREF(s) argument shall be if true, the link will be added using an IDREF or IDREFS type attribute. The returned link handle shall be a unique value that identifies the link.
9.7.1.1 **Supplied arguments**

a) Source item designator.
b) Target item designator.
c) Link name.
d) Links collection name.
e) Directive to place the link(s) in IDREF(s).

9.7.1.2 **Returned arguments**

a) Link handle.

9.7.1.3 **Preconditions**

a) None.

9.7.1.4 **Postconditions**

a) The link was added to the item.
b) The link handle was returned.

9.7.1.5 **Exceptions**

a) None.

9.7.2 **GetLinkNode method**

The GetLinkNode method shall return the DOM node that describes the link. The link handle argument shall be a unique value that identifies the link. The returned link node designator shall have a Node interface.

9.7.2.1 **Supplied arguments**

a) Link handle.

9.7.2.2 **Returned arguments**

a) Link node designator.

9.7.2.3 **Preconditions**

a) None.

9.7.2.4 **Postconditions**

a) The link node has been returned.

9.7.2.5 **Exceptions**

a) None.
9.7.3 **RemoveLink method**

The RemoveLink method shall remove a link. The link handle argument shall be a unique value that identifies the link.

9.7.3.1 **Supplied arguments**

a) Link handle.

9.7.3.2 **Returned arguments**

a) None.

9.7.3.3 **Preconditions**

a) None.

9.7.3.4 **Postconditions**

a) The link has been removed from the item.

9.7.3.5 **Exceptions**

a) None.

9.8 **Item Dynamics Services**

9.8.1 **InvokeMethod method**

The InvokeMethod method shall invoke a method on a specified object and returns the result. The object designator argument shall be the object that will receive the method invocation. The method designator argument shall be the name of the method to invoke. The optional list of arguments argument shall be an array of arguments. The returned event invocation return value shall be the return value from the method underlying the event. If the item does not have a method corresponding to strMethod, an error occurs.

9.8.1.1 **Supplied arguments**

a) Object designator.
b) Method designator.
c) List of arguments.

9.8.1.2 **Returned arguments**

a) Event invocation return value.

9.8.1.3 **Preconditions**

a) None.

9.8.1.4 **Postconditions**

a) The method has been invoked.

9.8.1.5 **Exceptions**

a) Member not found.
9.8.2 MemberExists method

The MemberExists method shall return true if an item has a particular member. The item designator argument shall be the item to be tested. The member designator argument shall be the name of the member. The returned member exists indicator shall be the value true if has a member named by the member designator.

9.8.2.1 Supplied arguments

a) Item designator.
b) Member designator.

9.8.2.2 Returned arguments

a) Member exists indicator.

9.8.2.3 Preconditions

a) None.

9.8.2.4 Postconditions

a) None.

9.8.2.5 Exceptions

a) None.

9.8.3 AddProperty method

The AddProperty method shall add a property to an item. The item designator argument shall be the item that will receive a new property. The property name argument shall be the property name to add. The optional data type name argument shall be the data type name. The optional property value argument shall be the initial value.

9.8.3.1 Supplied arguments

a) Item designator.
b) Property name.
c) Data type name.
d) Property value.

9.8.3.2 Returned arguments

a) None.

9.8.3.3 Preconditions

a) None.

9.8.3.4 Postconditions

a) The property has been added to the item.
9.8.3.5 **Exceptions**

a) Out of memory.

9.8.4 **RemoveMember method**

The RemoveMember method shall remove a member from an item. The item designator argument shall be the item from which the member will be removed. The member designator argument shall be the member name to be removed.

9.8.4.1 **Supplied arguments**

a) Item designator.

b) Member designator.

9.8.4.2 **Returned arguments**

a) None.

9.8.4.3 **Preconditions**

a) None.

9.8.4.4 **Postconditions**

a) The property has been removed from the item.

9.8.4.5 **Exceptions**

a) None.

9.9 **Runtime Services**

9.9.1 **AddExtension method**

The AddExtension method shall add an extension object to the simulation runtime environment. The extension object designator argument shall be an object to extend the simulation’s runtime environment. The optional extension name argument shall be the name that will globally identify the extension object in the runtime environment. The optional directive to add the extension object's members as globals in the runtime environment argument shall be if true, the members of the extension object will be globally added to the runtime environment. The optional before item designator argument shall be the index number of an existing object to insert the extension before. The optional after item designator argument shall be the index number of an existing object to insert the extension after.

9.9.1.1 **Supplied arguments**

a) Extension object designator.

b) Extension name.

c) Directive to add the extension object's members as global in the runtime environment.

d) Before item designator.

e) After item designator.

9.9.1.2 **Returned arguments**

a) None.
9.9.1.3  **Preconditions**
   a) None.

9.9.1.4  **Postconditions**
   a) The extension object has been added to the runtime environment.
   b) Out of memory.

9.9.1.5  **Exceptions**
   a) None.

9.9.2  **RemoveExtension method**

The RemoveExtension method shall remove an extension that was previously added with AddExtension. The element designator argument shall be the index number or name of the extension.

9.9.2.1  **Supplied arguments**
   a) Element designator.

9.9.2.2  **Returned arguments**
   a) None.

9.9.2.3  **Preconditions**
   a) None.

9.9.2.4  **Postconditions**
   a) The extension object has been removed from the runtime environment.

9.9.2.5  **Exceptions**
   a) Item not found.

9.9.3  **ClearExtensions method**

The ClearExtensions method shall remove all extensions from the simulation.

9.9.3.1  **Supplied arguments**
   a) Element designator.

9.9.3.2  **Returned arguments**
   a) None.

9.9.3.3  **Preconditions**
   a) None.
9.9.3.4 Postconditions
   a) All extensions have been removed.

9.9.3.5 Exceptions
   a) None.

9.10 Time Services

9.10.1 CurrentTime property

The CurrentTime property shall contain or set the simulation’s current time. This is a read-write property. The returned current time value shall be a date/time value.

9.10.1.1 Supplied arguments
   a) None.

9.10.1.2 Returned arguments
   a) Current time value.

9.10.1.3 Preconditions
   a) None.

9.10.1.4 Postconditions
   a) Current time has been returned.

9.10.1.5 Exceptions
   a) None.

9.10.2 EndTime property

The EndTime property shall contain or set the simulation’s end time. This is a read-write property. The returned end time value shall be a date/time value.

9.10.2.1 Supplied arguments
   a) None.

9.10.2.2 Returned arguments
   a) End time value.

9.10.2.3 Preconditions
   a) None.

9.10.2.4 Postconditions
   a) End time has been returned.
9.10.2.5 Exceptions

a) None.

9.10.3 CurrentTimeSet notification

The CurrentTimeSet notification shall signal that the current time was set. This notification shall occur after the CurrentTime property has changed. The previous time value argument shall be the previous value of the CurrentTime property. This only occurs when assigning a value to the CurrentTime property.

9.10.3.1 Supplied arguments

a) Previous time value.

9.10.3.2 Returned arguments

a) None.

9.10.3.3 Preconditions

a) None.

9.10.3.4 Postconditions

a) Notification subscribers have been notified that the simulation current time has been set.

9.10.3.5 Exceptions

a) None.

9.11 Event Definition Services

9.11.1 EventClasses property

The EventClasses property shall contain a collection of event classes. This property shall be read-only. The returned list of event class designators shall be an Items collection containing all of the event classes in the simulation.

9.11.1.1 Supplied arguments

a) None.

9.11.1.2 Returned arguments

a) List of event class designators.

9.11.1.3 Preconditions

a) None.

9.11.1.4 Postconditions

a) The list of event class designators has been returned.
9.11.1.5 Exceptions
   a) None.

9.11.2 SetEventClassPriority method
The SetEventClassPriority method shall set the priority for an event class. The event class designator argument shall be the event class name or its object reference. The priority level argument shall be the new priority value, which can be a negative or positive integer. By default, events have a priority of zero.

9.11.2.1 Supplied arguments
   a) Event class designator.
   b) Priority level.

9.11.2.2 Returned arguments
   a) None.

9.11.2.3 Preconditions
   a) None.

9.11.2.4 Postconditions
   a) The priority of the event class has been set.

9.11.2.5 Exceptions
   a) None.

9.11.3 RetractEvent method
The RetractEvent method shall retract an event that has been posted or scheduled, and returns the event. The queued event handle argument shall be the handle to an event that is scheduled or posted. The returned item event designator shall be the item event that was retracted.

9.11.3.1 Supplied arguments
   a) Queued event handle.

9.11.3.2 Returned arguments
   a) Item event designator.

9.11.3.3 Preconditions
   a) None.

9.11.3.4 Postconditions
   a) The event has been retracted.

9.11.3.5 Exceptions
   a) None.
9.11.4  RetractAllEvents method

The RetractAllEvents method shall retract all events that have been posted or scheduled to a target item. The optional target item designator argument shall be the target item that will have its events retracted. If no events have been posted or scheduled to the target item, no error shall occur. If the target item designator is not provided, then the events shall be retracted for all items.

9.11.4.1  Supplied arguments
   a) Target item designator.

9.11.4.2  Returned arguments
   a) None.

9.11.4.3  Preconditions
   a) None.

9.11.4.4  Postconditions
   a) All events posted and scheduled for the item (or all items) have been retracted.

9.11.4.5  Exceptions
   a) None.

9.12  Event Information Services

9.12.1  EventCount property

The EventCount property shall contain the number of scheduled and posted events in the simulation. This property shall be read-only. The returned total event count shall be the total number of scheduled and posted events in the simulation.

9.12.1.1  Supplied arguments
   a) None.

9.12.1.2  Returned arguments
   a) Total event count.

9.12.1.3  Preconditions
   a) None.

9.12.1.4  Postconditions
   a) The total event count has been returned.

9.12.1.5  Exceptions
   a) None.
9.12.2  GetEvent method

The GetEvent method shall return the ItemEvent object at the specified index in the simulation's event queue. The index number or queued event handle argument shall be an index number or a handle identifying the item event. The returned item event designator shall be the item event instance.

9.12.2.1  Supplied arguments

a)  Index number or queued event handle.

9.12.2.2  Returned arguments

a)  Item event designator.

9.12.2.3  Preconditions

a)  None.

9.12.2.4  Postconditions

a)  The item event designator (or a null reference) has been returned.

9.12.2.5  Exceptions

a)  None.

9.12.3  PeekEvent method

The PeekEvent method shall return the next event in the event queue, but does not remove it. The returned item event designator shall be the item next event in the event manager.

9.12.3.1  Supplied arguments

a)  None.

9.12.3.2  Returned arguments

a)  Item event designator.

9.12.3.3  Preconditions

a)  None.

9.12.3.4  Postconditions

a)  The item event designator (or a null reference) has been returned.

9.12.3.5  Exceptions

a)  None.
9.13 Event Execution Services

9.13.1 DoNextEvent method

The DoNextEvent method shall send the next scheduled or posted event to the designated target and return the event. The returned item event designator shall be the event that was processed. If no events are on the event queue, this method shall return a null item event designator.

Model–defined event sinks that are enabled shall be triggered when this method processes an item event that meets the conditions of the event sink. Once triggered, the owning item’s corresponding method shall be invoked.

This method shall set the simulation’s current time to match the time of the processed event before dispatching the event.

9.13.1.1 Supplied arguments

a) None.

9.13.1.2 Returned arguments

a) Item event designator.

9.13.1.3 Preconditions

a) None.

9.13.1.4 Postconditions

a) The simulation has been set to the Started state.
b) The next event has been processed.
c) The simulation’s current time was set to the time of the processed event.

9.13.1.5 Exceptions

a) None.

9.13.2 Run method

The Run method shall run the simulation. This method contains a loop that calls DoNextEvent.

9.13.2.1 Supplied arguments

a) None.

9.13.2.2 Returned arguments

a) None.

9.13.2.3 Preconditions

a) None.

9.13.2.4 Postconditions

a) The simulation has been set to the Ended state.
9.13.2.5 **Exceptions**
   a) None.

9.13.3 **Abort method**

The Abort method shall abort a simulation.

9.13.3.1 **Supplied arguments**
   a) None.

9.13.3.2 **Returned arguments**
   a) None.

9.13.3.3 **Preconditions**

None.

9.13.3.4 **Postconditions**
   a) The simulation has been set to the Aborted state.

9.13.3.5 **Exceptions**
   a) None.

9.13.4 **CurrentEvent property**

The CurrentEvent property shall contain the event that is currently being processed by a call to DoNextEvent. This property shall be read-only. The returned item event designator shall be the current event. Avoid using this property when it is likely that another thread could change the value during processing.

9.13.5 **ForwardEvent method**

The ForwardEvent method shall send the designated event to a specific object. The item event designator argument shall be the item event that was processed. The object designator argument shall be the object that will receive the event. The optional directive to include item class prefix argument shall be if true, the method to be invoked on the object shall be formatted with the prefix of the item class (or type name) of the original event target. Default is false. The returned event invocation return value shall be the return value from the method underlying the event. This method does not change the original source or target objects.

9.13.5.1 **Supplied arguments**
   a) Item event designator.
   b) Object designator.
   c) Directive to include item class prefix.

9.13.5.2 **Returned arguments**
   a) Event invocation return value.
9.13.5.3 **Preconditions**

a) None.

9.13.5.4 **Postconditions**

a) The designated event has been forwarded to the object.

9.13.5.5 **Exceptions**

a) None.

9.14 **Event Sink Services**

9.14.1 **AddEventSink method**

The AddEventSink method shall add an event sink to the simulation runtime environment. The owner item designator argument shall be the item that owns the event sink, and therefore receives events. The optional event sink name argument shall be the name of the event sink. The optional list of event class designators argument shall be the event classes. The optional event method designator argument shall be the name of the method that will be invoked in the owner when an event occurs. The optional list of item class designators argument shall be the item classes. The optional list of item source item designators argument shall be the source items. The optional item class property designators argument shall be the names of properties that cause the event sink to be triggered when they are changed. The optional property method designator argument shall be the name of the method that will be invoked in the owner when a property changes. The optional directive to trigger on location changes shall be either true or false. If true, the event sink shall be triggered when an event source’s location changes. The optional location method designator argument shall be the name of the method that will be invoked in the owner when a Location changes. The optional directive to trigger when items are created and destroyed argument shall be if true, causes the event sink to be triggered when an event source is created or destroyed. Default is false. The optional creation method designator argument shall be the name of the method that will be invoked in the owner when an event source is created. The optional destruction method designator argument shall be the name of the method that will be invoked in the owner when an event source is destroyed. The optional directive to enable argument shall be if true, the event sink shall be enabled (default is true). The returned event sink handle shall be a unique value that identifies the event sink.

9.14.1.1 **Supplied arguments**

a) Owner item designator.
b) Event sink name.
c) List of event class designators.
d) Event method designator.
e) List of item class designators.
f) List of item source item designators.
g) Item class property designators.
h) Property method designator.
i) Directive to trigger on location changes.
j) Location method designator.
k) Directive to trigger when items are created and destroyed.
l) Creation method designator.
m) Destruction method designator.
n) Directive to enable.
9.14.1.2 Returned arguments
a) Event sink handle.

9.14.1.3 Preconditions
a) None.

9.14.1.4 Postconditions
a) The event sink has been added.

9.14.1.5 Exceptions
a) Out of memory.

9.14.2 GetEventSinkNode method
The GetEventSinkNode method shall return the DOM node that describes the event sink. The event sink handle argument shall be a unique value that identifies the event sink. The returned event sink node designator shall have a Node interface.

9.14.2.1 Supplied arguments
a) Event sink handle.

9.14.2.2 Returned arguments
a) Event sink node designator.

9.14.2.3 Preconditions
a) None.

9.14.2.4 Postconditions
a) The event sink node has been returned.

9.14.2.5 Exceptions
a) None.

9.14.3 RemoveEventSink method
The RemoveEventSink method shall remove an event sink. The link handle argument shall be a unique value that identifies the event sink.

9.14.3.1 Supplied arguments
a) Link handle.

9.14.3.2 Returned arguments
a) None.
9.14.3.3 **Preconditions**
   a) None.

9.14.3.4 **Postconditions**
   a) The event sink has been removed.

9.14.3.5 **Exceptions**
   a) None.

9.15 **Event Monitoring Services**

9.15.1 **BeforeEvent notification**

The BeforeEvent notification shall signal that an event is about to be received. This notification shall occur immediately before an event reaches an item. The item designator argument shall be the item that will receive the event. The event class designator argument shall be the name of the event. The list of arguments argument shall be an array containing the event arguments.

9.15.1.1 **Supplied arguments**
   a) Item designator.
   b) Event class designator.
   c) List of arguments.

9.15.1.2 **Returned arguments**
   a) None.

9.15.1.3 **Preconditions**
   a) None.

9.15.1.4 **Postconditions**
   a) Notification subscribers have been notified that the item event is about to be received by the item.

9.15.1.5 **Exceptions**
   a) None.

9.15.2 **BeforeItemEvent notification**

The BeforeItemEvent notification shall signal that an ItemEvent is about to be received. This notification shall occur immediately before an item event reaches an item. The item event designator argument shall be the ItemEvent object that contains information about the event.

9.15.2.1 **Supplied arguments**
   a) Item event designator.
9.15.2.2 **Returned arguments**

a) None.

9.15.2.3 **Preconditions**

a) None.

9.15.2.4 **Postconditions**

a) Notification subscribers have been notified that the item event is about to be received by the item.

9.15.2.5 **Exceptions**

a) None.

9.15.3 **EventSent notification**

The EventSent notification shall signal that an event has been sent. This notification shall occur immediately after an event is sent in the simulation. The item designator argument shall be the item that received the event. The event class designator argument shall be the name of the event. The list of arguments argument shall be an array containing the event arguments.

9.15.3.1 **Supplied arguments**

a) Item designator.
b) Event class designator.
c) List of arguments.

9.15.3.2 **Returned arguments**

a) None.

9.15.3.3 **Preconditions**

a) None.

9.15.3.4 **Postconditions**

a) Notification subscribers have been notified that the event has been received by the item.

9.15.3.5 **Exceptions**

a) None.

9.15.4 **ItemEventSent notification**

The ItemEventSent notification shall signal that an event has been sent. This notification shall occur after an item event has been sent. The item event designator argument shall be the ItemEvent object that contains information about the event.

9.15.4.1 **Supplied arguments**

a) Item event designator.
9.15.4.2 Returned arguments
   a) None.

9.15.4.3 Preconditions
   a) None.

9.15.4.4 Postconditions
   a) Notification subscribers have been notified that the item event has been received by the item.

9.15.4.5 Exceptions
   a) None.

9.15.5 EventPosted notification
The EventPosted notification shall signal that an event has been posted to the simulation's event queue. This notification shall occur immediately after an event is posted in the simulation. The item event designator argument shall be the ItemEvent object that contains information about the event.

9.15.5.1 Supplied arguments
   a) Item event designator.

9.15.5.2 Returned arguments
   a) None.

9.15.5.3 Preconditions
   a) None.

9.15.5.4 Postconditions
   a) Notification subscribers have been notified that the item event has been posted in the simulation's event queue.

9.15.5.5 Exceptions
   a) None.

9.15.6 EventScheduled notification
The EventScheduled notification shall signal that an event has been scheduled. This notification shall occur immediately after an event is scheduled in the simulation. The item event designator argument shall be the ItemEvent object that contains information about the event.

9.15.6.1 Supplied arguments
   a) Item event designator.
9.15.6.2 Returns arguments
   a) None.

9.15.6.3 Preconditions
   a) None.

9.15.6.4 Postconditions
   a) Notification subscribers have been notified that the item event has been scheduled in the simulation's event queue.

9.15.6.5 Exceptions
   a) None.

9.15.7 EventRetracted notification

The EventRetracted notification shall signal that an event was retracted. This notification shall occur immediately after an event is retracted in the simulation. The item event designator argument shall be the ItemEvent object that contains information about the event.

9.15.7.1 Supplied arguments
   a) Item event designator.

9.15.7.2 Returns arguments
   a) None.

9.15.7.3 Preconditions
   a) None.

9.15.7.4 Postconditions
   a) Notification subscribers have been notified that the item event has been removed from the simulation's event queue.

9.15.7.5 Exceptions
   a) None.

9.15.8 EventOccurred notification

The EventOccurred notification shall signal that an event has been received. This notification shall occur after a scheduled or posted notification has been processed. The item designator argument shall be the item that received the event. The event class designator argument shall be the name of the event. The list of arguments argument shall be an array containing the event arguments. This notification does not fire for events sent with SendEvent.

9.15.8.1 Supplied arguments
   a) Item designator.
   b) Event class designator.
c) List of arguments.

9.15.8.2 Returned arguments

a) None.

9.15.8.3 Preconditions

a) None.

9.15.8.4 Postconditions

a) Notification subscribers have been notified that the event has been received.

9.15.8.5 Exceptions

a) None.

9.15.9 ItemEventOccurred notification

The ItemEventOccurred notification shall signal that an item event has occurred. This notification shall occur after an item event has been processed. The item event designator argument shall be the ItemEvent object that contains information about the event. This notification does not fire for events sent with SendItemEvent.

9.15.9.1 Supplied arguments

a) Item event designator.

9.15.9.2 Returned arguments

a) None.

9.15.9.3 Preconditions

a) None.

9.15.9.4 Postconditions

a) Notification subscribers have been notified that the item event has been received.

9.15.9.5 Exceptions

a) None.

9.16 Statistical Services

9.16.1 RandomSeed property

The RandomSeed property shall contain or set the random seed. This property shall be read-write. The returned random seed value shall be the random seed. The random seed is a double precision value.

9.16.2 Random method

The Random method shall return a random number for the specified distribution type. The random number generator name argument shall be the name of a random number generator that provides a
distribution of random numbers. The optional variable argument list argument shall be a variable number of parameters as defined by the distribution. The returned random number shall be a random value returned by the Random function. If the specific distribution does not exist, this method triggers the LoadRandomNumberGenerator event in an attempt to load the generator from an external source.

9.16.2.1 Supplied arguments

a) Random number generator name.
b) Variable argument list.

9.16.2.2 Returned arguments

a) Random number.

9.16.2.3 Preconditions

a) None.

9.16.2.4 Postconditions

a) A random number has been returned.

9.16.2.5 Exceptions

a) None.

9.16.3 RandomizeSeed method

The RandomizeSeed method shall randomize the random seed used by random number generators.

9.16.3.1 Supplied arguments

a) None.

9.16.3.2 Returned arguments

a) None.

9.16.3.3 Preconditions

a) None.

9.16.3.4 Postconditions

a) The random seed has been randomized.

9.16.3.5 Exceptions

a) None.

9.16.4 GetRandomNumberGenerator method

The GetRandomNumberGenerator method shall return a new or existing random number generator for the specified distribution type and distribution parameters. The random number generator name argument shall be the name of a distribution to return. The optional variable argument list argument shall be a variable number of parameters as defined by the distribution. The returned random number generator
designator shall be the returned random number generator. If the specific distribution does not exist, this method triggers the LoadRandomNumberGenerator event in an attempt to load the generator from an external source.

9.16.4.1 **Supplied arguments**

a) Random number generator name.

b) Variable argument list.

9.16.4.2 **Returned arguments**

a) Random number generator designator.

9.16.4.3 **Preconditions**

a) None.

9.16.4.4 **Postconditions**

a) A new random number generator has been returned.

9.16.4.5 **Exceptions**

a) None.

9.16.5 **LoadRandomNumberGenerator notification**

The LoadRandomNumberGenerator notification shall signal that a random number generator needs to be loaded. This notification shall occur when the simulator needs to load the random number generator that was identified in a call to the Random method, for which it could not load the random number generator itself. The random number generator name argument shall be the type name of a random number. The list of arguments shall be an array containing the random number specific arguments. The random number generator designator argument shall be the output parameter for storing the random number generator. This object must have a default property that returns a random value. It is the responsibility of the loading procedure to initialize the random number generator.

9.16.5.1 **Supplied arguments**

a) Random number generator name.

b) List of arguments.

c) Random number generator designator.

9.16.5.2 **Returned arguments**

a) Random number generator designator.

9.16.5.3 **Preconditions**

a) None.

9.16.5.4 **Postconditions**

a) Notification subscribers have been notified that the designated random number needs to be loaded.

b) The random number generator may have been loaded.
9.16.5.5 Exceptions

a) None.

9.16.6 CreateRandomUniform method

The CreateRandomUniform method shall return a new RandomNumberGenerator object that generates a uniform distribution of random numbers. The lower bound argument shall be the smallest number generated from the generator. The upper bound argument shall be the largest number generated from the generator. The optional random seed shall be the initial random seed value. The optional multiplier value argument shall be one of the values from the table of multipliers in
ANNEX A. The returned random number generator designator shall be the RandomNumberGenerator object reference that has been created.

9.16.6.1 Supplied arguments

- a) Lower bound.
- b) Upper bound.
- c) Random seed.
- d) Multiplier value.

9.16.6.2 Returned arguments

- a) Random number generator designator.

9.16.6.3 Preconditions

- a) None.

9.16.6.4 Postconditions

- a) A new random number generator object has been returned.

9.16.6.5 Exceptions

- a) None.

9.16.7 CreateRandomExponential method

The CreateRandomExponential method shall return a new RandomNumberGenerator object that generates an exponential distribution of random numbers. The mean value argument shall be the mean value of the distribution. The optional random seed shall be the initial random seed value. The optional multiplier value argument shall be one of the values from the table of multipliers in
ANNEX A. The returned random number generator designator shall be the RandomNumberGenerator object reference that has been created.

9.16.7.1 **Supplied arguments**

a) Mean value.
b) Random seed.
c) Multiplier value.

9.16.7.2 **Returned arguments**

a) Random number generator designator.

9.16.7.3 **Preconditions**

a) None.

9.16.7.4 **Postconditions**

a) A new random number generator object has been returned.

9.16.7.5 **Exceptions**

a) None.

9.16.8 **CreateRandomTriangular method**

The CreateRandomTriangular method shall return a new RandomNumberGenerator object that generates a triangular distribution of random numbers. The minimum value argument shall be the minimum value of the distribution. The maximum value argument shall be the maximum value of the distribution. The mode value argument shall be the mode value of the distribution. The optional random seed shall be the initial random seed value. The optional multiplier value argument shall be one of the values from the table of multipliers in
ANNEX A. The returned random number generator designator shall be the RandomNumberGenerator object reference that has been created.

9.16.8.1 Supplied arguments

a) Minimum value.
b) Mode value.
c) Maximum value.
d) Random seed.
e) Multiplier value.

9.16.8.2 Returned arguments

a) Random number generator designator.

9.16.8.3 Preconditions

a) None.

9.16.8.4 Postconditions

a) A new random number generator object has been returned.

9.16.8.5 Exceptions

a) None.

9.16.9 CreateRandomNormal method

The CreateRandomNormal method shall return a new RandomNumberGenerator object that generates a normal distribution of random numbers. The mean value argument shall be the mean value of the distribution. The standard deviation value argument shall be the standard deviation of the distribution. The optional random seed shall be the initial random seed value. The optional multiplier value argument shall be one of the values from the table of multipliers in
ANNEX A. The returned random number generator designator shall be the RandomNumberGenerator object reference that has been created.

9.16.9.1 Supplied arguments

   a) Mean value.
   b) Standard deviation value.
   c) Maximum value.
   d) Random seed.
   e) Multiplier value.

9.16.9.2 Returned arguments

   a) Random number generator designator.

9.16.9.3 Preconditions

   a) None.

9.16.9.4 Postconditions

   a) A new random number generator object has been returned.

9.16.9.5 Exceptions

   a) None.

9.16.10 CreateTimeAverage method

The CreateTimeAverage method shall return a new TimeAverage object. The returned time average designator shall be the TimeAverage object reference that has been created.

9.16.10.1 Supplied arguments

   a) None.

9.16.10.2 Returned arguments

   a) Time average designator.

9.16.10.3 Preconditions

   a) None.

9.16.10.4 Postconditions

   a) A new time average object has been returned.

9.16.10.5 Exceptions

   a) None.

9.16.11 CreateBasicStatistics method

The CreateBasicStatistics method shall return a new BasicStatistics object. The returned basic statistics designator shall be the BasicStatistics object reference that has been created.
9.16.11.1 **Supplied arguments**
   a) None.

9.16.11.2 **Returned arguments**
   a) Time average designator.

9.16.11.3 **Preconditions**
   a) None.

9.16.11.4 **Postconditions**
   a) A new time average object has been returned.

9.16.11.5 **Exceptions**
   a) None.

9.17 **Support Services**

9.17.1 **CreateArray method**

The `CreateArray` method shall return a new array containing the elements specified in the arguments. The optional variable argument list argument shall be the objects and values to be placed in the array. The returned array designator shall be the array that has been created.

9.17.1.1 **Supplied arguments**
   a) Variable argument list.

9.17.1.2 **Returned arguments**
   a) Array designator.

9.17.1.3 **Preconditions**
   a) None.

9.17.1.4 **Postconditions**
   a) A new array has been returned.

9.17.1.5 **Exceptions**
   a) None.

9.17.2 **CreateCollection method**

The `CreateCollection` method shall return a new `Collection` object. The optional variable argument list argument shall be the objects and values to be placed into the collection. The returned collection designator shall be the object reference of the collection that has been created.
9.17.2.1 **Supplied arguments**
   a) Variable argument list.

9.17.2.2 **Returned arguments**
   a) Collection designator.

9.17.2.3 **Preconditions**
   a) None.

9.17.2.4 **Postconditions**
   a) A new collection has been returned.

9.17.2.5 **Exceptions**
   a) None.
10 ITEM EVENT INTERFACE

10.1 EventID property

The EventID property shall contain the unique event identifier for the item event. This property shall be read-only. The event’s relative temporal order among other events shall be obtainable by performing a left-to-right comparison of character values encoded in this property.

10.2 Name property

The Name property shall contain the event class name of the item event. This property shall be read-write.

10.3 Time property

The Time property shall contain the point in time when the event should occur. This property shall be read-write. If a time is not specified, the event will be posted as a next event for the item.

10.4 Priority property

The Priority property shall contain the priority of the event, which assists the in ordering events that have identical times. This property shall be read-write. Events with lower priority values shall be scheduled or posted to occur before those with higher priority values. Events with equal priority values will occur in the order in which they were scheduled or posted by the simulation engine.

10.5 Format property

The Format property shall contain the format of the event. This property shall be read-write. The format directive shall be one of the following values to specify the format.

- Method format (default).
- Item event format.

If the method format directive is used, the arguments of the item event shall be mapped directly to the arguments of the invoked method. If the item event format directive used, the item event instance shall be mapped to the single argument of the invoked method.

10.6 Cascade property

The Cascade property shall contain the cascade operation of the event. This property shall be read-write. The cascade directive shall be one of the following values to specify how the event should cascade after it has been processed by the target.

- Do not cascade (default).
- Cascade up.
- Cascade down.

10.7 Source property

The Source property may contain an object reference to the item that initiated the event. This property shall be read-write. This property may contain a null reference.

10.8 Target property

The Target property may contain an object reference to the item that’s intended to receive the event. This property shall be read-write. This property may contain a null reference.
10.9 Cancel property

The Cancel property shall contain a value that indicates whether or not the operation of the event has been canceled. This property shall be read-write. The cancel directive shall be one of the following values:

- Do not cancel (default).
- Cancel the event.
- Cancel the event from cascading to children.
- Cancel the event from cascading to siblings.

10.10 Arguments zproperty

The Arguments property shall contain an indexed array of argument values. This property shall be read-write.
11 ITEMS COLLECTION INTERFACE

An Items collection shall provide access to a list of items.

11.1 Collection Management Services

11.1.1 Item method

The Item method shall return the item at a specified index in the collection. The item index argument shall be either an index number or a name identifying a simulation item. The returned item designator shall be the returned item instance. If the value of item index argument is a number, then this method shall return the item designator at the zero-based index position within the collection. If the numeric value of the item index argument is less than zero or greater than or equal to the number of items, this method shall raise the exception: index out of range. Furthermore, if the index is not a number, then this method shall return the item designator that has its Name property that matches the index. If the item index argument does not match an item’s name in the collection, this method shall raise the exception: item not found. This method shall be the default member of the Items collection.

11.1.1.1 Supplied arguments

a) Item index.

11.1.1.2 Returned arguments

a) Item designator.

11.1.1.3 Preconditions

a) None.

11.1.1.4 Postconditions

a) The item designator has been returned.

11.1.1.5 Exceptions

a) Item not found.

b) Index out of range.

11.1.2 Count property

The Count property shall contain the number of items in the collection. This property shall be read-only. The returned item count shall be the number of items in the collection.

11.1.3 Exists method

The Exists method shall return true if an item exists in the collection. This property shall be read-only. The item index argument shall be either an index number or a name identifying a simulation item. The returned item designator shall be the returned item instance. If the value of item index argument is a number, then this method shall return a success indicator value of true if the item index is greater than zero and less than number of items. Furthermore, if the index is not a number, then this method shall return a success indicator value of true if the item index value matches the Name property of one of the items in the collection. Otherwise this method shall return a success indicator value of false.
11.1.3.1 **Supplied arguments**
   a) Item index.

11.1.3.2 **Returned arguments**
   a) Success indicator.

11.1.3.3 **Preconditions**
   a) None.

11.1.3.4 **Postconditions**
   a) The success indicator has been returned.

11.1.3.5 **Exceptions**
   a) Item not found.
   b) Index out of range.

11.1.4 **IndexOf method**

The IndexOf method shall return the zero-based ordinal position of an item in the collection. The item
designator argument shall be an item instance of which the index number is requested. If the designator
is not in the collection, this method shall raise the exception: item not found.

11.1.4.1 **Supplied arguments**
   a) Item designator.

11.1.4.2 **Returned arguments**
   a) Index number.

11.1.4.3 **Preconditions**
   a) None.

11.1.4.4 **Postconditions**
   a) Index number of the item designator has been returned.

11.1.4.5 **Exceptions**
   a) Item not found.
12 ITEM EVENTS COLLECTION INTERFACE

An Items collection shall provide access to a list of item events.

12.1 Collection Management Services

12.1.1 Item method

The Item method shall return the item event at a specified index in the collection. The item event index argument shall be either an index number or a queued event handle identifying an item event. The returned item event designator shall be the returned item event instance. If the value of item event index argument is a number, then this method shall return the item event designator at the zero-based index position within the collection. If the numeric value of the item index argument is less than zero or greater than or equal to the number of items, this method shall raise the exception: index out of range. Furthermore, if the index is a queued event handle, then this method shall return the item event designator that has its EventID property that matches the index. If the item event index argument does not match an item’s EventID in the collection, this method shall raise the exception: item not found. This method shall be the default member of the ItemEvents collection.

12.1.1.1 Supplied arguments

a) Item event index.

12.1.1.2 Returned arguments

a) Item event designator.

12.1.1.3 Preconditions

a) None.

12.1.1.4 Postconditions

a) The item event designator has been returned.

12.1.1.5 Exceptions

a) Index out of range.

12.1.2 Count property

The Count property shall contain the number of item events in the collection. This property shall be read-only. The returned item event count shall be the number of item events in the collection.
13 COLLECTION INTERFACE

A collection object shall provide access to a list of values of mixed data types by index number or by key value. A key shall be associated with no more than one entry in the collection.

13.1 Collection Management Services

13.1.1 DuplicateKeyAction property

The DuplicateKeyAction property shall contain the action to be performed when an attempt to add a duplicate key to the collection is performed. This property shall be read-write. The action shall be one of the following values:
- Restrict – raise the duplicate key exception (default).
- Cancel – ignore the add operation.
- Leave key – add but without using the key.
- Steal key – add and take the key away from the existing value.
- Replace – replace the existing value that has the key with the new value.

13.1.2 Item method

The Item method shall return the value at a specified index in the collection. The value index argument shall be either an index number or a key value identifying a value in the collection. The returned stored value shall be the value stored at the designated index. If the value of value index argument is a number, then this method shall return the item designator at the zero-based index position within the collection. If the numeric value of the value index argument is less than zero or greater than or equal to the number of items, this method shall raise the exception: index out of range. Furthermore, if the index is not a number, then this method shall return the value stored with the key value that matches the index. If the value index argument does not match an item’s key value in the collection, this method shall raise the exception: item not found. This method shall be the default member of the collection.

13.1.2.1 Supplied arguments

a) Value index.

13.1.2.2 Returned arguments

a) Stored value.

13.1.2.3 Preconditions

a) None.

13.1.2.4 Postconditions

a) The stored value has been returned.

13.1.2.5 Exceptions

a) Item not found.
b) Index out of range.

13.1.3 GetItem method

The GetItem method shall return the value at a specified index in the collection. The value index argument shall be either an index number or a key value identifying a value in the collection. The default
value argument shall be the default value to be returned when the value index does not locate a value in the collection. The default value of the default value argument shall be a null value. The returned stored value shall be the value stored at the designated index. If the value of value index argument is a number, then this method shall return the item designator at the zero-based index position within the collection. If the numeric value of the value index argument is less than zero or greater than or equal to the number of items, this method shall return the default value. Furthermore, if the index is not a number, then this method shall return the value stored with the key value that matches the index. If the value index argument does not match an item’s key value in the collection, this method shall return the default value.

13.1.3.1 Supplied arguments

a) Value index.

b) Default value.

13.1.3.2 Returned arguments

a) Stored value.

13.1.3.3 Preconditions

a) None.

13.1.3.4 Postconditions

a) The item designator has been returned.

13.1.3.5 Exceptions

a) Item not found.

b) Index out of range.

13.1.4 Count property

The Count property shall contain the number of items in the collection. This property shall be read-only. The returned item count shall be the number of items in the collection.

13.1.5 Exists method

The Exists method shall return true if an item exists in the collection. This property shall be read-only. The value index argument shall be either an index number or a key value identifying a simulation item. The returned item designator shall be the returned item instance. If the value of value index argument is a number, then this method shall return a success indicator value of true if the value index is greater than zero and less than number of items. Furthermore, if the index is not a number, then this method shall return a success indicator value of true if the value index value matches the Name property of one of the items in the collection. Otherwise this method shall return a success indicator value of false.

13.1.5.1 Supplied arguments

a) Value index.

13.1.5.2 Returned arguments

a) Success indicator.
13.1.5.3 **Preconditions**
   a) None.

13.1.5.4 **Postconditions**
   a) The success indicator has been returned.

13.1.5.5 **Exceptions**
   a) Item not found.
   b) Index out of range.

13.1.6 **IndexOf method**

The **IndexOf** method shall return the zero-based ordinal position of an item in the collection. The key value argument shall be the key to be found. If the designator is not in the collection, this property shall raise the exception: item not found.

13.1.6.1 **Supplied arguments**
   a) Key value.

13.1.6.2 **Returned arguments**
   a) Index number.

13.1.6.3 **Preconditions**
   a) None.

13.1.6.4 **Postconditions**
   a) Index number of the item designator has been returned.

13.1.6.5 **Exceptions**
   a) Item not found.

13.1.7 **KeyAt method**

The **KeyAt** method shall return the key value at a designated position in the collection. The index number argument shall be the index number in the range of the collection. The returned key value argument shall be the key associated with the designated position.

13.1.7.1 **Supplied arguments**
   a) Index number.

13.1.7.2 **Returned arguments**
   a) Key value.

13.1.7.3 **Preconditions**
   a) None.
13.1.7.4 **Postconditions**

a) Index number of the item designator has been returned.

13.1.7.5 **Exceptions**

a) Index out of range.

13.1.8 **Add method**

The Add method shall add a value to the collection. The data value argument shall be the value to be stored in the collection. The optional key value argument shall be the key to be associated with the data value. The optional before index argument shall be the index number or key value of an item in the collection before which to place the added value. The optional after index argument shall be the index number or key value of an item in the collection after which to place the added value. If both before index and after index arguments are specified, this method shall raise the exception: invalid argument.

13.1.8.1 **Supplied arguments**

a) Data value.
b) Key value.
c) Before index.
d) After index.

13.1.8.2 **Returned arguments**

a) None.

13.1.8.3 **Preconditions**

a) None.

13.1.8.4 **Postconditions**

a) The value has been added to the collection.

13.1.8.5 **Exceptions**

a) Item not found.
b) Index out of range.
c) Out of memory.
d) Invalid argument.

13.1.9 **Remove method**

The Remove method shall remove and return the value at a specified index in the collection. The value index argument shall be either an index number or a key value identifying a value in the collection. The returned stored value shall be the value stored at the designated index. If the value of value index argument is a number, then this method shall return the value at the zero-based index position within the collection. If the numeric value of the value index argument is less than zero or greater than or equal to the number of items, this method shall raise the exception: index out of range. Furthermore, if the index is not a number, then this method shall return the value stored with the key value that matches the index. If the value index argument does not match an item’s name in the collection, this method shall raise the exception: item not found.
13.1.9.1 **Supplied arguments**

a) Value index.

13.1.9.2 **Returned arguments**

a) Stored value.

13.1.9.3 **Preconditions**

a) None.

13.1.9.4 **Postconditions**

a) The value at the index has been removed.
   b) The collection’s count has been decreased by one.
   c) The value at the index has been returned.

13.1.9.5 **Exceptions**

a) Item not found.
   b) Index out of range.

13.1.10 **Clear method**

The Clear method shall remove all the values in the collection.

13.1.10.1 **Supplied arguments**

a) None.

13.1.10.2 **Returned arguments**

a) None.

13.1.10.3 **Preconditions**

a) None.

13.1.10.4 **Postconditions**

a) The collection has been cleared.

13.1.10.5 **Exceptions**

a) Item not found.
   b) Index out of range.
14 ARRAY INTERFACE

An array object shall provide access to a list of values of mixed data types by index number.

14.1 Array Management Services

14.1.1 Item method

The Item method shall return the value at a specified index in the array. The index number argument shall be the position of an entry in the list. The returned stored value shall be the value stored at the designated position. This method shall return the value at the zero-based index position within the array. If the numeric value of the value index argument is less than zero or greater than or equal to the number of items, this method shall raise the exception: index out of range. This method shall be the default member of the array.

14.1.1.1 Supplied arguments

a) Index number.

14.1.1.2 Returned arguments

a) Stored value.

14.1.1.3 Preconditions

a) None.

14.1.1.4 Postconditions

a) The stored value has been returned.

14.1.1.5 Exceptions

a) Index out of range.

14.1.2 Count property

The Count property shall contain the number of items in the array. This property shall be read-only. The returned item count shall be the number of items in the array.

14.1.3 Add method

The Add method shall add a value to the end of the array. The data value argument shall be the value to be stored in the array.

14.1.3.1 Supplied arguments

a) Data value.

14.1.3.2 Returned arguments

a) None.

14.1.3.3 Preconditions

a) None.
14.1.3.4 Postconditions
   a) The value has been added to the array.

14.1.3.5 Exceptions
   a) Out of memory.

14.1.4 Insert method
The Insert method shall insert a value into a designated position in the array. The data value argument shall be the value to be stored in the collection. The index number argument shall be the zero-based location to be associated with the data value. The index number argument is less than zero or greater than the count of entries in the array then this method shall raise the exception: index out of range.

14.1.4.1 Supplied arguments
   a) Data value.
   b) Index number.

14.1.4.2 Returned arguments
   a) None.

14.1.4.3 Preconditions
   a) None.

14.1.4.4 Postconditions
   a) The value has been inserted into the array.

14.1.4.5 Exceptions
   a) Index out of range.
   b) Out of memory.

14.1.5 Remove method
The Remove method shall remove and return the value at a designated position in the array. The index number argument shall be the zero-based position of the value to be removed. The returned stored value shall be the value stored at the designated index number. If the value of the index number argument is less than zero or greater than or equal to the number of items, this method shall raise the exception: index out of range.

14.1.5.1 Supplied arguments
   a) Index number.

14.1.5.2 Returned arguments
   a) Stored value.
14.1.5.3 **Preconditions**

a) None.

14.1.5.4 **Postconditions**

a) The value at the index has been removed.
b) The array’s count has been decreased by one.
c) The value at the index has been returned.

14.1.5.5 **Exceptions**

a) Index out of range.

14.1.6 **Clear method**

The Clear method shall remove all the values in the array.

14.1.6.1 **Supplied arguments**

a) None.

14.1.6.2 **Returned arguments**

a) None.

14.1.6.3 **Preconditions**

a) None.

14.1.6.4 **Postconditions**

a) The array has been cleared.

14.1.6.5 **Exceptions**

a) Index out of range.
15 RANDOM NUMBER GENERATOR INTERFACE

A random number generator object shall provide the capability to generate random numbers for a particular distribution. Instantiation of random generator objects is implementation dependent.

15.1 Random Number Generation Services

15.1.1 RandomValue method

The RandomValue method shall return a new random number each time it is invoked. The returned random number shall be a value within the random number generator’s distribution.

15.1.1.1 Supplied arguments

a) None.

15.1.1.2 Returned arguments

a) Random number.

15.1.1.3 Preconditions

a) None.

15.1.1.4 Postconditions

a) The item event designator has been returned.

15.1.1.5 Exceptions

a) Index out of range.
16 BASIC STATISTICS INTERFACE

A BasicStatistics object shall provide elementary statistics about a set of random numbers.

16.1 Basic Statistics Services

16.1.1 AddValue method

The AddValue method shall add a random number to the set. The random number argument shall be a double precision numeric value.

16.1.1.1 Supplied arguments

a) Random number.

16.1.1.2 Returned arguments

a) None.

16.1.1.3 Preconditions

a) None.

16.1.1.4 Postconditions

a) The random number has been added.

16.1.1.5 Exceptions

a) None.

16.1.2 Mean property

The Mean property shall contain the arithmetic mean of the random numbers added to the set. This property shall be read-only. The average shall be a double precision numeric value.

16.1.3 Variance property

The Variance property shall contain the arithmetic variance of all of the random numbers that were added to the set. This property shall be read-only. The variance shall be a double precision numeric value.

16.1.4 StandardDeviation property

The StandardDeviation property shall contain the arithmetic standard deviation of all of the random numbers that were added to the set. This property shall be read-only. The standard deviation shall be a double precision numeric value.

16.1.5 Count property

The Count property shall contain the quantity of random numbers added to the set. This property shall be read-only. The count shall be an integer value.

16.1.6 Maximum property

The Maximum property shall contain the largest random number that was added to the set. This property shall be read-only. The maximum number shall be a double precision numeric value.
16.1.7 Minimum property

The Minimum property shall contain the smallest random number that was added to the set. This property shall be read-only. The max number shall be a double precision numeric value.

16.1.8 Reset method

The Reset method shall clear the set.

16.1.8.1 Supplied arguments

a) None.

16.1.8.2 Returned arguments

a) None.

16.1.8.3 Preconditions

a) None.

16.1.8.4 Postconditions

a) The set of random numbers has been cleared.

16.1.8.5 Exceptions

a) None.
17  TIME AVERAGE INTERFACE

A TimeAverage object shall provide elementary time average statistics about a set of random numbers and their associated times.

17.1 Time Average Services

17.1.1  AddValue method

The AddValue method shall add a random number to the set at a point in time. The random number argument shall be a double precision numeric value. The current time value argument shall be the time associated with the random number.

17.1.1.1  Supplied arguments

a) Random number.
b) Current time value.

17.1.1.2  Returned arguments

a) None.

17.1.1.3  Preconditions

a) None.

17.1.1.4  Postconditions

a) The random number has been added.

17.1.1.5  Exceptions

a) None.

17.1.2  Initialize method

The Initialize method shall initialize the time average with a random number at a starting point in time. The random number argument shall be a double precision numeric value. The starting time value argument shall be the time associated with the random number.

17.1.2.1  Supplied arguments

a) Random number.
b) Starting time value.

17.1.2.2  Returned arguments

a) None.

17.1.2.3  Preconditions

a) None.

17.1.2.4  Postconditions

a) The random number has been added.
17.1.2.5 **Exceptions**

a) None.

17.1.3 **LastValue property**

The LastValue property shall contain the last random number added to the set. This property shall be read-only. The last value shall be a double precision numeric value.

17.1.4 **CurrentTime property**

The CurrentTime property shall contain the current time value of the time average. This property shall be read-only. The current time shall be a time value.

17.1.5 **Average property**

The Average property shall contain the time average value of the random numbers added to the set. This property shall be read-only. The average shall be a double precision numeric value.

17.1.6 **Count property**

The Count property shall contain the quantity of random numbers added to the set. This property shall be read-only. The count shall be an integer value.

17.1.7 **Maximum property**

The Maximum property shall contain the largest random number that was added to the set. This property shall be read-only. The maximum number shall be a double precision numeric value.

17.1.8 **Reset method**

The Reset method shall clear the set.

17.1.8.1 **Supplied arguments**

a) None.

17.1.8.2 **Returned arguments**

a) None.

17.1.8.3 **Preconditions**

a) None.

17.1.8.4 **Postconditions**

a) The set of random numbers has been cleared.

17.1.8.5 **Exceptions**

a) None.
17.1.9 Skip method

The Skip method shall set the point in time without providing a value. The current time value argument shall be the new current time.

17.1.9.1 Supplied arguments

a) Current time value.

17.1.9.2 Returned arguments

a) None.

17.1.9.3 Preconditions

a) None.

17.1.9.4 Postconditions

a) The current time has been set.

17.1.9.5 Exceptions

a) None.
18 ERROR INTERFACE

An Error object shall contain information about simulation exceptions. An SRML engine shall create instances of this interface when unhandled exceptions are generated.

18.1 Description Property

The Description property shall contain the text description of the exception. This property shall be read-only.

18.2 XPath PROPERTY

The XPath property shall contain an XPath expression that locates the node that generated the error. This property shall be read-only.

18.3 Script Property

The Script property shall contain the portion of script that produced the error. This property shall be read-only.

18.4 Number Property

The Number property shall contain the platform-specific error number. This property shall be read-only.

18.5 Line Property

The Line property shall contain the line number of script that produced the error. This property shall be read-only.

18.6 Column Property

The Column property shall contain the column number of script that produced the error. This property shall be read-only.

18.7 Clear Method

The Clear method shall clear the error.

18.7.1 Supplied arguments

a) None.

18.7.2 Returned arguments

a) None.

18.7.3 Preconditions

a) None.

18.7.4 Postconditions

a) The error has been cleared.
18.7.5 **Exceptions**

a) None.
19 ARGUMENT SERIALIZER INTERFACE

An ArgumentSerializer object shall provide the capability to load and save item event arguments of varying data types. Instantiation of an ArgumentSerializer is implementation dependent.

19.1 Argument Serialization Services

19.1.1 Load method

The Load method shall return the value of an item event argument. The argument node designator argument shall be the DOM object reference of the argument node of the item event to be loaded. The parameter node designator argument shall be the DOM object reference of the parameter node of the event class of the item event to be loaded. The returned argument value shall be the runtime value of the argument.

19.1.1.1 Supplied arguments

a) Argument node designator.
b) Parameter node designator.

19.1.1.2 Returned arguments

a) Argument value.

19.1.1.3 Preconditions

a) None.

19.1.1.4 Postconditions

a) The argument value has been returned.

19.1.1.5 Exceptions

a) Type mismatch.

19.1.2 Save method

The Save method shall store the runtime value of an item event argument. The argument node designator argument shall be the DOM object reference of the argument node of the item event to be loaded. The parameter node designator argument shall be the DOM object reference of the parameter node of the event class of the item event to be loaded. This method shall place the argument value to be stored into the value attributed of the argument node designator.

19.1.2.1 Supplied arguments

a) Argument node designator.
b) Parameter node designator.
c) Argument value.

19.1.2.2 Returned arguments

a) None.
19.1.2.3 **Preconditions**

a) None.

19.1.2.4 **Postconditions**

a) The argument value has been stored.

19.1.2.5 **Exceptions**

a) Type mismatch.
**ANNEX A – TABLE OF RANDOM NUMBER MULTIPLIERS**

<table>
<thead>
<tr>
<th>Multiplier Index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>950706376</td>
</tr>
<tr>
<td>2</td>
<td>742938285</td>
</tr>
<tr>
<td>3</td>
<td>1226874159</td>
</tr>
<tr>
<td>4</td>
<td>62089911</td>
</tr>
<tr>
<td>5</td>
<td>1343714438</td>
</tr>
<tr>
<td>6</td>
<td>630360016</td>
</tr>
<tr>
<td>7</td>
<td>397204094</td>
</tr>
<tr>
<td>8</td>
<td>16807</td>
</tr>
<tr>
<td>9</td>
<td>2307085864</td>
</tr>
<tr>
<td>10</td>
<td>44485709377909</td>
</tr>
</tbody>
</table>
ANNEX B – SRML ENGINE IDL BINDING

Listing 1: SRML IDL Binding:

/* SRML Interface Definition

Revision History:

Initial 1.0 2011-12-30
Steven W. Reichenthal
*/

interface SRCascadeConstants
{
    const short srmlCascadeDefault = 0;
    const short srmlCascadeNone = 1;
    const short srmlCascadeUp = 2;
    const short srmlCascadeDown = 4;
};

interface SREventFormatConstants
{
    const short srmlEventFormatDefault = 0;
    const short srmlEventFormatMethod = 1;
    const short srmlEventFormatItemEvent = 2;
};

interface SRDispatchConstants
{
    const short srmlDispatchSend = 1;
    const short srmlDispatchPost = 2;
    const short srmlDispatchSchedule = 3;
};

interface SRCancelEventConstants
{
    const short srmlCancelUnspecified = 0;
    const short srmlCancelEvent = 1;
    const short srmlCancelEventChildren = 2;
    const short srmlCancelEventSiblings = 4;
};

interface SRMemberTypeConstants
{
    const short srmlMemberTypeUnspecified = 0;
    const short srmlMemberTypeScript = 1;
    const short srmlMemberTypeProperty = 2;
    const short srmlMemberTypeLink = 3;
    const short srmlMemberTypeLinksItem = 4;
    const short srmlMemberTypeLinksCollection = 5;
    const short srmlMemberTypeIDREF = 6;
    const short srmlMemberTypeIDREFsItem = 7;
    const short srmlMemberTypeIDREFsCollection = 8;
    const short srmlMemberTypeItemMember = 9;
    const short srmlMemberTypeID = 10;
    const short srmlMemberTypeLocalID = 11;
    const short srmlMemberTypeName = 12;
    const short srmlMemberTypeLocationFixed = 13;
    const short srmlMemberTypeEventSink = 14;
    const short srmlMemberTypeLinkProperty = 15;
    const short srmlMemberTypeLinksProperty = 16;
    const short srmlMemberTypeLinksPropertyItem = 17;
interface SRNotificationConstants
{
    const short srmlNotificationNone = 0,
    const short srmlNotificationItemCreated = 1;
    const short srmlNotificationItemDeleted = 2;
    const short srmlNotificationItemDestroyed = 4;
    const short srmlNotificationItemPropertyChanged = 8;
    const short srmlNotificationItemLocationChanged = 16;
    const short srmlNotificationEventOccurred = 32;
    const short srmlNotificationLoadItemSource = 64;
    const short srmlNotificationEventSent = 128;
    const short srmlNotificationBeforeEvent = 256;
    const short srmlNotificationEventScheduled = 512;
    const short srmlNotificationEventPosted = 1024;
    const short srmlNotificationEventRetracted = 2048;
    const short srmlNotificationStateChanged = 4096;
    const short srmlNotificationForwardEventsToMainScript = 8192;
};

interface SRLoadConstants
{
    const short srmlLoadDefault = 0;
    const short srmlLoadEvents = 1;
    const short srmlLoadItems = 2;
};

interface SRSaveConstants
{
    const short srmlSaveDefault = 0;
    const short srmlSaveEvents = 1;
    const short srmlSaveResolveItemIDs;
    const short srmlSaveDOMOnly = 4;
    const short srmlSaveNoItemIDs = 8;
    const short srmlSaveNoItems = 16
};

interface SRArgumentLoadSaveConstants
{
    const short srmlArgumentLoadSaveCancel = 1;
    const short srmlItemEventLoadSaveManualArguments = 2;
};

interface SRBroadcastConstants
{
    const short srmlBroadcastSend = 1;
    const short srmlBroadcastPost = 2;
};

interface SRStateConstants
{
    const short srmlStateUnloaded = 10;
    const short srmlStateLoading = 20;
    const short srmlStateLoaded = 30;
    const short srmlStateStarted = 40;
    const short srmlStateEnded = 50;
    const short srmlStateAborted = 60;
    const short srmlStateError = 70;
};

interface ItemEvent
attribute SRCascadeConstants Cascade;
attribute SREventFormatConstants Format;
attribute SRCancelEventConstants Cancel;
attribute Node? Node;
attribute DOMString Name;
attribute DateTime Time;
attribute short Priority;
attribute object? Source;
attribute object? Target;
readonly attribute any EventHandle;
attribute any[] Arguments;
readonly attribute unsigned long ArgumentCount;
readonly attribute DOMString EventID;
};

interface Simulator
{
    Simulation CreateSimulation();
};

interface Items
{
    object Item(DOMString Index);
    object Item(long Index);
    boolean Exists(DOMString Index);
    boolean Exists(long Index);
    attribute long Count;
    object GetEnumerator();
};

typedef DOMString Handle;

interface ItemIntrinsic
{
    readonly attribute DOMString ItemID;
    attribute object Object;
    readonly attribute Items Items;
    readonly attribute DOMString[] Links;
    readonly attribute DOMString[] EventSinks;
    readonly attribute object? ItemClass;
    readonly attribute object Simulation;
    readonly attribute object Self;
    readonly attribute Node Node;
    readonly attribute object Location;
    readonly attribute object Script;
    readonly attribute ItemEvents ItemEvents;
    object Item(DOMString Index);
    object Item(long Index);
    object GetItem(DOMString Index, object Default);
    object GetItem(long Index, object Default);
    object? FindItem(VARIANT Query, optional boolean Cache);
    Items FindItems(DOMString Query, boolean Cache);
    object CreateItem(object Prototype, optional object Location, optional any Data);
    Items CreateItems(object Prototype, optional long Quantity = 1, optional object Location, optional any Data);
    any SendEvent(object Target, any Event, any... Arguments);
    any SendItemEvent(object Target, any Event, SRCascadeConstants Cascade, any... Arguments);
    ItemEvent CreateItemEvent(object Target, DOMString Name, any... Arguments);
    Handle PostEvent(object Target, any Event, any... Arguments);
    Handle PostItemEvent(object Target, any Event, SRCascadeConstants Cascade, any... Arguments);
    Handle ScheduleEvent(object Target, any Event, DateTime, any... Arguments);
Handle ScheduleItemEvent(object Target, any Event, SRCascadeConstants Cascade, Date Time, any... Arguments);
Handle ScheduleBroadcast(object Target, any Event, SRCascadeConstants Cascade, Date Time, any... Arguments);
any BroadcastEvent(SRDispatchConstants Dispatch, object Target, SRBroadcastConstants Broadcast, any Event, SRCascadeConstants Cascade, any... Arguments);
Handle DispatchEvent(SRDispatchConstants Dispatch, ItemEvent Event);
object GetEnumerator();
}

interface SimulationIntrinsic : ItemIntrinsic
{
readonly attribute SRStateConstants State;
attribute Document Document;
readonly attribute ItemCount;
attribute Items ItemClasses;
Handle[] ItemIDs(long Index);
void Load(DOMString Filename, optional any Data, optional SRLoadSaveConstants Flags);
void LoadXML(DOMString XML, optional any Data, optional SRLoadSaveConstants Flags);
void LoadItems(optional any Data, optional SRLoadSaveConstants Flags);
void DeleteItem(object Item);
void DestroyItem(object Item);
void ReleaseAll();
void Save(DOMString Filename, optional SRLoadSaveConstants Flags);
attribute IArgumentSerializer ArgumentSerializer;
readonly attribute Error;
boolean IsItemLocationFixed(object Item, boolean Result);
void SetItemLocation(object Item, object Location);
Handle AddLink(object Owner, optional object Target, optional DOMString Name, optional DOMString LinksName, optional boolean IDs, optional DOMString EventClasses, optional DOMString EventMethod, optional DOMString ItemClasses, optional DOMString WithPropertiesChanged, optional DOMString PropertyMethod, optional DOMString WithLocationChanged, optional DOMString LocationMethod, optional boolean WithItemLifetime, optional DOMString CreatedMethod, optional DOMString DestroyedMethod, optional boolean Enabled = true, optional any Before, optional any After);
Node GetLinkNode(DOMString LinkID);
void RemoveLink(Handle LinkID);
any InvokeMethod(object Item, DOMString Method, any... Arguments);
boolean MemberExists(object Item, DOMString MemberName);
void AddProperty(object Item, DOMString PropertyName, optional DOMString DataType, optional any Value);
void RemoveMember(object Item, DOMString MemberName);
void AddExtension(object Extension, optional any Name, boolean AddMembers, optional any Before, optional any After);
void RemoveExtension(any Index);
void ClearExtensions();
attribute Date CurrentTime();
attribute Date EndTime;
Items EventClasses();
void SetEventClassPriority(DOMString EventClass, short Priority);
void SetEventClassPriority(object EventClass, short Priority);
ItemEvent RetractEvent(Handle EventHandle);
void RetractAllEvents(optional any Target);
long EventCount();
ItemEvent GetEvent(long Index);
ItemEvent PeekEvent(optional long Index);
ItemEvent DoNextEvent();
void Run();
void Abort();
ItemEvent CurrentEvent();
any ForwardEvent(ItemEvent Event, object Object, optional boolean IncludeItemClassPrefix = true);
Handle AddEventSink(object OwnerItem, optional DOMString Name, optional DOMString EventClasses, optional DOMString EventMethod, optional DOMString ItemClasses, optional any[] SourceItems, optional DOMString WithPropertiesChanged, optional DOMString PropertyMethod, optional any[] WithLocationChanged, optional DOMString LocationMethod, optional any[] WithItemLifetime, optional DOMString CreatedMethod, optional DOMString DestroyedMethod, optional boolean Enabled = true, optional any Before, optional any After);
Node GetEventSinkNode(DOMString EventSinkID);
void RemoveEventSink(Handle EventSink);
attribute double RandomSeed;
void RandomizeSeed();
RandomNumberGenerator GetRandomNumberGenerator(DOMString Name, any... Arguments);
RandomNumberGenerator CreateRandomUniform(double LowerBound, double UpperBound, optional long Seed, optional long Multiplier = 1);
RandomNumberGenerator CreateRandomExponential(double Mean, optional long Seed, optional long Multiplier = 1);
RandomNumberGenerator CreateRandomTriangular(double Min, double Mode, double Max, optional long Seed, optional long Multiplier = 1);
RandomNumberGenerator CreateRandomNormal(double Mean, double StandardDeviation, optional long Seed, optional long Multiplier = 1);
TimeAverate CreateTimeAverage();
BasicStatistics CreateBasicStatistics();
Array CreateArray(Array... Arguments);
Collection CreateCollection(any... Arguments);

interface IArgumentSerializer
{
void Save(Node nodeArgument, Node nodeParameter, any Data);
any Load(Node nodeArgument, node nodeParameter);
};

interface BooleanBox
{
attribute boolean Value;
};

interface ObjectBox
{
attribute Object Value;
};

interface RandomNumberGeneratorBox
{
attribute RandomNumberGenerator Value;
};

interface ISimulationNotifications
{
void EventOccurred(ItemEvent Event);
nvoid ItemCreated(DOMString ItemID);
nvoid BeforeItemDeleted(DOMString ItemID, BooleanBox Cancel);
nvoid ItemDestroyed(DOMString ItemID);
nvoid ItemPropertyChanged(object Item, DOMString PropertyName, any NewValue);
nvoid ItemLocationChanged(object Item, object PreviousLocation);
nvoid LoadRandomNumberGenerator(DOMString Type, any[] Arguments, RandomNumberGeneratorBox RandomNumberGeneratorBox);
void LoadItemSource(IXMLDOMNode ode, DOMString Type, DOMString Source, ObjectBox Source);
void BeforeEvent(object Item, DOMString Event, any[] Arguments);
void BeforeItemEvent(ItemEvent Event);
void CurrentTimeSet(Date PreviousTime);
void EventScheduled(ItemEvent Event);
void EventPosted(ItemEvent Event);
void ItemEventSent(ItemEvent Event);
void EventSent(object Item, DOMString Event, any[] Arguments);
void EventRetracted(ItemEvent Event);
void Idle(BooleanBox Cancel);
void BeginProgress(DOMString Activity, long Range, BooleanBox Cancel);
void Progress(DOMString Activity, long iActivity, BooleanBox Cancel);
void EndProgress();
void StateChanged(SRStateConstants NewState);
};

interface Error
{
    readonly attribute long Number;
    readonly attribute DOMString XPath;
    readonly attribute DOMString Description;
    readonly attribute long Line;
    readonly attribute long Column;
    readonly attribute DOMString Script;
    HRESULT Clear();
};